



especialista universitario en
Tecnologías Java Enterprise
para Aplicaciones Web

Servidores de aplicaciones

Índice

1. INTRODUCCIÓN A LOS SERVIDORES DE APLICACIONES	3
1.1. APLICACIONES DE EMPRESA	3
1.2. ¿QUÉ ES UN SERVIDOR DE APLICACIONES?	3
2. INSTALACIÓN DEL SERVIDOR BEA WEBLOGIC	8
2.1. INSTALACIÓN DEL SERVIDOR DE APLICACIONES ..	8
2.2. ARRANQUE DEL DOMINIO Y CONSOLA DE ADMINISTRACIÓN	12
2.3. ARRANQUE Y CONFIGURACIÓN	18
3. ADMINISTRACIÓN DEL SERVIDOR DE APLICACIONES	28
3.0. REVISIÓN DE CONCEPTOS	28
3.1. CREACIÓN Y CONFIGURACIÓN DE UN NUEVO DOMINIO	29
3.2. USO DEL NODEMANAGER	41
3.3. DESPLIEGUE DE APLICACIONES	46
3.4. GESTIÓN DE SEGURIDAD	54
3.5. ADMINISTRACIÓN DESDE LÍNEA DE COMANDOS	60
4. CREACIÓN DE UN CLUSTER	64
4.1. CONFIGURACIÓN BÁSICA DE UN CLUSTER	64
4.2. CONFIGURACIÓN DE UN SERVIDOR PROXY	69
4.3. CONFIGURACIÓN DE LA REPLICACIÓN DE MEMORIA	72
5. JNDI	74
5.1. JNDI: BÚSQUEDA DE OBJETOS MEDIANTE SU NOMBRE LÓGICO	74
5.2. PROGRAMAR CON JNDI	75

5.3. WEBLOGIC Y JNDI.....	76
5.4. CLASES DE ARRANQUE Y PARADA.....	77

6. ACCESO A BASES DE DATOS CON WEBLOGIC	79
--	-----------

6.1. CONFIGURACIÓN DE LAS FUENTES DE DATOS Y EL POOL DE CONEXIONES.....	79
---	----

6.2. USO DE LAS FUENTES DE DATOS EN UNA APLICACIÓN	87
--	----

EJERCICIOS	89
-------------------	-----------

Tema 1: Introducción a los servidores de aplicaciones

En este tema veremos una pequeña introducción a los servidores de aplicaciones. Comentaremos los términos más utilizados y los conceptos que usaremos más adelante.

1.1. Aplicaciones de empresa

El concepto de servidor de aplicaciones está relacionado con el concepto de sistema distribuido. Un sistema distribuido, en oposición a un sistema monolítico, permite mejorar tres aspectos fundamentales en una aplicación: la alta disponibilidad, la escalabilidad y el mantenimiento. En un sistema monolítico un cambio en las necesidades del sistema (aumento considerable del número de visitas, aumento del número de aplicaciones, etc.) provoca un colapso y la adaptación a dicho cambio puede resultar catastrófica. Vamos a ver estas características con ejemplos.

- La **alta disponibilidad** hace referencia a que un sistema debe estar funcionando las 24 horas de día los 365 días al año. Para poder alcanzar esta característica es necesario el uso de técnicas de balanceo de carga y de recuperación ante fallos (*failover*).
- La **escalabilidad** es la capacidad de hacer crecer un sistema cuando se incrementa la carga de trabajo (el número de peticiones). Cada máquina tiene una capacidad finita de recursos y por lo tanto sólo puede servir un número limitado de peticiones. Si, por ejemplo, tenemos una tienda que incrementa la demanda de servicio, debemos ser capaces de incorporar nuevas máquinas para dar servicio.
- El **mantenimiento** tiene que ver con la versatilidad a la hora de actualizar, depurar fallos y mantener un sistema. La solución al mantenimiento es la construcción de la lógica de negocio en unidades reusables y modulares.

1.2. ¿Qué es un servidor de aplicaciones?

El estándar J2EE permite el desarrollo de aplicaciones de empresa de una manera sencilla y eficiente. Una aplicación desarrollada con las tecnologías J2EE permite ser desplegada en cualquier servidor de aplicaciones o servidor web que cumpla con el estándar. Un servidor de aplicaciones es una implementación de la especificación J2EE. La arquitectura J2EE es la siguiente:

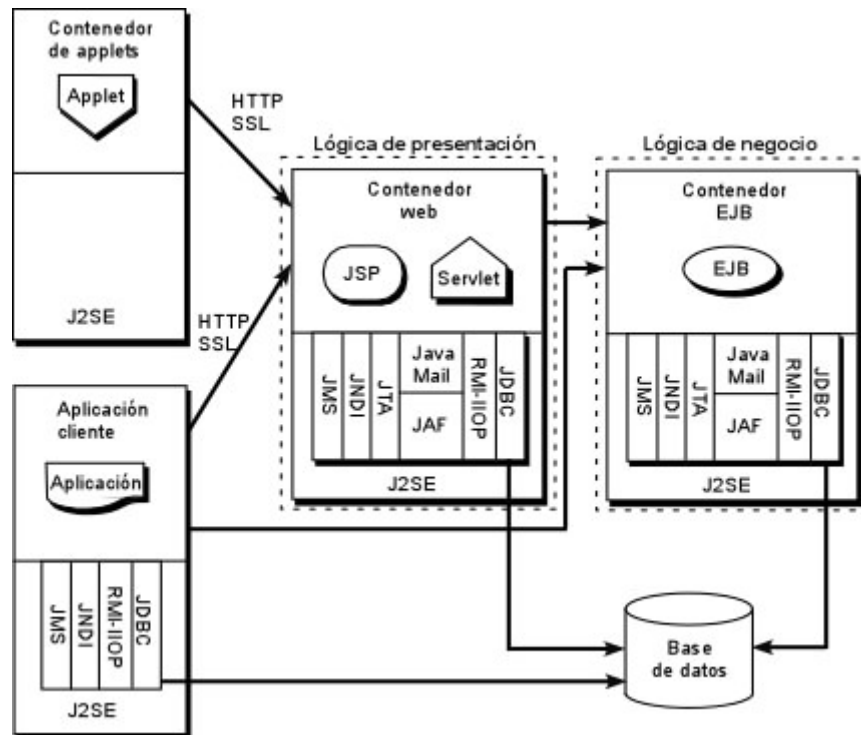


Figura 1. Arquitectura J2EE.

Definimos a continuación algunos de los conceptos que aparecen en la figura 1:

- **Cliente web:** Es usualmente un navegador e interactúa con el contenedor web haciendo uso de HTTP. Recibe páginas HTML o XML y puede ejecutar applets y código JavaScript.
- **Aplicación cliente:** Son clientes que no se ejecutan dentro de un navegador y pueden utilizar cualquier tecnología para comunicarse con el contenedor web o directamente con la base de datos.
- **Contenedor web:** Es lo que comúnmente denominamos servidor web. Es la parte *visible* del servidor de aplicaciones. Utiliza los protocolos HTTP y SSL (seguro) para comunicarse.
- **Servidor de aplicaciones:** Proporciona servicios que soportan la ejecución y disponibilidad de las aplicaciones desplegadas. Es el corazón de un gran sistema distribuido.

Frente a la tradicional estructura en dos capas de un servidor web (ver Figura 2) un servidor de aplicaciones proporciona una estructura en tres capas que permite estructurar nuestro sistema de forma más eficiente. Un concepto que debe quedar claro desde el principio es que no todas las aplicaciones de empresa necesitan un servidor de aplicaciones para funcionar. Una pequeña aplicación que acceda a una base de datos no muy compleja y que no sea distribuida probablemente no necesitará un servidor de aplicaciones, tan solo con un servidor web (usando servlets y jsp) sea suficiente.

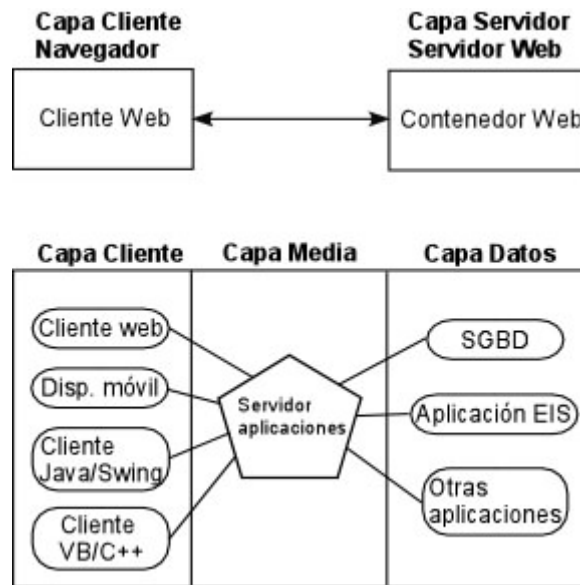


Figura 2. Arquitectura en dos capas frente a tres capas utilizando el servidor de aplicaciones.

Como hemos comentado, un servidor de aplicaciones es una implementación de la especificación J2EE. Existen diversas implementaciones, cada una con sus propias características que la pueden hacer más atractiva en el desarrollo de un determinado sistema. Algunas de las implementaciones más utilizadas son las siguientes:

- BEA WebLogic
- IBM WebSphere
- Sun-Netscape IPlanet
- Oracle IAS
- Borland AppServer
- HP Bluestone

Los dos primeros son los más utilizados en el mercado. Nosotros vamos a utilizar el servidor BEA WebLogic. La principal ventaja de WebLogic es que podemos crear un sistema con varias máquinas con distintos sistemas operativos: Linux, Unix, Windows NT, etc. El sistema funciona sin importarle en qué máquina está corriendo el servidor. En la versión 7.1 WebLogic presenta las siguientes compatibilidades:

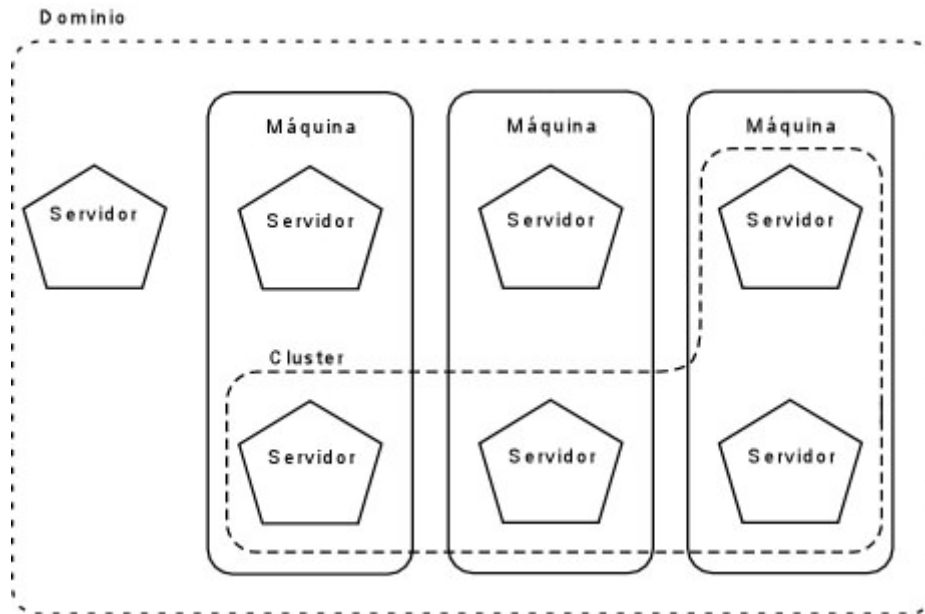
Estándar	Versión
HTTP	1.1
Arquitectura de conector J2EE	1.0
EJB	2.0
JDBC	2.0
JNDI	1.2
JSP	1.2

JTA	1.0.1a
JMS	1.0.2b
RMI	1.0
RMI/IIOP	1.0
Servlet	2.3
LDAP	2
SSL	3
X.509	3

Otros conceptos que aparecerán a lo largo de este módulo:

- **Servidor proxy:** Centraliza peticiones de los clientes y las reenvía hacia otras máquinas. Puede servir como nivel de indirección y seguridad. También puede ser usado para realizar balanceo de carga.
- **Cortafuegos (firewall):** Proporciona servicios de filtrado, autorización y autenticación. Puede actuar como proxy y ayuda a manejar los ataques de los *hackers*.
- **Máquina:** Representa una unidad física donde reside un servidor. Una máquina se define como tipo Unix o no Unix (Windows NT, etc.).
- **Servidor:** Un servidor es una instancia de la clase *weblogic.Server* ejecutándose dentro de una máquina virtual de Java. Un servidor está alojado en una máquina, pero una máquina puede contener varios servidores. Si un servidor no lo declaramos en ninguna máquina WLS asume que está en una creada por defecto.
- **Dominio:** Un dominio es una unidad administrativa. Sirve para declarar varios servidores, aplicaciones, etc. y que todos ellos estén asociados mediante el nombre del dominio.
- **Clustering (asociación):** Los *clusters* permiten asociar máquinas y servidores para que actúen de forma conjunta como una única instancia. La creación de un cluster va a permitir el balanceo de carga y la recuperación frente a fallos.
- **Balanceo de carga:** Es una técnica utilizada para distribuir las peticiones entre varios servidores de tal forma que todos los servidores respondan al mismo número de peticiones.
- **Recuperación ante fallos (failover):** Permite evitar la caída de un sistema cuando una máquina deja de funcionar o funciona incorrectamente.
- **Puerto de escucha:** Un servidor tiene varios puertos por los que puede "escuchar" las peticiones. Existen puertos ya asignados a aplicaciones concretas, como por ejemplo el puerto de http que suele ser el 80. Los puertos permiten que varias aplicaciones puedan atender distintas peticiones en la misma máquina. Un puerto en una dirección se especifica de la siguiente manera: *http://localhost:7001/direc* . Con :7001 indicamos el puerto que estamos *atacando*. Los puertos del 0 al 1023 son reservados por el sistema. Podemos disponer de los puertos del 1024 al 65536. Hay que tener en cuenta que dos servicios no pueden estar escuchando en el mismo puerto.

- **Modo producción y modo desarrollo.** Hablaremos muy a menudo de modo desarrollo y modo producción. El modo desarrollo es cuando nos encontramos desarrollando nuestra aplicación y no está disponible exteriormente. El modo producción es cuando está funcionando a pleno rendimiento y tenemos clientes que se encuentran utilizándola. Por defecto, un dominio se arranca en modo desarrollo.



Tema 2: Instalación del servidor Bea WebLogic

Los pasos a seguir en la instalación de un servidor de aplicaciones es la siguiente:

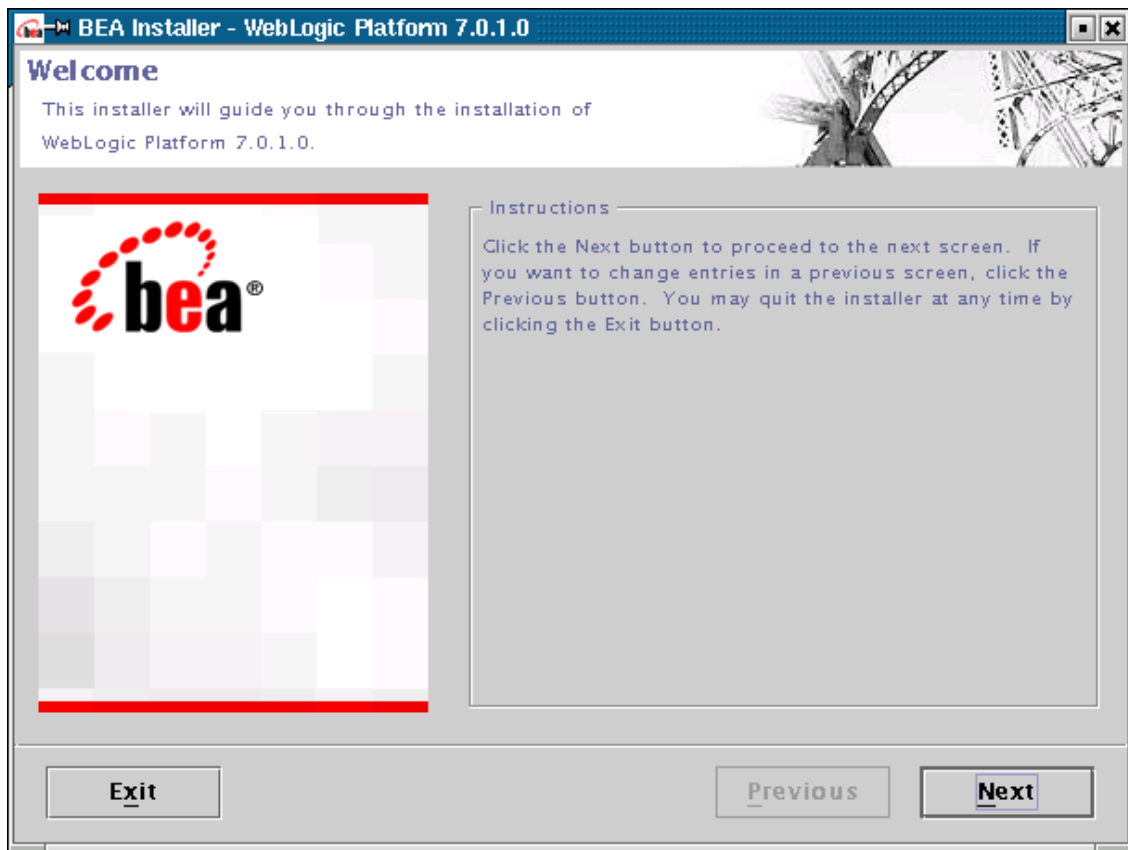
1. Instalación del software. Esta acción copia los ficheros necesarios y crea la estructura inicial de directorios.
2. Configuración de dominios. Debemos configurar el o los dominios necesarios y todos los componentes dentro de cada dominio (servidores, cluster, máquinas, etc.).

2.1. Instalación del servidor de aplicaciones

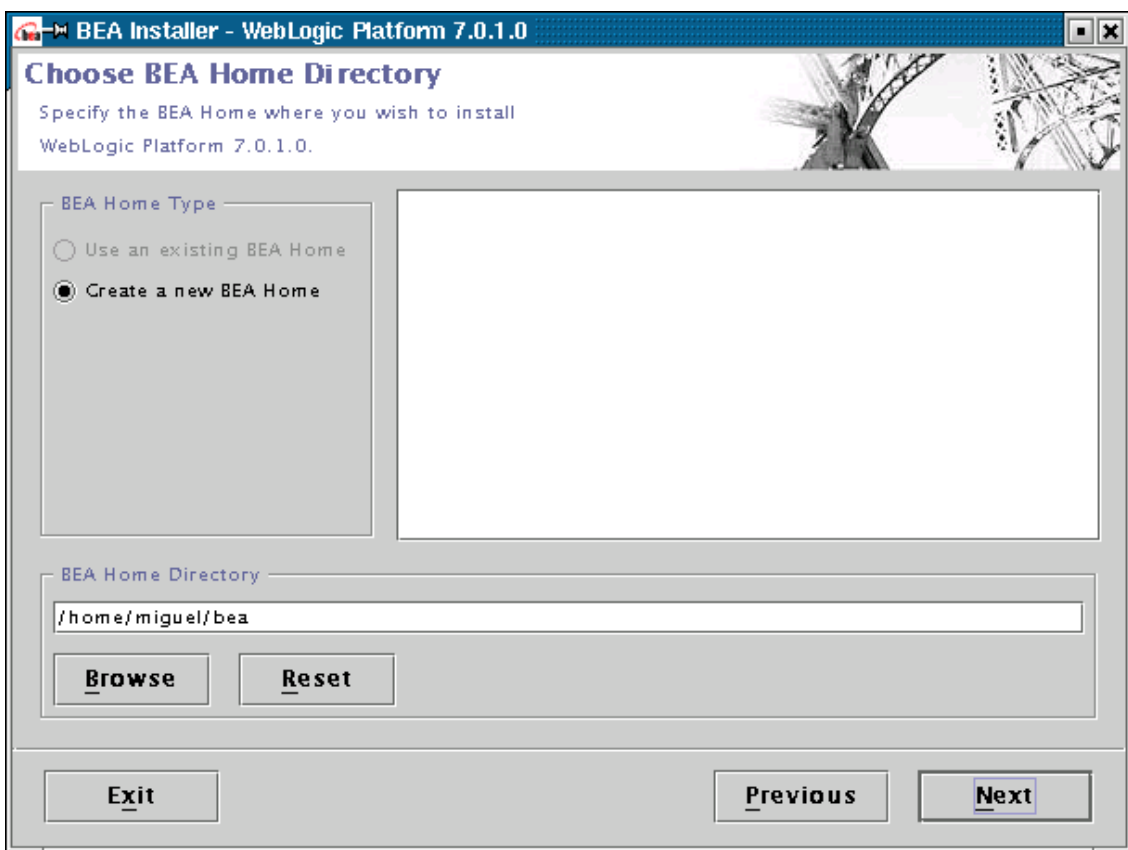
Vamos a instalar el servidor de aplicaciones Bea WebLogic. La instalación descrita aquí es para la versión 7.0 y bajo el sistema operativo Linux. Los requerimientos del sistema para la instalación de esta versión son:

- Memoria: 256Mb mínimo (512Mb aconsejable)
- Espacio en disco: 400Mb
- Versión de Java JDK 1.3.1 (se instala junto con el servidor). Podemos utilizar otra versión de Java, pero es aconsejable consultar la información que Bea muestra en <http://e-docs.bea.com/wls/certifications/certifications/index.html> para comprobar la compatibilidad entre versión.

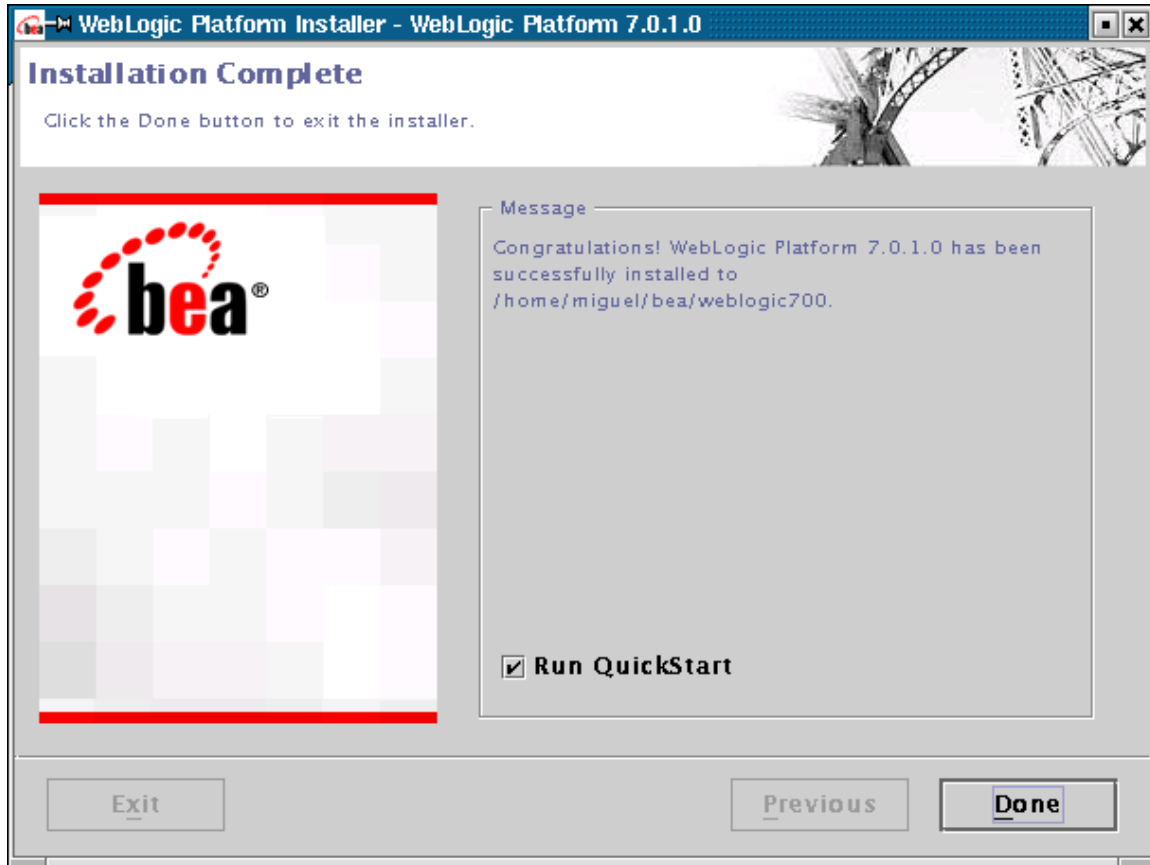
Ejecutamos el fichero *platform701_linux.bin* (no es necesario ser superusuario para instalar el servidor de aplicaciones). Esperamos hasta que nos aparezca la siguiente pantalla.



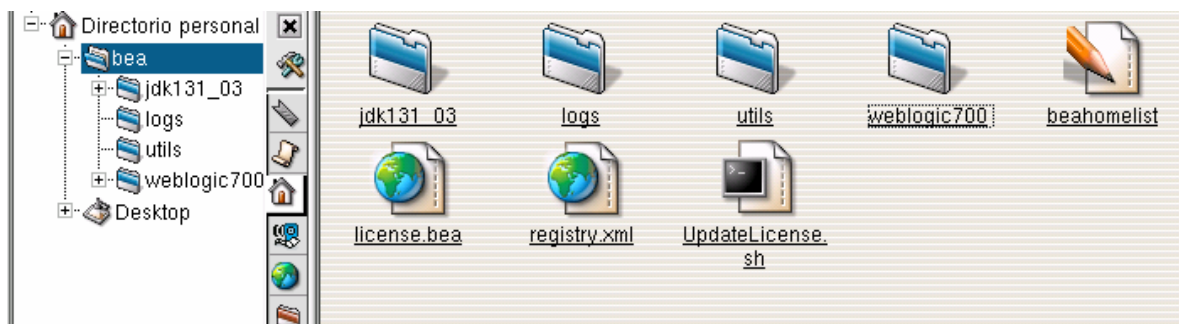
Pinchamos en *Next*. Si ya disponemos de un directorio creado lo podemos elegir de la lista. Si no podemos dejar el mostrado por defecto.



En la pantalla siguiente elegimos la instalación típica y después el directorio por defecto (weblogic700). Empezará la instalación que durará unos minutos. Cuando finalice nos aparecerá la ventana siguiente. Deseleccionamos la opción *Run QuickStart* y pinchamos en *Done*. Hemos finalizado la instalación del servidor.



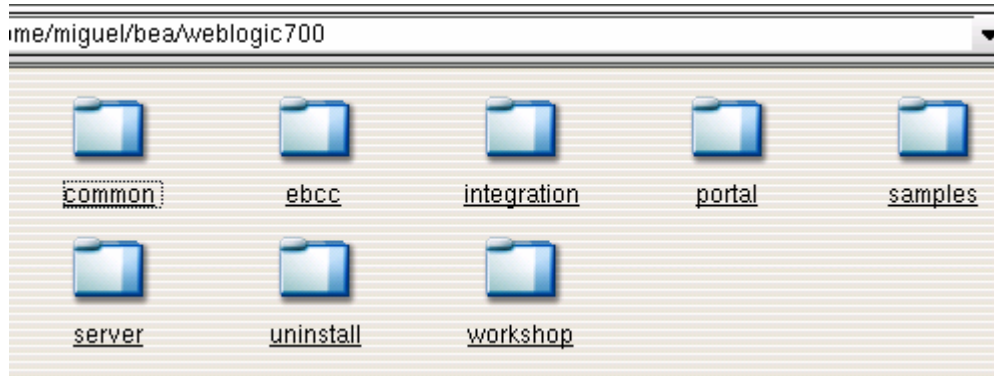
La estructura de directorios creada en la instalación es la siguiente:



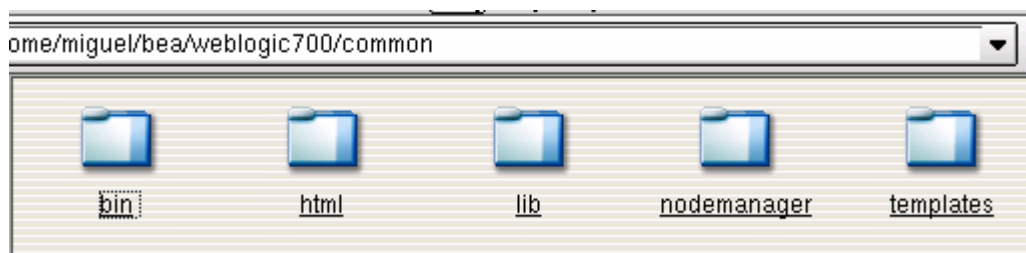
El directorio *jdk131_03* contiene la distribución 1.3.1 de J2SE de Sun. Si tenemos una versión actualizada de Java simplemente la añadiremos al CLASSPATH. En este punto debemos tener un cuidado especial y comprobar si la versión de Java es soportada por la versión del servidor de aplicaciones. Para comprobarlo visitar la página de Bea. El directorio de *logs* contiene el fichero log de instalación. El directorio *utils* contiene algunas utilidades que iremos viendo conforme las utilicemos. El siguiente directorio, *weblogic700*, es el que contiene todas las librerías, clases y herramientas adicionales para el

funcionamiento de nuestro servidor. El fichero *license.bea* contiene la información de nuestra licencia en formato XML. Contendrá información de la fecha de expiración de la licencia, de qué características disponemos (número de puestos, número de IPs, etc.), y toda la información necesaria para la ejecución del servidor.

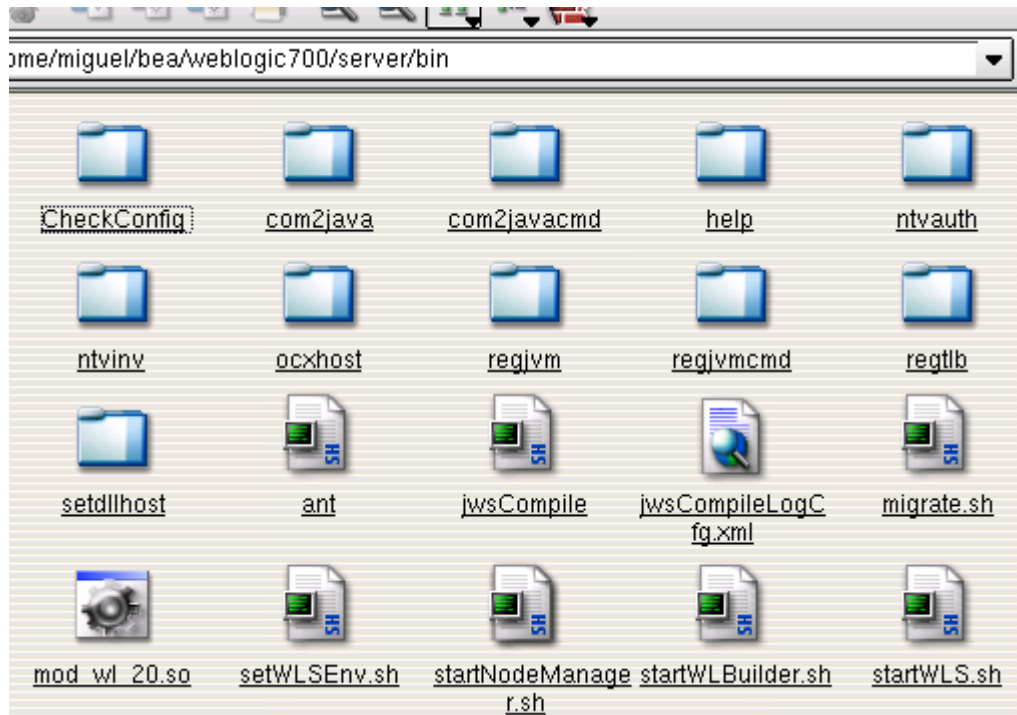
A su vez, el directorio *weblogic700* contiene los siguientes subdirectorios:



Nos interesan, de momento, el directorio *common* y el *server*. *Common* contiene los subdirectorios mostrados en la siguiente figura. En el directorio *bin* tenemos una herramienta para crear dominios. El directorio *nodemanager* contiene ficheros de configuración para el Node Manager.



El directorio *server* contiene datos y utilidades relacionadas con el servidor de aplicaciones. En el directorio *bin* (mostrado en la siguiente figura) tenemos varias aplicaciones y los *scripts* para arrancar el servidor de aplicaciones y el Node Manager. El ejecutable para arrancar un servidor que se crea en nuestro dominio llama a estos ejecutables. En otro directorio dentro de *server*, el subdirectorio *lib*, tenemos el fichero *weblogic.jar* que tendremos que incluir en el *classpath* cuando queramos realizar una aplicación que utilice los recursos de WebLogic. También disponemos en este directorio de los ficheros que gestionan las políticas de seguridad.



2.2. Arranque del dominio y consola de administración

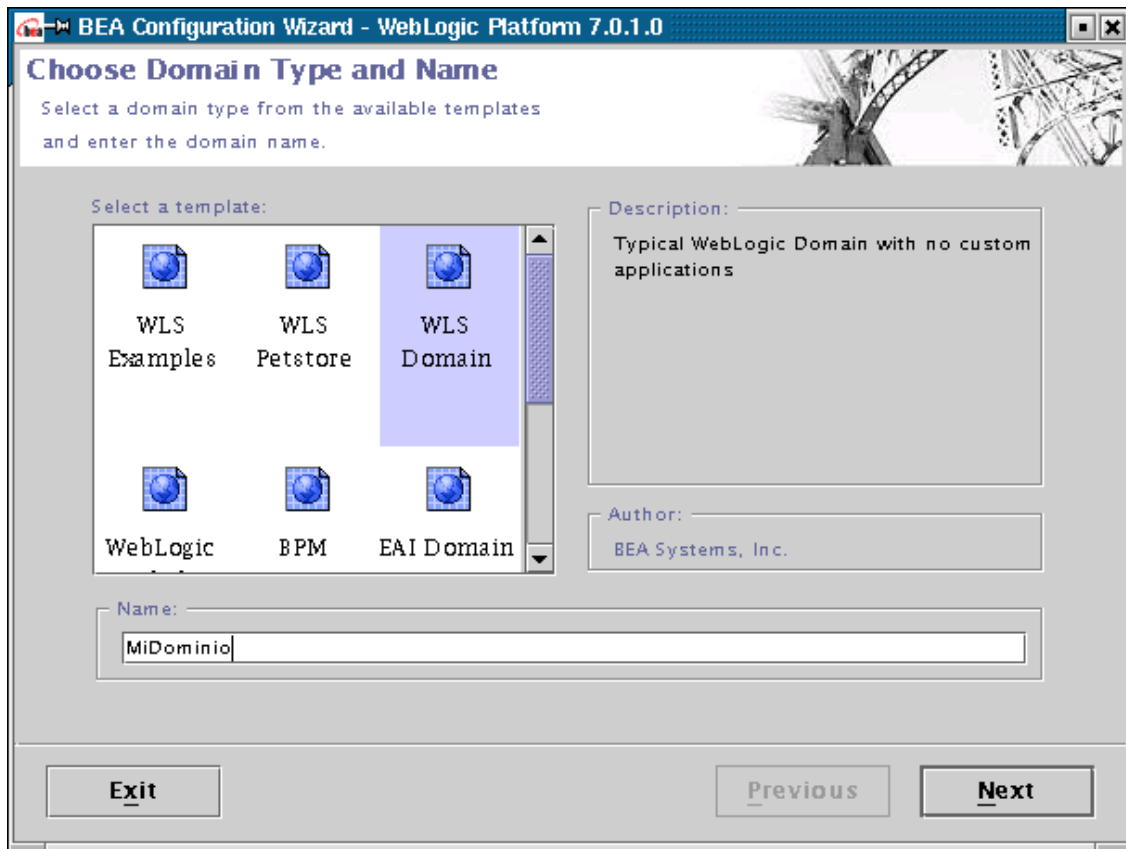
Antes de empezar a definir los elementos que soportan la ejecución del servidor de aplicaciones vamos a comentar algunos conceptos sobre los distintos tipos de servidores.

Como ya hemos comentado, nuestra principal unidad de trabajo es el dominio. El dominio no es más que una agrupación de todos los componentes que utilizamos para nuestro trabajo (servidores, máquinas, aplicaciones, etc.). Un ejemplo de uso de dominios es el siguiente. Cuando se desarrolla una aplicación se suele separar la fase de desarrollo de una aplicación con la fase de producción (cuando la aplicación ya está funcionando hacia el usuario y dando servicio). Para manejar esta situación podemos tener creados dos dominios, uno para desarrollo y otro para producción. A pesar de contener exactamente los mismos componentes funcionan de forma independiente.

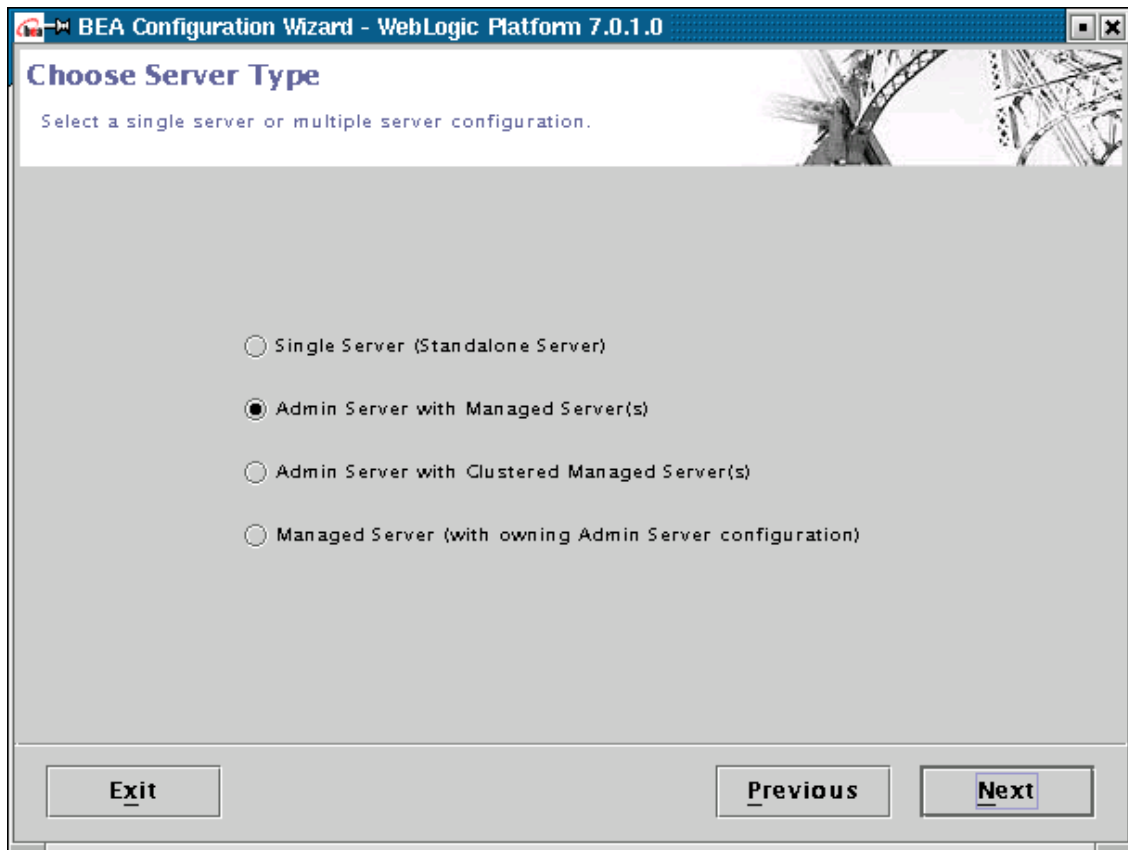
Dentro de un dominio vamos a tener máquinas y servidores. Al menos debemos tener un servidor en nuestro dominio, que llamaremos de *administración*. El servidor de administración es único en el dominio y va a realizar, como su nombre indica, tareas administrativas. Podemos tener más servidores, que llamaremos *administrados* (*managed*). De este tipo de servidor podemos tener tantos como queramos.

Vamos a empezar a crear nuestro primer dominio. Vamos a llamarlo *MiDominio* y contendrá dos servidores alojados en la misma máquina: *Servidor1* y *Servidor2*. El servidor 1 será el de administración. Utilizaremos un asistente que incorpora Weblogic para crear el dominio y los servidores. Nos situamos en `$HOME_BEA/weblogic700/common/bin` (`$HOME_BEA` es el directorio donde hemos instalado Weblogic, en mi máquina `/home/miguel/bea`) y ejecutamos `.dmwiz.sh`

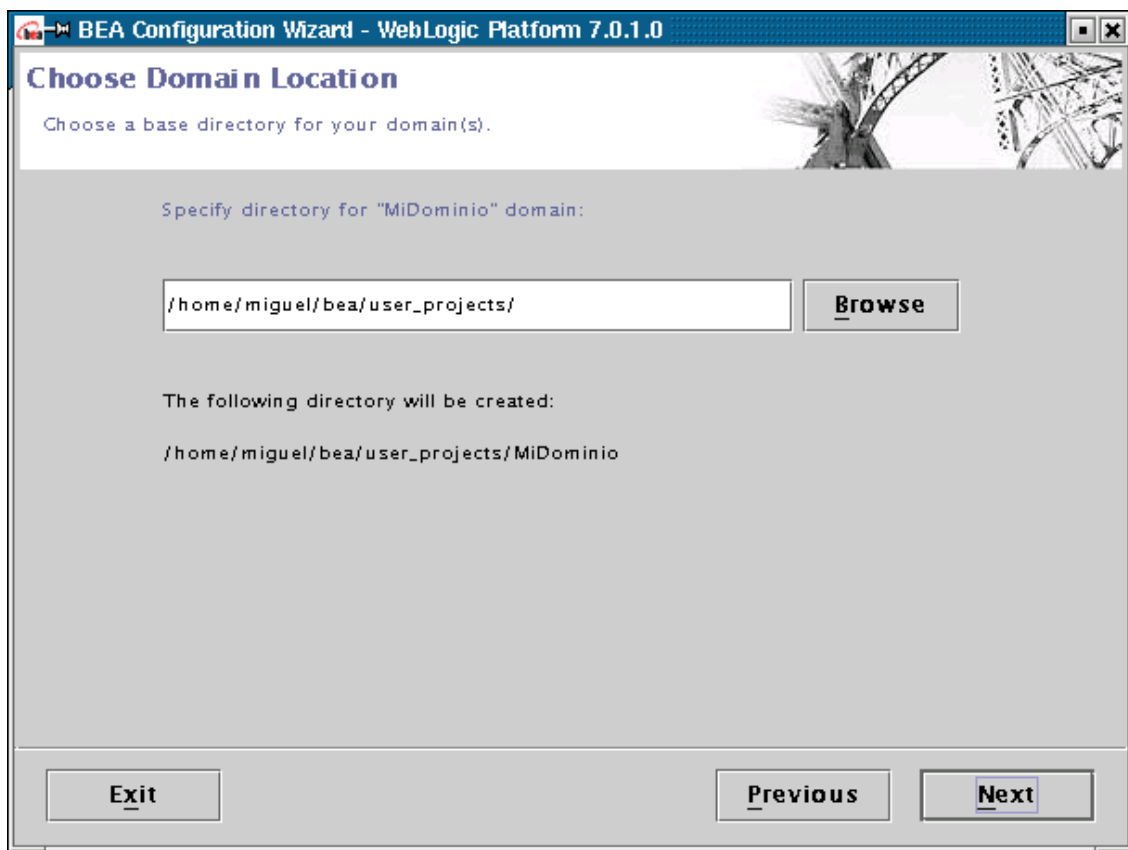
Nos aparecerá la ventana que se muestra en la siguiente figura, en la que seleccionaremos *WLS domain* y le damos el nombre que queramos, en nuestro caso *MiDominio*. Pulsamos en el botón *Next*.



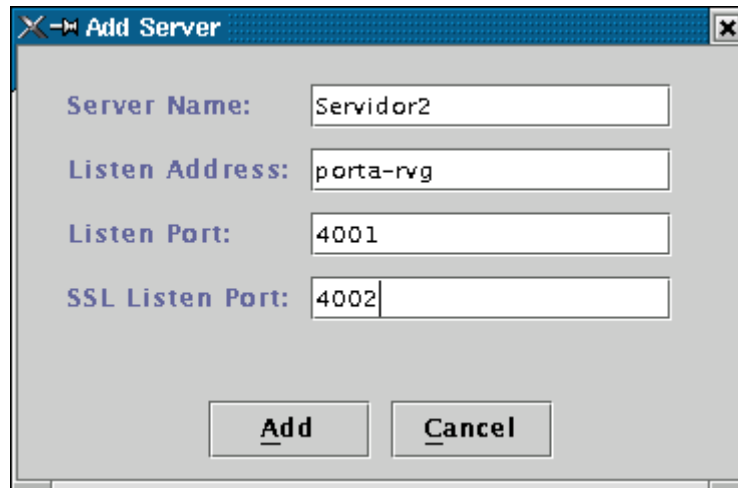
Ahora debemos seleccionar si queremos instalar sólo un servidor, un servidor de administración con servidores administrados, lo mismo en cluster o un servidor administrado. Vamos a seleccionar la opción mostrada: un servidor de administración con servidores administrados.



Pulsamos *Next* y nos indica que introduzcamos el directorio de instalación (dejamos el que viene por defecto). Pulsamos *Next*.



En la siguiente pantalla nos pide que creamos los servidores administrados. Pinchamos en *Add* y nos aparecerá la siguiente figura en la que introducimos los datos del servidor. Fijaros que vamos a utilizar el puerto 4001 y el 4002. Debéis cambiar la dirección de escucha para poner la dirección de vuestra máquina.

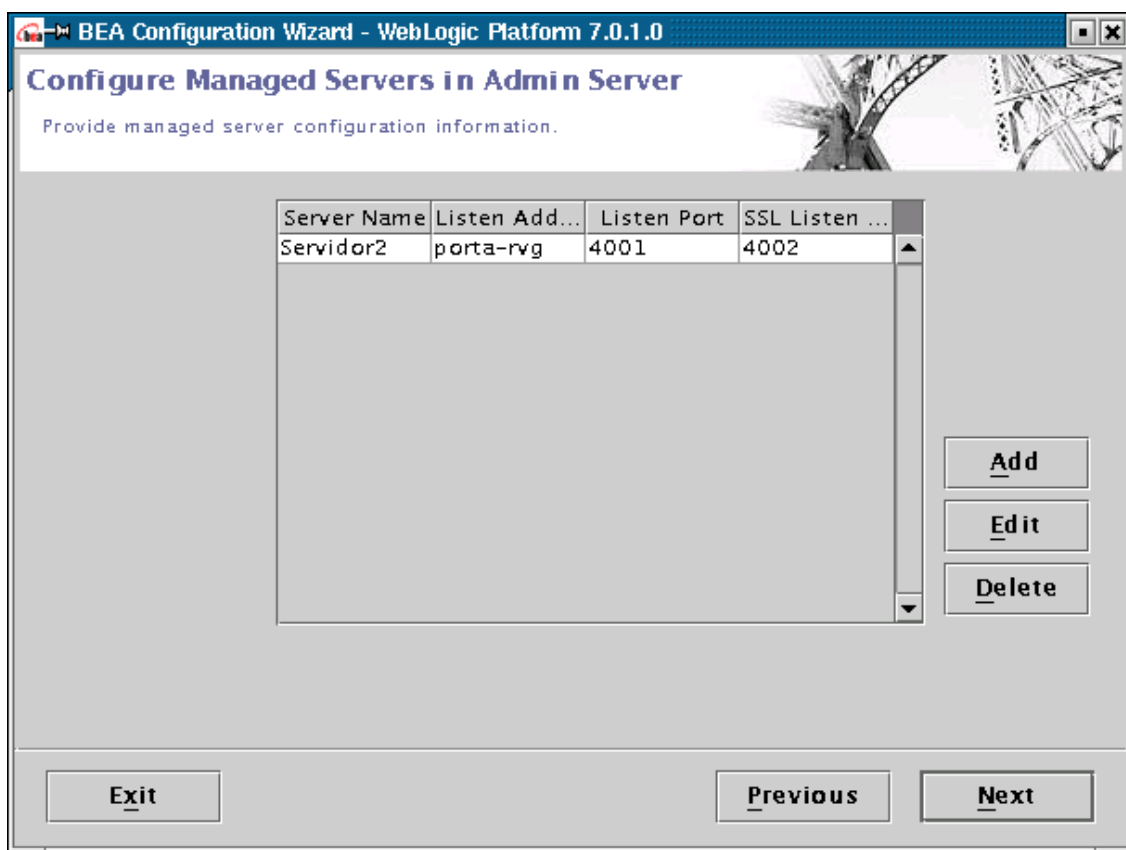


The 'Add Server' dialog box contains the following fields and buttons:

Field Label	Value
Server Name:	Servidor2
Listen Address:	porta-rvg
Listen Port:	4001
SSL Listen Port:	4002

Buttons: **Add**, **Cancel**

Podemos añadir tantos servidores como queramos.

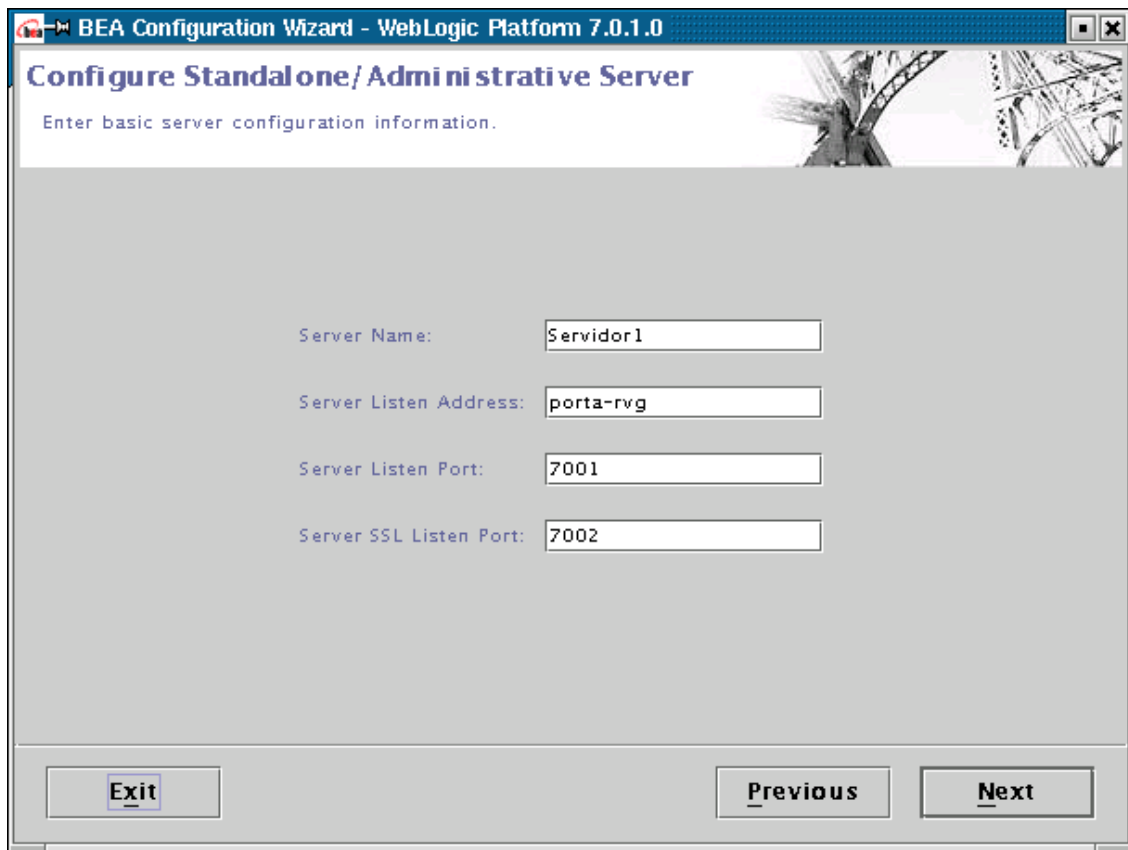


The 'BEA Configuration Wizard - WebLogic Platform 7.0.1.0' window shows the 'Configure Managed Servers in Admin Server' step. It includes a table with the following data:

Server Name	Listen Add...	Listen Port	SSL Listen ...
Servidor2	porta-rvg	4001	4002

Buttons on the right: **Add**, **Edit**, **Delete**. Buttons at the bottom: **Exit**, **Previous**, **Next**.

Al pinchar en *Next* nos aparece la configuración del servidor de administración. También aquí debéis poner la dirección de vuestra máquina. Fijaros que hemos puesto un puerto distinto al anterior, porque estamos en la misma máquina.



BEA Configuration Wizard - WebLogic Platform 7.0.1.0

Configure Standalone/Administrative Server

Enter basic server configuration information.

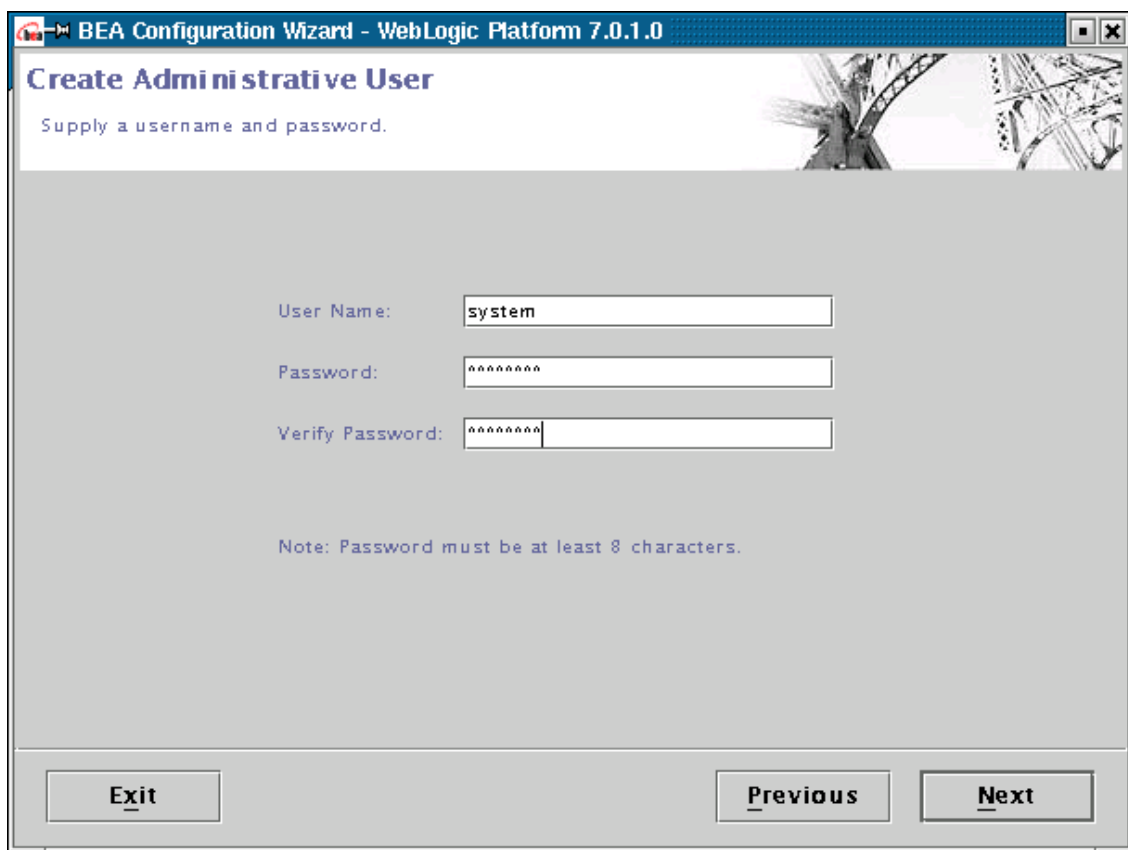
Server Name:

Server Listen Address:

Server Listen Port:

Server SSL Listen Port:

Por último, debemos crear un usuario de administración y asignarle una contraseña. Vamos a utilizar *system* y *weblogic* como contraseña.



BEA Configuration Wizard - WebLogic Platform 7.0.1.0

Create Administrative User

Supply a username and password.

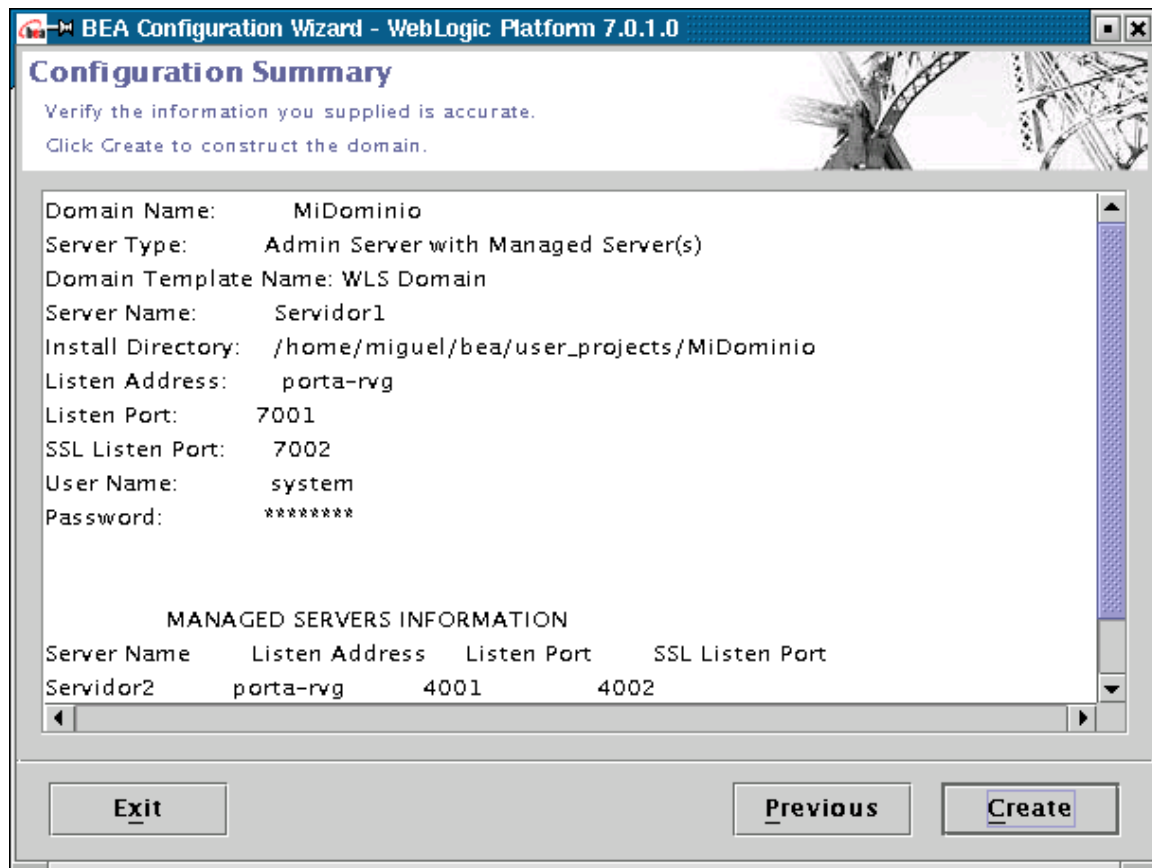
User Name:

Password:

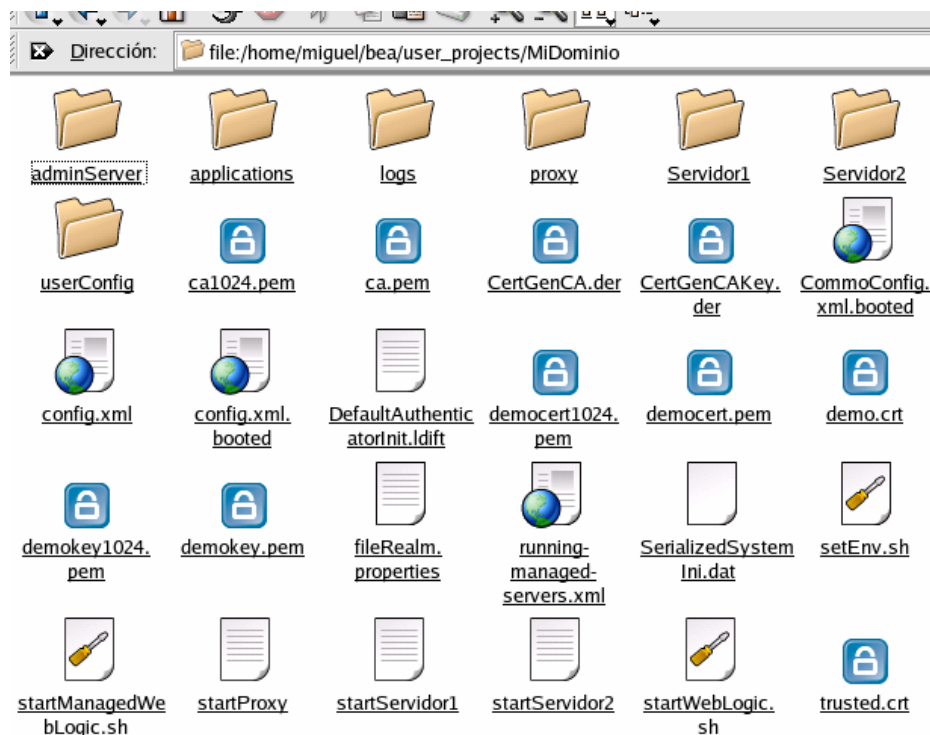
Verify Password:

Note: Password must be at least 8 characters.

El último paso es confirmar los datos introducidos y pinchar en *Create* para crear el dominio.



Una vez creado el dominio la estructura de directorios creada es la siguiente:



Tenemos un directorio por cada servidor creado, donde se guardan datos específicos del servidor (por ejemplo el fichero log). En esta figura tenemos cuatro servidores: *adminServer*, *proxy*, *Servidor1* y *Servidor2*. El fichero *config.xml* contiene los datos del dominio (nombre de los servidores, máquinas, dominio, etc., nombre de las aplicaciones y su configuración, etc.). Los ficheros *.pem* sirven para la seguridad. Los ficheros *startWebLogic.sh* y *startManagedWebLogic.sh* sirven para arrancar al servidor de administración y a los administrados.

2.3. Arranque y configuración

Para poner en marcha los servidores debemos utilizar unos ejecutables que se encuentran en `$HOME_BEA/user_projects/MiDominio`. Primero debemos arrancar el servidor de administración. Para ello ejecutamos desde línea de comandos: *./startWebLogic.sh* Nos pedirá el usuario y la contraseña que hemos definido antes. Cuando nos aparezca el siguiente mensaje ya está arrancado el servidor:

<Server started in RUNNING mode>

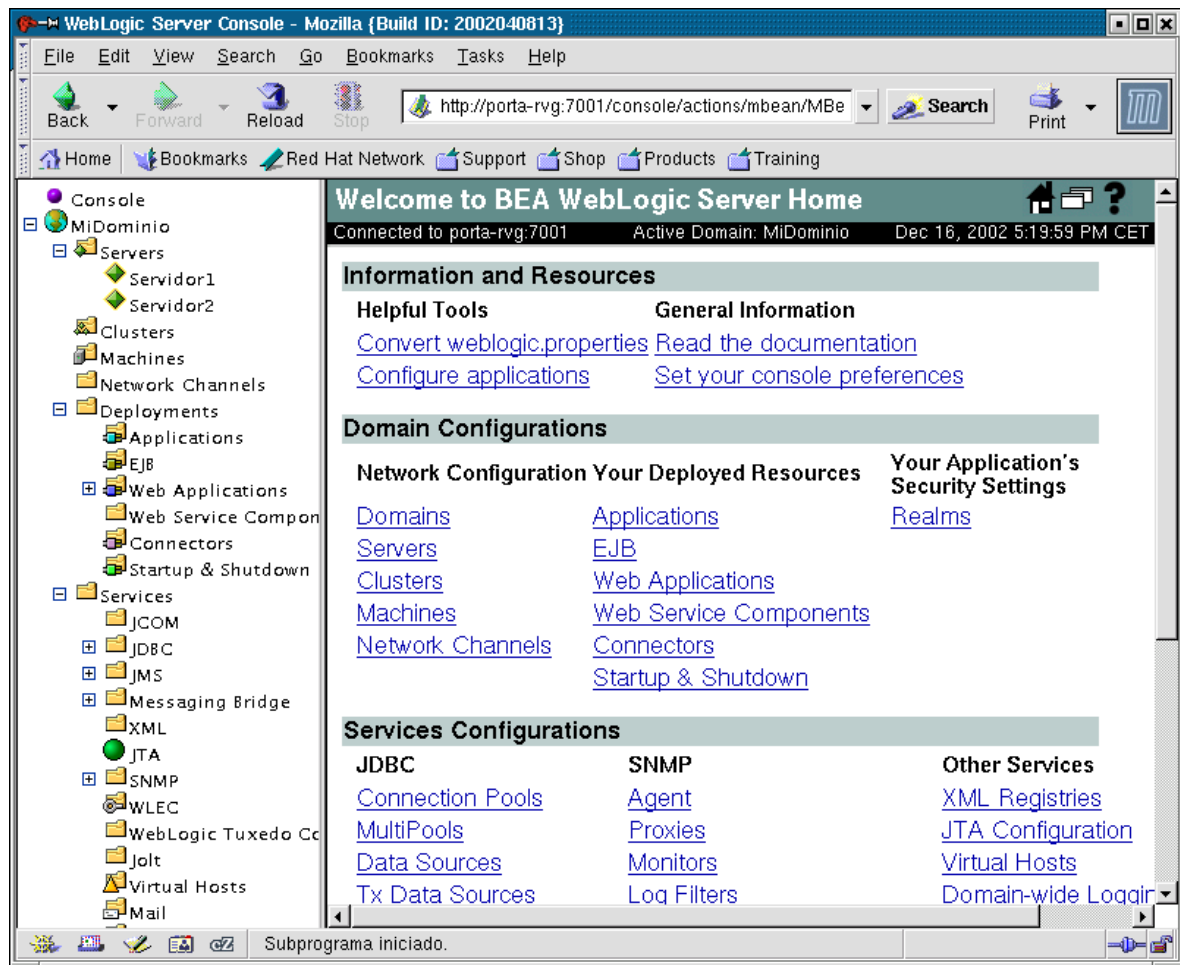
En este momento podemos arrancar los servidores administrados. Para ponerlos en marcha debemos utilizar el siguiente comando con los parámetros indicados:

./startManagedServer.sh nombre_servidor dirección_servidor_administración

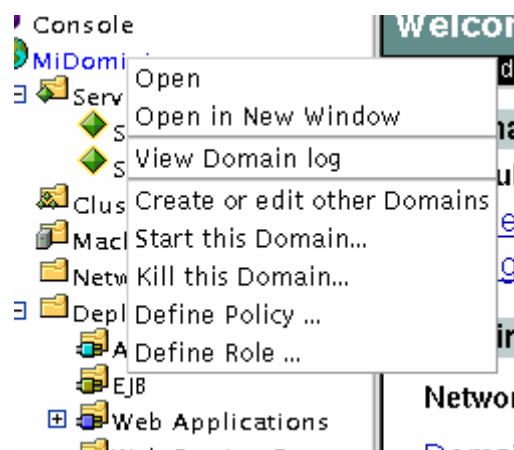
Por ejemplo, para arrancar el servidor administrado creado anteriormente debemos ejecutar el siguiente comando:

./startManagedServer.sh servidor2 http://localhost:7001

El servidor de administración nos facilita una aplicación que permite administrar nuestro dominio. Con la consola podemos configurar los atributos de los distintos recursos, hacer despliegues de aplicaciones, monitorizar el uso de recursos, ver mensajes de log y poner en marcha o parar los distintos servidores de nuestro dominio. La consola se gestiona con un navegador en la siguiente dirección: *http://localhost:7001/console*. Nos aparecerá una página donde se nos solicita el usuario y la contraseña. Una vez introducida nos aparecerá la siguiente página.



La parte de la izquierda es una applet en forma de árbol jerárquico que nos muestra todas las opciones que podemos configurar en el dominio. En la parte superior del árbol tenemos el nombre del dominio. Dentro del dominio, las primeras opciones nos permiten configurar los elementos del dominio (servidores, máquinas, cluster, etc.). A continuación podemos realizar despliegues de aplicaciones, aplicaciones web, EJBs, etc. La última opción contiene los servicios configurables (JDBC, JMS, Virtual Hosts, etc.). Los elementos de la parte izquierda disponen de un menú adicional que se obtiene pinchando con el botón derecho sobre un elemento del árbol, como el mostrado en la siguiente figura:



Las opciones varían dependiendo del elemento seleccionado.

En la parte derecha tenemos las mismas opciones a las que podemos acceder mediante el applet. También nos irán apareciendo las detalles de configuración para un servicio o característica concreta elegida en la parte izquierda.

Vamos a ver las opciones más generales. Si pinchamos en el elemento *console* nos permite configurar opciones generales a todos los dominios. Nos aparecerá una página como la mostrada a continuación donde podemos:

- Seleccionar el idioma (inglés o japonés)
- Indicar el tiempo de refresco de las páginas.
- Indicar el tiempo de refresco de los datos gráficos (vistos un poco más adelante).
- Si marcamos la opción *Remember Last Tab*, cuando pasemos de una opción a otra se acordará de la última solapa visitada en dicha opción.
- Al marcar la última opción nos permite disponer del árbol de navegación.

Console

Preferences Versions

Language: English

Auto-refresh every: 10 seconds

Poll for graph data every: 500 ms

☒ Remember Last Tab

☒ Use Navigation Tree

Apply



En la otra solapa simplemente nos da información de versión.

Console

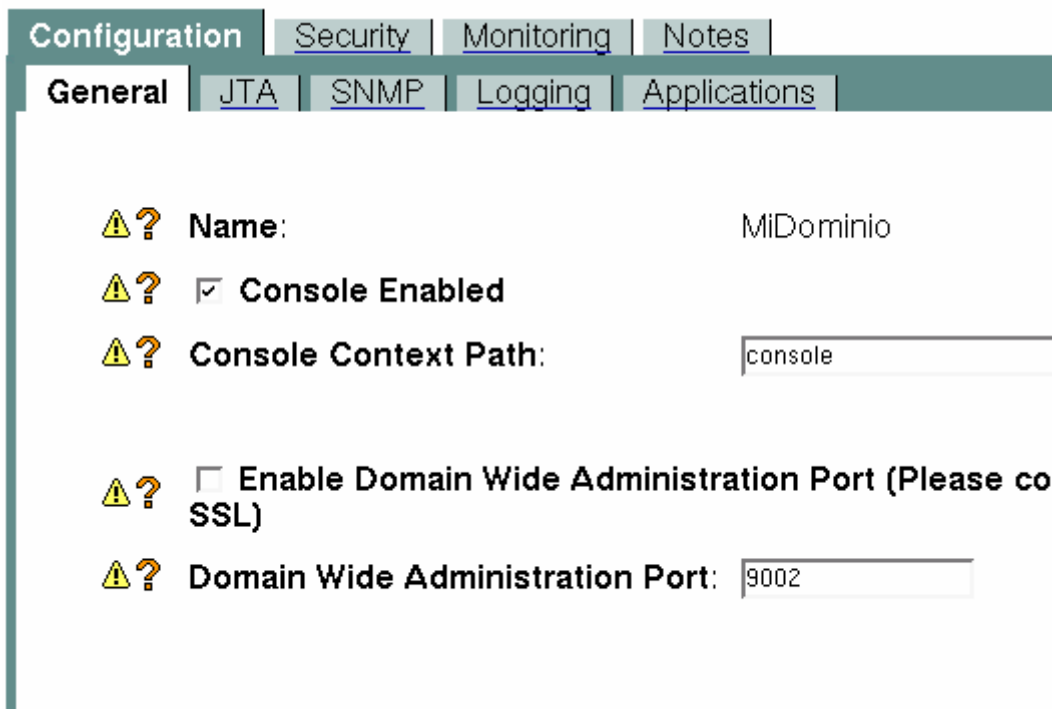
Preferences Versions











Console Release Build
7.0.1.0

Server Release Build
7.0.1.0

Pasamos a las opciones para el dominio. Pinchamos en el nombre de nuestro dominio y nos aparece una ventana como la siguiente. Los símbolos que aparecen a la izquierda de las opciones significan lo siguiente:  nos da información sobre la opción y  nos indica que es necesario reiniciar uno o varios servidores si cambiamos esa opción. En la configuración general (la solapa actual) podemos configurar las siguientes opciones:

- Habilitar la consola. En modo producción suele ser habitual deshabilitar la consola, para que no pueda ser accedida desde el exterior.
- La siguiente opción nos permite dar un nombre distinto a la aplicación de la consola. Si, por ejemplo, damos el nombre *miconsola*, para acceder a la consola tendríamos que teclear `http://dirección:puerto/miconsola`.
- La habilitación del puerto de administración, si marcada, permite que todos los elementos del dominio se comuniquen con el servidor de administración mediante una conexión segura. Además podemos configurar un puerto adicional (no puede ser el seguro del servidor de administración) para dichas comunicaciones. Esta opción permite que podamos arrancar un servidor en modo *standby* en el cual el servidor no escucha las peticiones que le llegan a su puerto, pero se permite una comunicación con el servidor de administración. También permite separar las peticiones de aplicación (llegan de las aplicaciones que usan el sistema) de las peticiones de administración (generadas por o hacia el servidor de administración). De esta manera una petición del servidor de administración puede ser atendida sin tener que esperar su turno dentro de las peticiones de aplicación.




Configuration	
General	JTA SNMP Logging Applications
 	Name: MiDominio
 	<input checked="" type="checkbox"/> Console Enabled
 	Console Context Path: console
 	<input type="checkbox"/> Enable Domain Wide Administration Port (Please configure SSL)
 	Domain Wide Administration Port: 9002


En la solapa de *Logging* (las opciones JTA y SNMP las veremos más adelante) podemos configurar el fichero log del dominio. El fichero log almacena toda la información y mensajes del dominio. Las opciones son las siguientes:


- Podemos cambiar el nombre del fichero log.
- La siguiente opción permite especificar el tipo de rotación. Las opciones a elegir son por tamaño o por tiempo. La rotación permite que el fichero log no vaya creciendo indefinidamente. Si elegimos por tamaño, se cogerá el valor del parámetro *File Min Size* y, cuando el fichero de log alcance ese tamaño, creará un nuevo fichero de log, renombrando el anterior. Si, por ejemplo, el nombre del fichero de log es *wl-domain.log* y hemos seleccionado una rotación por tamaño y 500k de tamaño mínimo, cuando el fichero alcance ese tamaño el sistema cambiará el nombre del fichero por *wl-domain.0* y creará uno nuevo, *wl-domain.log*, donde se seguirá almacenando la salida del sistema. Cuando se vuelva a superar ese límite se le dará el nombre *wl-domain.1* y así sucesivamente. El otro tipo de rotación, de tiempo, actúa de manera similar, pero especificando un tiempo de rotación. Cuando el reloj del sistema llega a esa hora se produce el cambio de fichero. En esta opción, podemos especificar cada cuantos días se produce el cambio, cambiando el valor de *File Time Span*.
- La penúltima opción permite limitar el número de ficheros a almacenar. Si la activamos toma el valor de la siguiente opción *File Count* y, cuando el contador de fichero alcance ese valor, empieza desde cero sobrescribiendo el primero.
- Si pinchamos en la opción *View Domain Log* se nos permite ver el fichero log (ver siguientes figuras).


Connected to port 7001 Native Domain: wl-domain-000-10, Log...


Configuration	Security	Monitoring	Notes	
General	JTA	SNMP	Logging	Applications


 **File Name:**


 **Rotation Type:**

 **File Min Size:** **k**

 **Rotation Time:**

 **File Time Span:** **hours**

 ☐ **Number Of Files Limited**

 **File Count:**

[View domain log](#)

Visualización del fichero de log.

Connected to pona-rvg:7001 Active Domain: mldominio Dec 18, 2002 7:13:10 P

[Customize this view...](#)

Timestamp	Server	Sev
30-nov-02 14:20:47 CET	Servidor1	Not
30-nov-02 14:20:47 CET	Servidor1	Not
30-nov-02 14:21:32 CET	Servidor1	Not
30-nov-02 14:21:47 CET	Servidor1	Not
30-nov-02 14:21:48 CET	Servidor1	Not
30-nov-02 14:21:48 CET	Servidor1	Not

Si pinchamos en *Customize this view* se nos permite configurar los mensajes, mostrando las opciones que queremos que se vean en el fichero de log.

Columns:

Available		Chosen
messageid	→	timestamp
machine		server
user		severity
thread	←	subsystem

Servers:

Available		Chosen
Servidor1	→	
Servidor2	←	

Subsystems:

Available		Chosen
Application Poller	→	
Cluster		
COM		
common	←	

Available	Chosen
-----------	--------

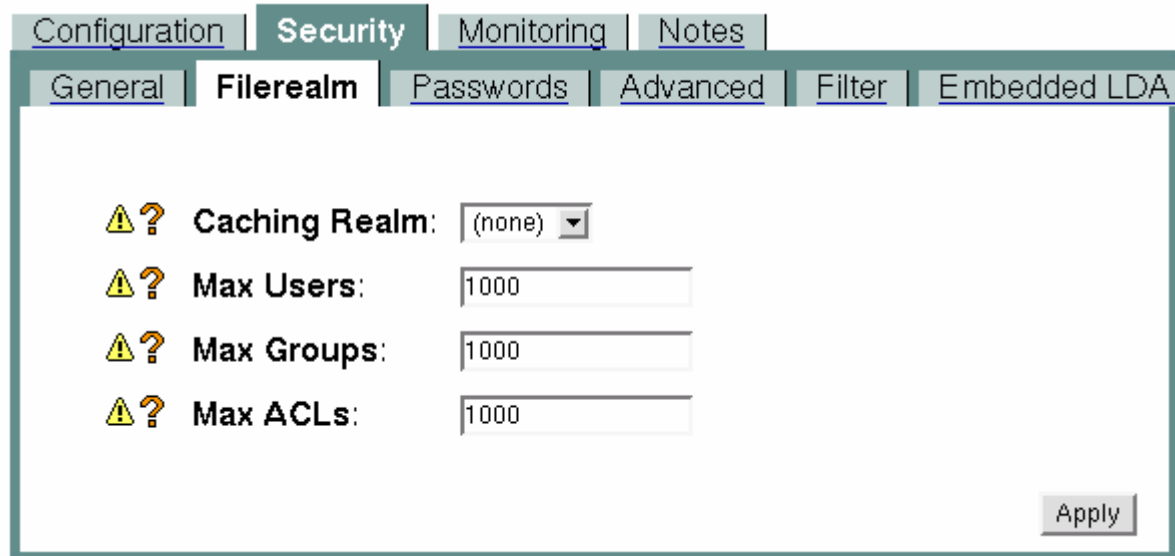
En la siguiente figura, si activamos la primera opción, *Auto Deployed Enabled*, el sistema se encarga de comprobar si existen aplicaciones nuevas. Estas aplicaciones se pueden crear como subdirectorios del directorio de aplicaciones. La siguiente opción indica cada cuantos milisegundos se comprueba si existe una aplicación nueva. En modo producción se deshabilita esta comprobación.

The screenshot shows the 'Applications' configuration tab. At the top, there are tabs for 'Configuration', 'Security', 'Monitoring', and 'Notes'. Below these, there are sub-tabs: 'General', 'JTA', 'SNMP', 'Logging', and 'Applications'. The 'Applications' sub-tab is selected. The main content area contains two settings: a checkbox labeled 'Auto Deployed Enabled' which is checked, and a text input field labeled 'Auto Update Interval' with the value '3000'. A 'Refresh' button is located at the bottom right of the configuration area.

Pasamos a la solapa de seguridad. Todo lo referente a *realm* lo explicaremos más adelante. Si activamos la opción *Guest Disabled* no permitiremos que entre el usuario invitado.

The screenshot shows the 'Security' configuration tab. At the top, there are tabs for 'Configuration', 'Security', 'Monitoring', and 'Notes'. Below these, there are sub-tabs: 'General', 'Filerealm', 'Passwords', 'Advanced', 'Filter', and 'E'. The 'General' sub-tab is selected. The main content area contains three settings: a dropdown menu labeled 'Default Realm' with 'myrealm' selected, a text input field labeled 'Audit Provider Class' which is empty, and a checkbox labeled 'Guest Disabled' which is unchecked. A 'Refresh' button is located below the 'Guest Disabled' checkbox.

En el apartado de *FileRealm* podemos configurar opciones específicas de seguridad como: número máximo de usuarios, grupos y ACL.



The screenshot shows the WebLogic configuration console with the 'Security' tab selected. Under the 'Security' tab, the 'Filerealm' sub-tab is active. The configuration area contains four settings, each preceded by a yellow warning icon with a question mark:

- Caching Realm:** A dropdown menu currently set to '(none)'.
- Max Users:** A text input field containing the value '1000'.
- Max Groups:** A text input field containing the value '1000'.
- Max ACLs:** A text input field containing the value '1000'.

An 'Apply' button is located in the bottom right corner of the configuration area.

La siguiente solapa tiene que ver con características de la contraseña de acceso y el bloqueo de una cuenta por haber intentado acceder con una contraseña incorrecta. Si un usuario intenta acceder al sistema e introduce una contraseña incorrecta, cuando realice un determinado número de intentos la cuenta será deshabilitada. Las opciones son:



- Longitud mínima de contraseña. Indica el número de caracteres mínimo que debe tener la contraseña.
- La siguiente opción, si marcada, permite el bloqueo de una cuenta al intentar acceder con una contraseña errónea.
- La opción *Lockout Threshold* especifica el número de intentos erróneos que provocan el bloqueo de la cuenta.
- La siguiente es el número de minutos que se bloquea la cuenta.
- La opción *Lockout Reset Duration* indica el número de minutos durante los cuales se cuenta el número de intentos fallidos. Si marcamos cinco, si durante cinco minutos se han realizado cinco (el número indicado por *Lockout Threshold*) intentos fallidos, se produce el bloqueo.
- La última opción es el tamaño de la cache de intentos fallidos de cualquier usuario que el sistema almacenará.

El resto de opciones de seguridad se detallarán en el apartado de seguridad.

The screenshot shows the 'Security' tab with the 'Passwords' sub-tab selected. It contains several configuration fields, each preceded by a yellow warning icon with a question mark:





- Minimum Password Length:** 8
- Lockout Enabled:** ☒
- Lockout Threshold:** 5
- Lockout Duration:** 30
- Lockout Reset Duration:** 5
- Lockout Cache Size:** 5

An 'Apply' button is located at the bottom right of the configuration area.

Cuando seleccionamos la solapa *Monitoring* nos aparece un enlace que nos permite monitorizar los cluster y servidores de nuestro dominio. Si pinchamos en el enlace nos aparece la siguiente figura. Tenemos los servidores creados en nuestro dominio y se nos indica la dirección de escucha, los puertos de escucha y el estado del servidor. También tenemos dos enlaces, uno para parar o poner en marcha el servidor y otro para monitorizarlo (veremos más adelante estas opciones). Los iconos de la parte derecha nos permiten:  crear un nuevo servidor con las mismas características que éste y  eliminar este servidor.

Connected to porta-rvg:7001 Active Domain: MiDominio Dec 16, 2002 5:52:20 PM CE

 [Customize this view...](#)


Name	Machine	Listen Address	Listen Port	State	SSL Listen Port	Server Controls	
Servidor1		porta-rvg	7001	RUNNING	7002	[Start/Stop] [Monitor]	 
Servidor2		porta-rvg	4001	UNKNOWN	4002	[Start/Stop] [Monitor]	 

La última solapa, *Notes*, nos permite introducir notas asociadas a la configuración actual. Esto es común en la mayoría de opciones de configuración. Tienen un carácter informativo.

MiDominio

Connected to porta-rvg:7001Active Domain: MiDominioDec 1

[Configuration](#) | [Security](#) | [Monitoring](#) | **Notes**

 **Notes:**

Este es mi dominio de prueba

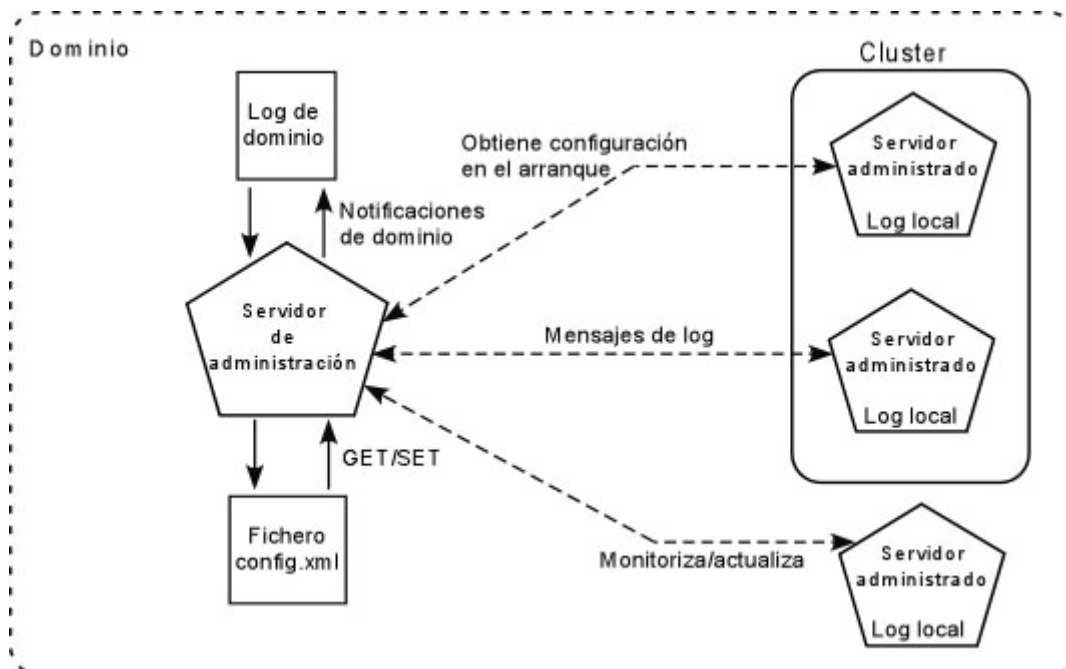
Tema 3: Administración del servidor de aplicaciones

En este tema aprenderemos a administrar un dominio. Veremos cómo podemos definir un nuevo dominio y todos los elementos asociados (máquinas, servidores, cluster, usuarios, etc.). También veremos herramientas adicionales como el NodeManager y las herramientas de administración desde línea de comandos.

3.0. Revisión de conceptos

Vamos a revisar algunos de los conceptos ya vistos con anterioridad. Dentro de un dominio debemos disponer de un servidor de administración. Este servidor se encargará, entre otras, de las siguientes tareas:

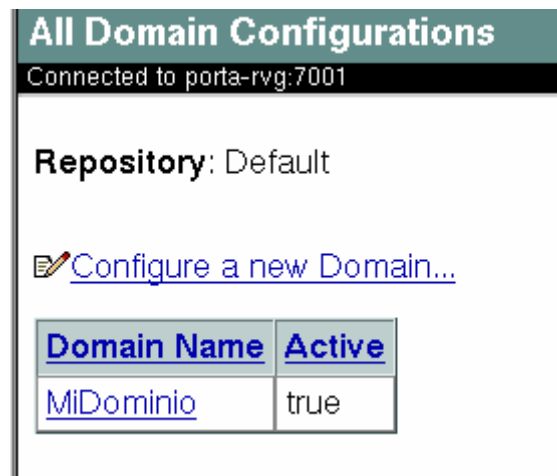
- Mantenimiento del fichero de log. Todos los mensajes de los distintos servidores así como los propios del dominio se almacenarán en el fichero log del dominio.
- Leer y escribir en el fichero *config.xml*. Este fichero contiene toda la configuración del dominio (nombre de los servidores, puertos, aplicaciones disponibles, etc.).
- Proporciona información de configuración cuando arrancan los servidores administrados. Por ello, el servidor de administración debe estar en marcha cuando arranquen el resto de servidores. Una vez arrancados todos podemos parar el de administración.
- Permite monitorizar el comportamiento de los servidores y cambiar su configuración.



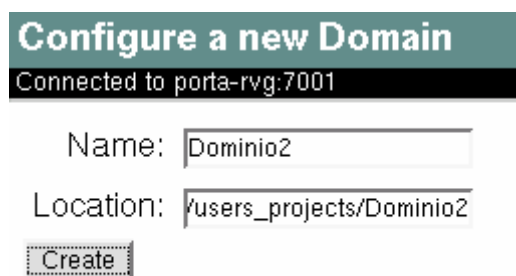
3.1. Creación y configuración de dominio

3.1.1. Definición de un nuevo dominio

En el apartado 2.2 utilizamos una utilidad que incorpora Weblogic para crear y configurar de manera sencilla un dominio. Ahora vamos a ver otra forma de configurar un dominio. Para crear un nuevo dominio pinchamos con el botón derecho encima del nombre del dominio y seleccionamos *Create or edit other domains...* Nos aparecerá una figura como la siguiente. Podemos pinchar en *Configure a new domain...* para crear un nuevo dominio.



Damos el nombre del dominio y la localización. Podemos utilizar el directorio por defecto: `$BEA_HOME/users_projects/nombre` (antes de crear el dominio debemos asegurarnos de crear el directorio). Una vez creada nos aparecerá la ventana de configuración de dominio que vimos en la sección 2.3. A pesar de dar esta opción, se aconseja crear el dominio con la herramienta utilizada en la sesión anterior, de otra forma sería necesaria una configuración manual del fichero `config.xml`.



Una vez creado el dominio junto con sus servidores, cuando arrancamos un servidor debemos suministrar cierta información. Vamos a configurar los ejecutables para que contengan esa información. Lo primero es el usuario y la contraseña con la que arrancamos los servidores. Editamos los ficheros:

```
startWebLogic.sh
startManagedWebLogic.sh
```

y les damos valor a las variables:

```
WLS_USER=system
WLS_PW=weblogic
```

donde *system* y *weblogic* son el usuario y la contraseña del sistema, respectivamente. El servidor de administración se arranca simplemente ejecutando *startWebLogic.sh*. Para los servidores administrados debemos indicar la dirección del servidor de administración. Para evitar tener que teclear esta información podemos crear un ejecutable que la contenga. Creamos un nuevo fichero que contendrá únicamente este comando:

```
./startManagedServer.sh servidor2 http://localhost:7001
```

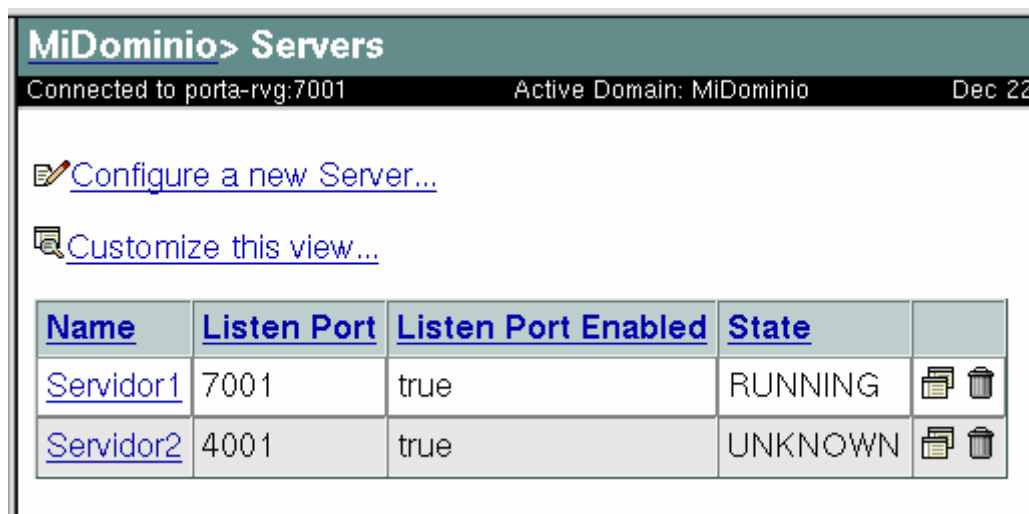
donde *servidor2* es el nombre del servidor que queremos arrancar y a continuación indicamos la dirección y el puerto de escucha del servidor de administración. Para que el fichero sea ejecutable, desde línea de comandos, ejecutamos:





```
chmod 777 startServidor2
```

donde *startServidor2* es el nombre del fichero. Podemos crear un fichero por cada servidor.

3.1.2. Definición de servidores dentro del dominio

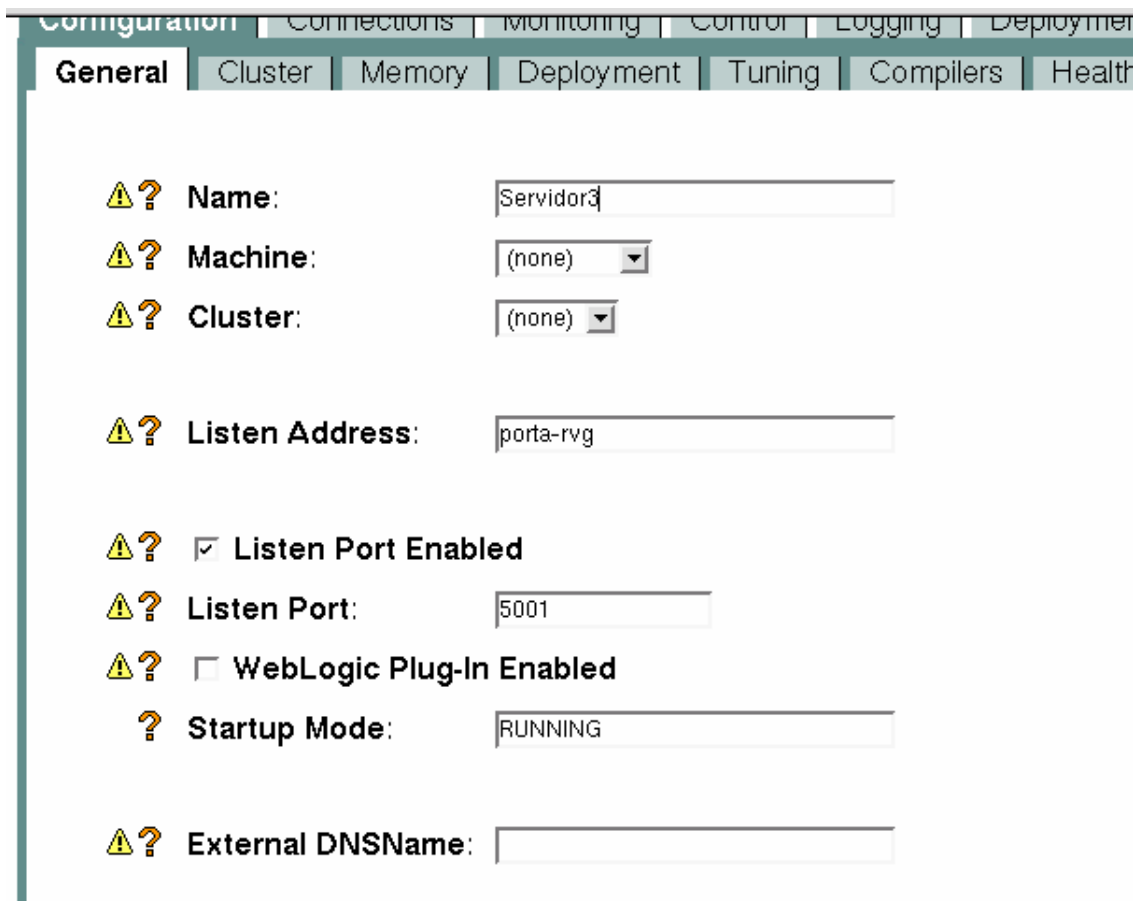
Un servidor de aplicaciones no es más que una instancia de la clase *weblogic.Server*. Vamos a ver cómo podemos crear nuevos servidores. Lo primero es pinchar en el icono *Servers*. Nos aparecerá una página como la siguiente donde tenemos los servidores definidos en nuestro dominio y podemos definir nuevos.



Name	Listen Port	Listen Port Enabled	State	
Servidor1	7001	true	RUNNING	 
Servidor2	4001	true	UNKNOWN	 

Cuando pinchamos en *Configure a new server..* nos aparece la siguiente página donde podemos configurar la siguiente información:

- **Name:** el nombre del servidor. Debe ser único en el dominio.
- **Machine:** si ya tenemos definidas máquinas podemos asignar este servidor a una de ellas. Esta operación la podemos hacer más adelante.
- **Cluster:** ídem al anterior pero con los clusters.
- **Listen Address:** la dirección (IP o DNS) de escucha del servidor.
- **Listen Port Enabled:** al activar esta opción permitimos la escucha por un puerto.
- **Listen Port:** el puerto de escucha del servidor. Dos servidores en la misma máquina no pueden tener el mismo puerto de escucha.
- **WebLogic Plug-in Enabled:** habilita el plugin de WebLogic (visto más adelante).
- **Startup mode:** el modo de arranque. Por defecto está en RUNNING, pero lo podemos poner en STANDBY. Este último modo hace que el servidor no responda a las peticiones.
- **External DNS Name:** esta opción sirve para cuando tenemos un *firewall* en nuestro sitio y la dirección interna de la máquina que aloja el servidor es distinta de la dirección externa.



The screenshot shows the 'Configuration' window for a new server. The 'General' tab is selected, and the 'Name' field is set to 'Servidor3'. The 'Machine' and 'Cluster' fields are set to '(none)'. The 'Listen Address' field is set to 'porta-rvg'. The 'Listen Port Enabled' checkbox is checked, and the 'Listen Port' field is set to '5001'. The 'WebLogic Plug-In Enabled' checkbox is unchecked. The 'Startup Mode' field is set to 'RUNNING'. The 'External DNSName' field is empty.

Configuration	Connections	Monitoring	Control	Logging	Deployment	
General	Cluster	Memory	Deployment	Tuning	Compilers	Health

Name:

Machine:

Cluster:

Listen Address:

Listen Port Enabled: ☒

Listen Port:

WebLogic Plug-In Enabled: ☐

Startup Mode:

External DNSName:

Pasamos a la solapa *Connections* (el resto de opciones se verán más adelante). Las solapas *SSL* y *SSL Ports* permiten configurar el comportamiento seguro de este servidor.

Connected to rvg7.i3a.ua.es:7001 Active Domain: J2EE Dec 22, 2002

Configuration **Connections** Monitoring Control Logging Deployments

SSL SSL Ports HTTP jCOM Tuning Protocols Summary

[Configure Key Store Providers](#)

⚠️ **Server Private Key Alias:** demokey

⚠️ **Server Private Key Passphrase:** [Change...](#)

⚠️ **Server Certificate File Name:** democert.pem

⚠️ ☐ **Client Certificate Enforced**

⚠️ ☐ **Client Certificate Requested But Not Enforced**

⚠️ **Export Key Lifespan:** 500

⚠️ ☒ **Hostname Verification Ignored**

⚠️ **Hostname Verifier:**

Si configuramos SSL y seleccionamos la opción *Enable SSL Listen Port* ya podemos comunicarnos con el servidor mediante el protocolo SSL y por el puerto indicado en la opción *SSL Listen Port*.

MiDominio> Servers> Servidor3

Connected to porta-rvg:7001 Active Domain: MiDominio Dec 22, 2002 9:01:10 PM

Configuration **Connections** Monitoring Control Logging Deployments

SSL **SSL Ports** HTTP jCOM Tuning Protocols Summary

Listen Address: porta-rvg

⚠️ ☐ **Enable SSL Listen Port (Please configure SSL)**

⚠️ **SSL Listen Port:** 7002

Enable Domain Wide Administration Port: false

Domain Wide Administration Port: 9002

⚠️ **Local Administration Port Override (0: no override):** 0

[Channel Overrides](#)

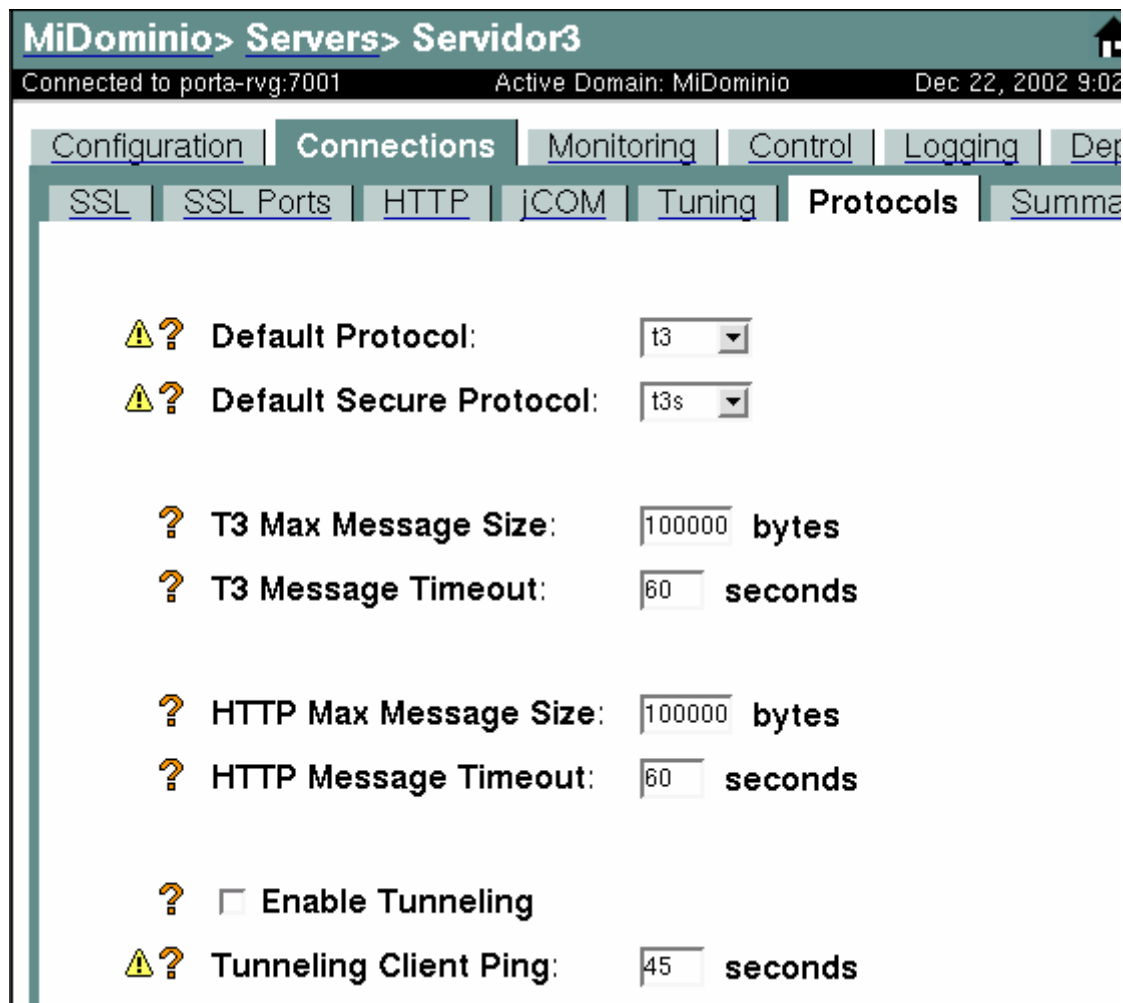
En la solapa *Connections->HTTP* podemos configurar la aplicación con la que responderá el servidor cuando se produzca una petición sin especificar ningún nombre. *Default Web Application*. También podemos configurar nuestro servidor para defenderlo de ataques DoS. Un ataque DoS (*Denial of Service*: denegación de servicio) tiene como consecuencia la caída del servidor. Lo suelen utilizar *hackers* para echar abajo un sistema. El ataque puede ser llevado a cabo de dos maneras:

- Por el envío de un paquete de datos demasiado grande que sobrepasa la memoria del sistema. La protección que podemos realizar es indicar en la opción **Max Post Size** el tamaño máximo de paquete a recibir. Por defecto está en -1 que indica que no existe limitación de tamaño.
- Por el envío de sucesivos paquetes incompletos. Cuando enviamos un paquete incompleto, el servidor queda esperando el resto del paquete. El envío de muchos de estos paquetes puede colapsar el servidor. Para combatir este tipo de ataque debemos configurar dos opciones: **Post Timeout Secs** es el tiempo de espera máximo para recibir el siguiente paquete; y **Max Post Time** es el tiempo de espera para mensajes incompletos. Los dos se expresan en segundos.

The screenshot shows the 'MiDominio> Servers> Servidor3' configuration window. The 'Connections' tab is active, and the 'HTTP' sub-tab is selected. The interface includes a top bar with 'Connected to porta-rvg:7001', 'Active Domain: MiDominio', and the date 'Dec 22, 2002'. Below the tabs, the 'HTTP' configuration section contains several settings:

- Default Web Application:** A dropdown menu currently set to '(none)'.
- Frontend Host:** An empty text input field.
- Frontend HTTP Port:** A text input field containing '0'.
- Frontend HTTPS Port:** A text input field containing '0'.
- Send Server Header Enabled:** A checkbox that is checked.
- Post Timeout Secs:** A text input field containing '30'.
- Max Post Time:** A text input field containing '-1' followed by the label 'seconds'.
- Max Post Size:** A text input field containing '-1' followed by the label 'bytes'.
- Enable Keepalives:** A checkbox that is checked.

En cuanto a los protocolos, el protocolo por defecto es el *t3*. Este protocolo es propio de Weblogic y es el que utiliza para comunicarse entre los servidores.



MiDominio> Servers> Servidor3

Connected to porta-rvg:7001 Active Domain: MiDominio Dec 22, 2002 9:02

Configuration **Connections** Monitoring Control Logging Dep

SSL SSL Ports HTTP jCOM Tuning **Protocols** Summa

⚠️ ? Default Protocol: t3

⚠️ ? Default Secure Protocol: t3s

? T3 Max Message Size: 100000 bytes

? T3 Message Timeout: 60 seconds

? HTTP Max Message Size: 100000 bytes

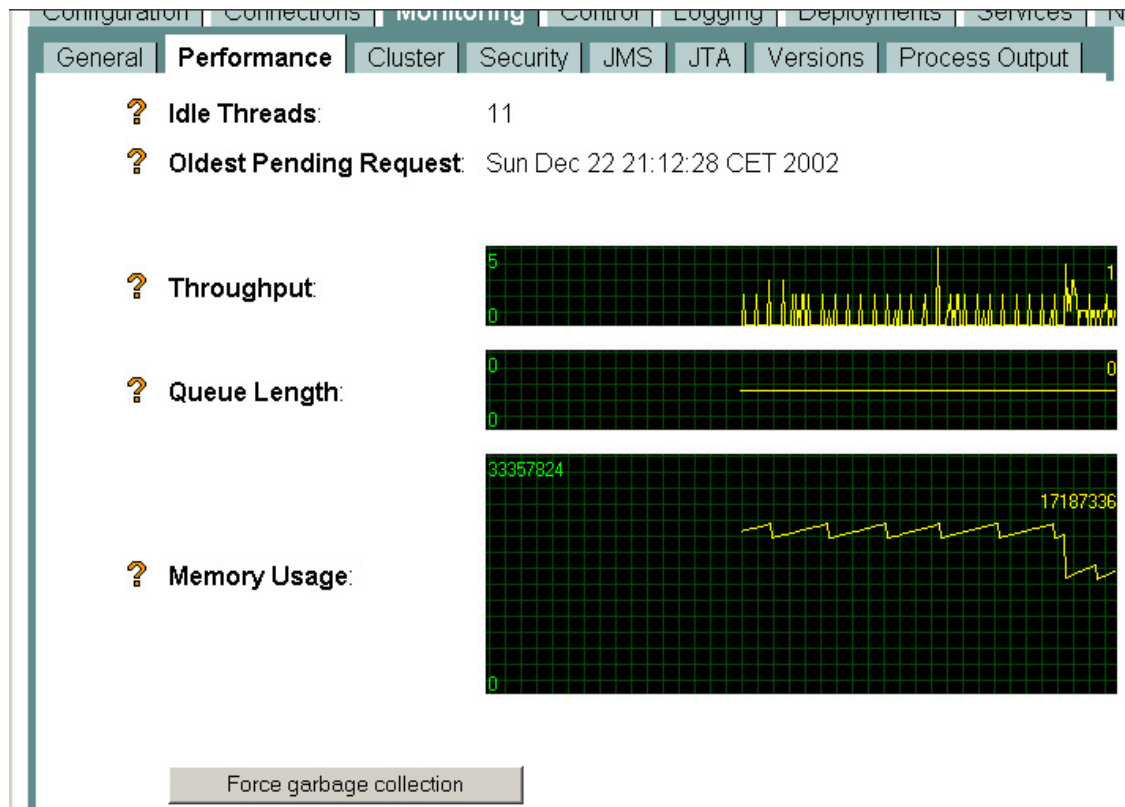
? HTTP Message Timeout: 60 seconds

? ☐ Enable Tunneling

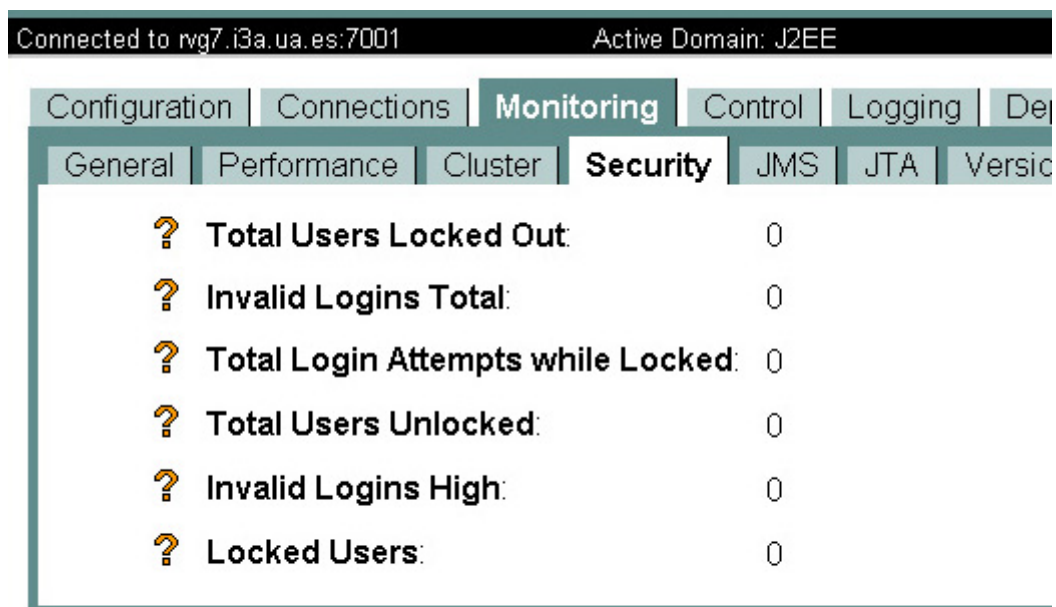
⚠️ ? Tunneling Client Ping: 45 seconds

En la solapa *Monitoring->Performance* podemos monitorizar en tiempo real algunas características del servidor:

- **Idle Threads:** es el número de hilos de ejecución disponibles.
- **Oldest Pending Request:** es el instante de tiempo en el que se produjo la última petición que no ha sido procesada por este servidor.
- **Throughput:** es la cantidad de datos que están pasando por el servidor. Se mide en kilobytes por segundo.
- **Queue Length:** es el número de peticiones pendientes de que un hilo de ejecución se libere.
- **Memory usage:** es la cantidad de memoria usada por el servidor. Al final de la figura se puede ver un botón con el título *Force Garbage Collector*. Si lo pinchamos forzamos el recolector de basura y se libera memoria. Eso es lo que ha sucedido cuando en la gráfica se ve el pico de caída.



La solapa *Security* nos permite monitorizar los intentos de entrada al sistema inválidos y el número de usuarios bloqueados. Puede servir para monitorizar intentos de acceso inválidos.



[View server log](#) [View JNDI tree](#)

En *Versions* vemos toda la información tanto del servidor de aplicaciones como del sistema operativo y Java.

The screenshot shows the 'Versions' tab in the WebLogic Administration Console. The top status bar indicates 'Connected to rvg7.i3a.ua.es:7001', 'Active Domain: J2EE', and the date 'Dec 22, 2002 9:28:01 PM C'. The navigation tabs include Configuration, Connections, Monitoring (selected), Control, Logging, Deployments, General, Performance, Cluster, Security, JMS, JTA, Versions (selected), and Process Output. The main content area displays the following system information:

WebLogic Version:	WebLogic Server 7.0 SP1 Mon Sep 9 22:46:58 PD 206753
JDK Vendor:	Sun Microsystems Inc.
JDK Version:	1.3.1_03
Operating System:	Linux
OS Version:	2.4.18-14

At the bottom of the console, there are two links: [View server log](#) and [View JNDI tree](#).

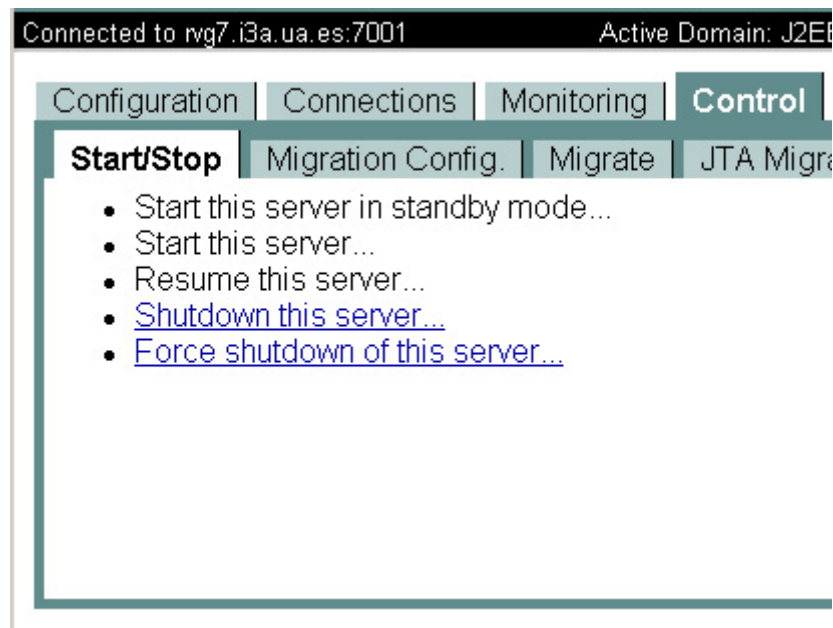
En la última solapa se puede visualizar la salida del servidor.

The screenshot shows the 'Process Output' tab in the WebLogic Administration Console. The top status bar is identical to the previous screenshot. The navigation tabs are the same, but 'Process Output' is now selected. The main content area displays three links with document icons:

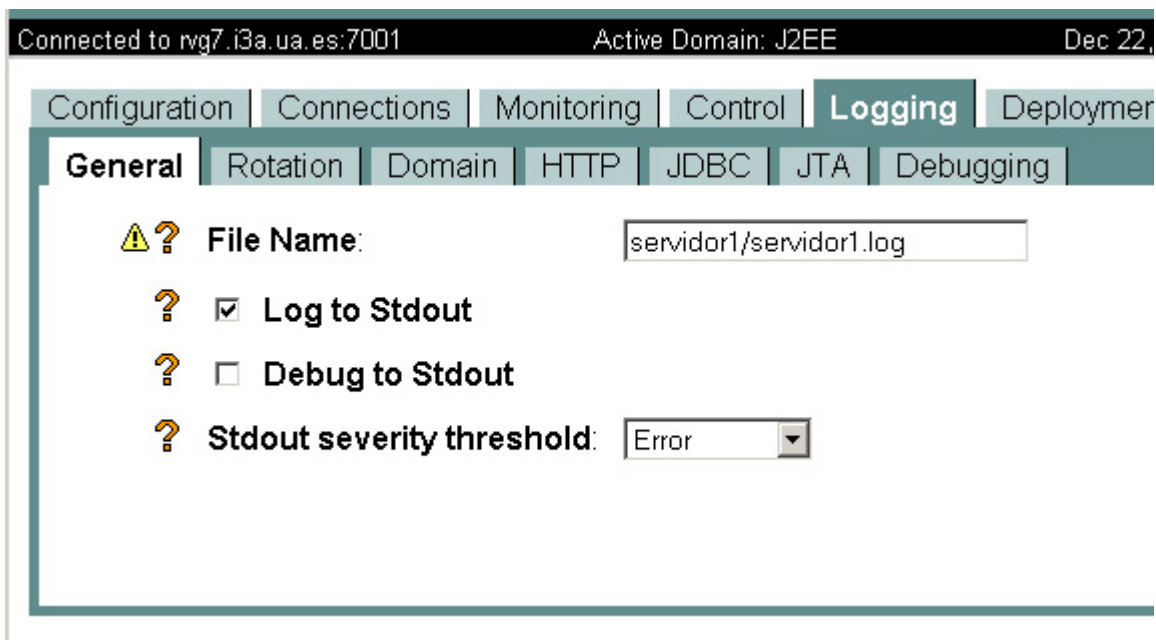
- [View Server output](#)
- [View Server error output](#)
- [View Node Manager output](#)

At the bottom, the same links are present: [View server log](#) and [View JNDI tree](#).

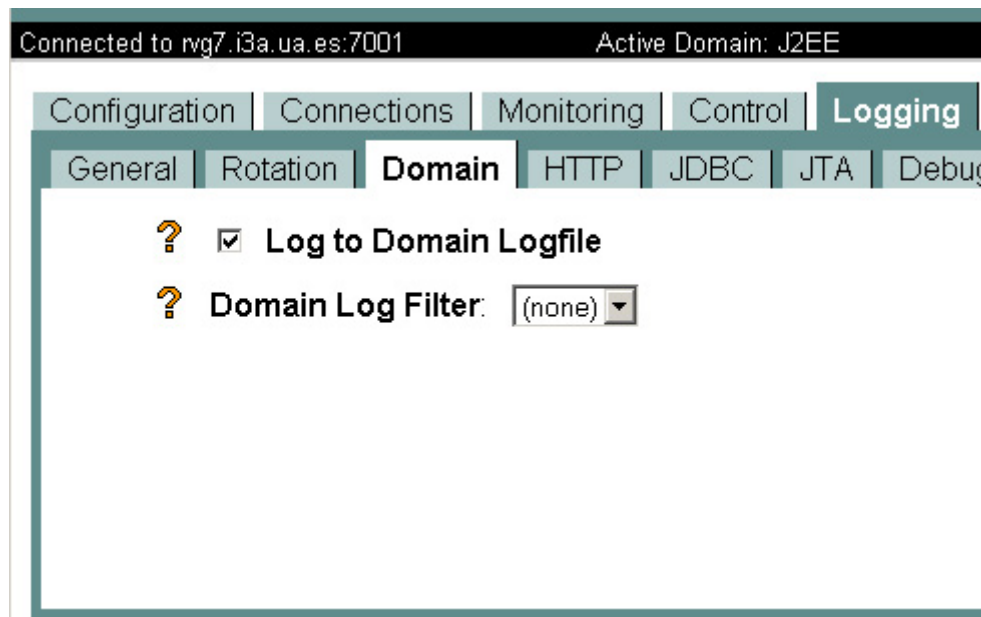
En la solapa *Control->Start/Stop* podemos arrancar y parar el servidor en distintos modos. La primera opción permite arrancar el servidor en modo de espera, donde no atiende peticiones. La segunda opción permite un arranque normal del servidor. La tercera arranca un servidor en modo espera. La cuarta opción para el servidor y la quinta fuerza la parada del servidor (en las ocasiones en las que el servidor no responda a la parada normal, utilizaremos esta opción para matarlo (*kill*))



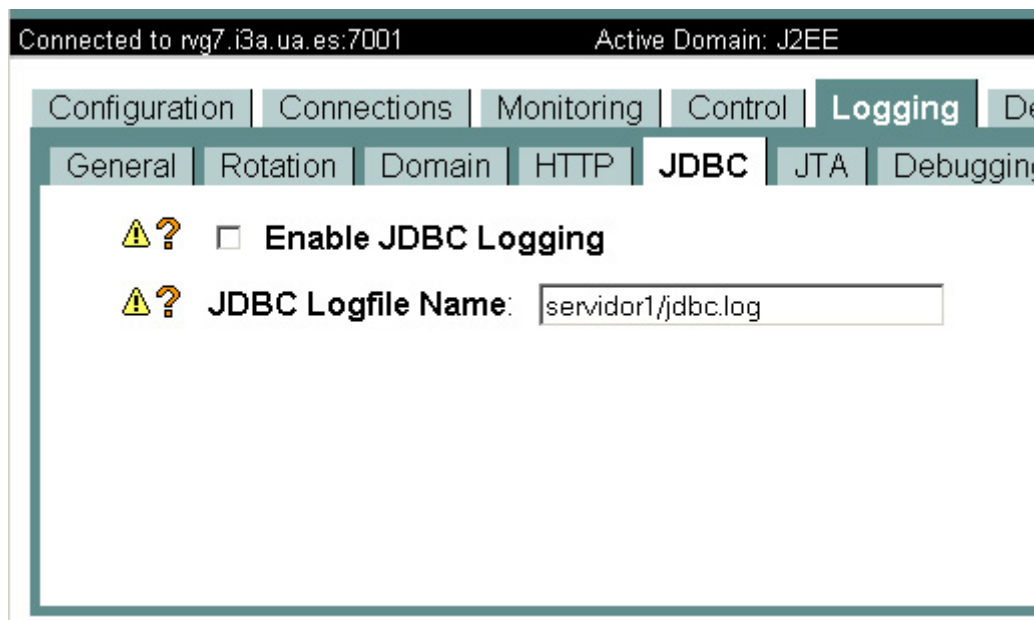
En la solapa *Logging* podemos configurar las opciones del fichero que contendrá los mensajes del servidor. En la siguiente figura se muestran los datos más genéricos, ya explicados en el apartado del log de dominio. La opción *Stdout severity threshold* permite cambiar el nivel de error que muestra los mensajes. Por defecto se mostrará un mensaje si existe un error. Podemos bajar o subir el nivel de error para que muestre un mensaje, por ejemplo, cuando existe una advertencia. La siguiente solapa, *Rotation*, permite modificar el tipo de rotación del fichero log. Las opciones también se detallaron anteriormente.



La solapa *Domain* permite que la salida del servidor se almacene en el fichero log del dominio.

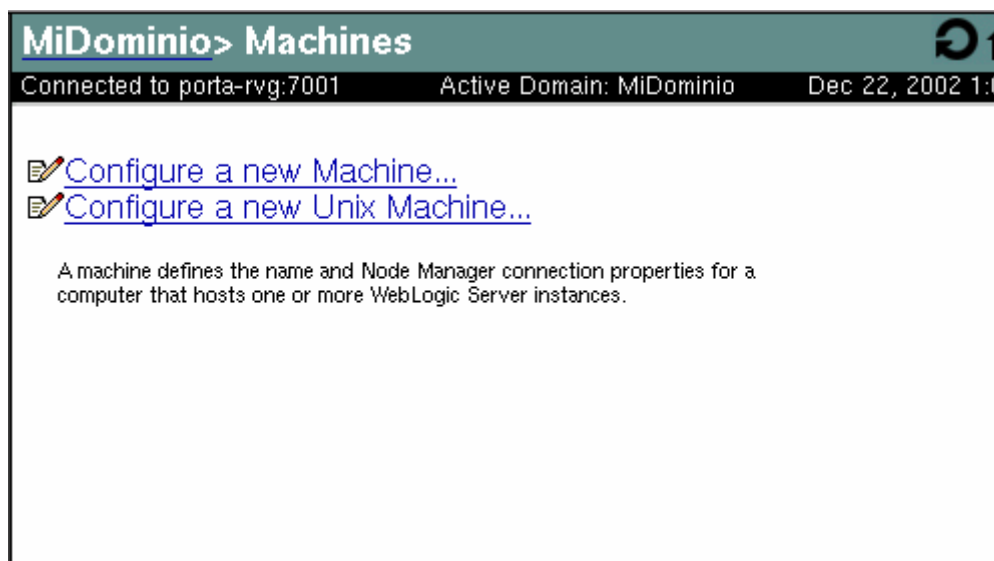


Si nos fijamos el fichero log para JDBC está deshabilitado por defecto. Esto es debido a que las operaciones de JDBC son numerosas y pueden llegar a saturar el sistema de ficheros. No permite la rotación del fichero log.






3.1.3. Definición de máquinas

Una máquina es un reflejo de una máquina física que soporta nuestro sistema. Weblogic se puede instalar en multitud de plataformas (Unix, Windows, Solaris, etc.). Una de las principales ventajas de Weblogic es que se pueden utilizar distintos tipos de máquinas en el mismo sistema. Por ejemplo, podemos tener una máquina Linux con un par de servidores y otra máquina Windows NT con otro par, funcionando todo dentro del mismo dominio y dando el mismo servicio. Cuando pinchamos en el icono *Machines* nos permite visualizar las máquinas que tiene nuestro sistema y también nos permite configurar nuevas máquinas. Podemos configurar una máquina de tipo Unix (Linux, etc.) o un tipo distinto de máquina. La diferencia es que en las máquinas de tipo Unix nos permite configurar opciones adicionales.





Cuando pinchamos en *Configure a new Unix Machine...* nos aparece la figura siguiente donde podemos dar nombre a la máquina. Este nombre es propio de Weblogic, es decir, no es nombre DNS ni IP. Las siguientes opciones, que sólo aparecen en máquinas Unix, son el nombre de usuario y de grupo con los que se ejecutarán los procesos instanciados en esa máquina. Una vez configurados estos datos, pinchamos en *Create*.



MiDominio> Unix Machines> Create a new UnixMachine...   



Connected to porta-rvg:7001 Active Domain: MiDominio Dec 22, 2002 1:10:27 PM CET



Configuration Monitoring



General Node Manager Servers Notes

  **Name:**




  ☐ **Enable Post-Bind UID**

  **Post-Bind UID:**

  ☐ **Enable Post-Bind GID**

  **Post-Bind GID:**


Debemos asignar los servidores que se ejecutarán en esa máquina. Para ello pinchamos en la solapa *Servers* y seleccionando el servidor en el grupo *Available* pinchamos en la flecha para pasarlo a *Chosen*. El resto de opciones se comentarán más adelante. La creación de una máquina necesita de una parada de los servidores.

MiDominio> Machines> maquina1   

Connected to porta-rvg:7001 Active Domain: MiDominio Dec 22, 2002 1:11:51 PM C

Configuration Monitoring

General Node Manager **Servers** Notes

 **Servers:**

Available

Servidor1
Servidor2

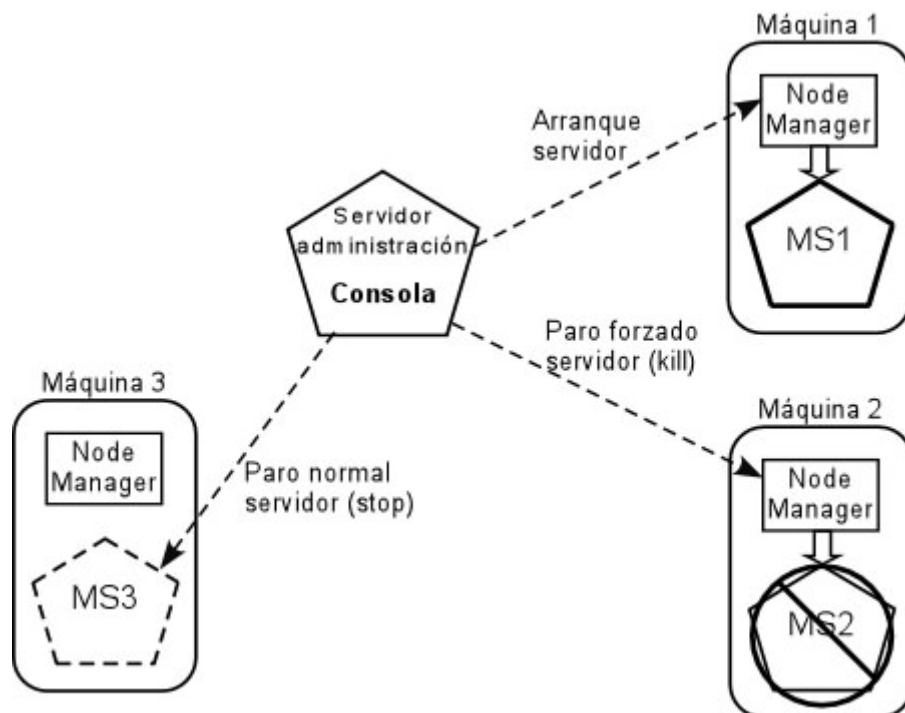
➔

➞

Chosen

3.2. Uso del NodeManager

Como ya hemos visto, un servidor de aplicaciones no es más que una instancia de la clase `weblogic.Server`. El `NodeManager` es una clase Java que nos va a permitir instanciar otras clases y así poder arrancar los servidores de aplicaciones. En concreto nos va a permitir arrancar servidores administrados uno a uno, todo un cluster o un dominio a la vez. El `NodeManager` es independiente del servidor de aplicaciones y debemos instanciar uno por cada máquina física que contenga servidores administrados. Recibe peticiones directamente del servidor de administración a través de la consola y puede ser ejecutado como un demonio Unix o un servicio Windows.



El `NodeManager` realiza las tareas mostradas en la anterior figura: permite arrancar un servidor que se encuentre en su máquina; forzar su paro (sólo para casos en los que no responda) y pararlo. Esta última opción, la de paro normal, también la puede realizar el servidor de administración directamente.

Para configurar el `NodeManager` debemos seguir los siguientes pasos:

1. Configurar el `NodeManager` y arrancarlo en todas las máquinas. Para ello vamos a configurar el script de arranque. Por motivos de seguridad el `NodeManager` consulta un fichero donde especificamos las direcciones que pueden comunicarse con él. En el caso de que el `NodeManager` se encuentre en la misma máquina, simplemente dejando el `localhost` es suficiente. Si está en otra máquina debemos dar permiso a la IP de la máquina del servidor. Para realizar esto editamos el fichero: `$BEA_HOME/weblogic700/common/nodemanager/config/nodemanager.hosts`

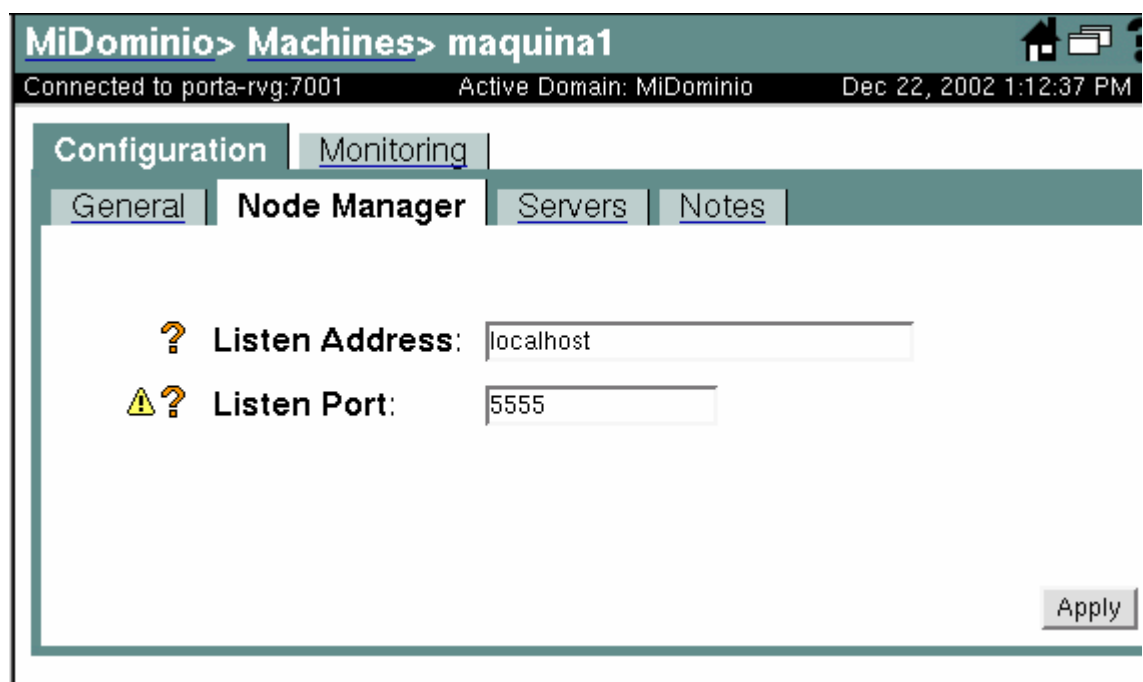
Por defecto tenemos el *localhost*. Si el NodeManager está en una máquina distinta añadimos la dirección IP de la máquina del servidor. Podemos añadir la dirección DNS del servidor, pero para que funcione tenemos que añadir una opción al ejecutable que arranca el NodeManager:

-Dweblogic.nodemanager.reverseDNSenabled=true

Ya podemos ejecutar el fichero `$BEA_HOME/weblogic700/server/bin/startNodeManager.sh`

El NodeManager está funcionando. No es necesario que el NodeManager esté funcionando cuando arrancamos el servidor de administración.

2. Con el NodeManager funcionando, vamos a configurar el servidor de administración para que lo pueda utilizar. Entramos en la consola de administración y pinchamos en la máquina donde va a estar el NodeManager. Entramos en la solapa de *Configuration* y pinchamos en *Node Manager*. Podemos configurar la dirección de escucha y el puerto (por defecto está el 5555).



Con el NodeManager funcionando ya podemos utilizarlo. En la solapa de *Monitoring* podemos comprobar si podemos establecer comunicación con el NodeManager de la máquina seleccionada. También podemos consultar el fichero log para detectar posibles fallos.

MiDominio> Machines> maquina1
 Connected to porta-rvg:7001 Active Domain: MiDominio Dec 22, 2002 1

[Configuration](#) | **Monitoring**

Node Manager Status | [log](#)

State : RUNNING



bea.home : /home/miguel/bea

weblogic.nodemanager.javaHome : /home/miguel/bea/jdk131_03

weblogic.nodemanager.listenAddress : localhost

weblogic.nodemanager.listenPort : 5555

CLASSPATH :
 /home/miguel/bea/jdk131_03/lib/tools.jar:/home/miguel/bea/weblog

MiDominio> Machines> maquina1  
 Connected to porta-rvg:7001 Active Domain: MiDominio Dec 22, 2002 1:51:30 PM

[Configuration](#) | **Monitoring**

Node Manager Status | **log**

<22-dic-02 13:41:59 CET> <Info> <NodeManager> <Starting Node Manager...>

<22-dic-02 13:41:59 CET> <Info> <NodeManager> <For information on command line options, try "weblogic.nodemanager.NodeManager help"...">

<22-dic-02 13:41:59 CET> <Info> <NodeManager> <Setting listenAddress to localhost..>

<22-dic-02 13:41:59 CET> <Info> <NodeManager> <Setting listenPort to 5.555..>

<22-dic-02 13:41:59 CET> <Info> <NodeManager> <Using KeyStore /home/miguel/bea/weblogic700/server/lib/cacerts..>

<22-dic-02 13:41:59 CET> <Info> <NodeManager> <Setting BEA home

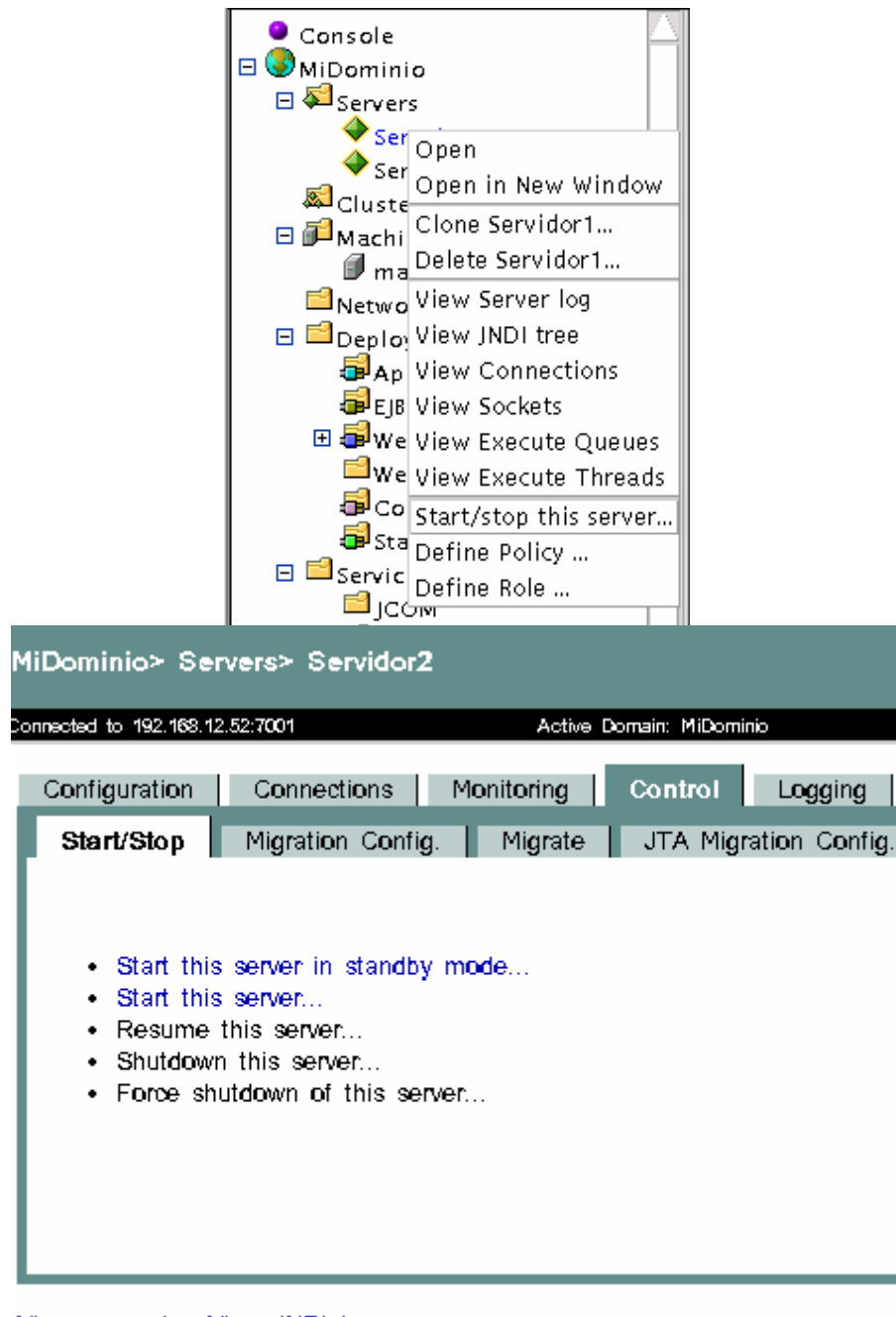
Por cada servidor que queramos arrancar, debemos configurar los datos del arranque remoto. Para ello pinchamos en el servidor y en la solapa de *Configuration*, en la subsolapa *Remote Start*. La siguiente información hace referencia siempre a directorios de la máquina remota:

- **Java Home:** directorio donde tenemos instalado Java.

- **BEA Home:** directorio donde tenemos instalado BEA
- **Root Directory:** directorio de trabajo del servidor. Podemos utilizar el mismo nombre de dominio dentro del directorio *users_projects*.
- **Class Path:** aquí podemos pasarle el *classpath* al comando Java. Como mínimo debemos añadir el fichero *weblogic.jar* que se encuentra en \$BEA_HOME/weblogic700/server/lib/weblogic.jar.
- **Arguments:** argumentos adicionales para el comando Java. Los mostrados en la figura siguiente indican la cantidad mínima y máxima de memoria a utilizar.
- **Security Policy File:** fichero de política de seguridad utilizado por Weblogic. Por defecto podemos usar el fichero: \$BEA_HOME/weblogic700/server/lib/weblogic.policy
- **Username:** el nombre de usuario para arrancar el servidor.
- **Password:** contraseña. Debemos indicarle la contraseña asociada al usuario.



Para arrancar un servidor a través del NodeManager pinchamos en el nombre del servidor con el botón derecho y seleccionamos *Start/stop this server...* Nos aparecerá la figura mostrada más abajo en la cual podemos poner en marcha el servidor. También podemos poner en marcha todo el dominio, pinchando con el botón derecho sobre el nombre del dominio y todos los servidores de un cluster pinchando sobre el nombre del cluster.



Por último, si queremos que el NodeManager se arranque cuando se encienda la máquina, debemos indicar al sistema operativo que lo haga. Vamos a detallar cómo se puede arrancar en Linux. Vamos a utilizar el fichero `/etc/rc.local`. En este fichero vamos a añadir un par de líneas que arrancarán el NodeManager. Las líneas a añadir serán:

```
su - weblogic -c "cd /home/weblogic/boa/weblogic700/server/bin;
./startNodeManager.sh"
```

El comando `su` permite cambiar de usuario. Aquí cambiamos a *weblogic* que es quien ha instalado el servidor de aplicaciones. Con la opción `-c` indicamos que ejecute un comando que es el que viene a continuación, cambiando al directorio indicado y ejecutando el NodeManager.

También podemos hacer lo mismo con el servidor de administración, utilizando el siguiente comando:

```
su - weblogic -c "cd /home/weblogic/bea/users_projects/MiDominio;  
./startWebLogic.sh"
```

3.3. Despliegue de aplicaciones

Una aplicación web es un conjunto de recursos en la parte del servidor que crean una aplicación interactiva. Estos recursos pueden contener todos o algunos de los siguientes componentes: servlets, JavaServer Pages, documentos estáticos (páginas HTML, imágenes, documentos PDF, etc.), clases, applets y beans en la parte del servidor. Una aplicación web se nos puede presentar de dos formas:

- Un fichero comprimido *.war*
- Una estructura de directorios expandida manteniendo la siguiente estructura:
 - NombreAplicación (nombre de la aplicación y documento raíz de la aplicación)
 - META-INF (información para herramientas de archivo)
 - Ficheros *.html*, *.jsp*
 - WEB-INF (directorio que contiene ficheros que no son servidos a clientes)
 - *classes* (directorio que contiene clases, servlets y applets)
 - *lib* (directorio con ficheros *.jar* usados en la aplicación)
 - *web.xml* (fichero descriptor de despliegue de la aplicación)
 - *weblogic.xml* (fichero descriptor para mapear recursos propios de Weblogic)

3.3.1. Despliegue de una aplicación en un fichero war











Para descargar una aplicación contenida en un fichero *war* se procede de la siguiente manera. Primero pinchamos en el icono de *Web applications*. Nos aparecerá una figura como la siguiente en la cual se nos muestra las aplicaciones disponibles y se nos permite configurar nuevas. Vamos a pinchar en *Configure a new Web Application*.

J2EE> Web Applications   





Connected to rvg7.i3a.ua.es:7001 Active Domain: J2EE Dec 23, 2002 11:59:13 AM

 [Configure a new Web Application...](#)

 [Customize this view...](#)

Name	Application	URI	Deployment Order	
proxyApp	proxyApp	proxyApp.war	1000	 
benefits	benefits	benefits.war	1000	 
certificate	certificate	certificate.war	1000	 
DefaultWebApp	DefaultWebApp	DefaultWebApp	1000	 
shoppingcart	shoppingcart	shoppingcart.war	1000	 

El primer paso es localizar el fichero que contendrá nuestra aplicación. Hablamos de ficheros `.war` pero podemos desplegar cualquiera de los tipos mostrados: `.jar`, `.war`, `.rar` y `.ear`. Pinchamos en el paso 1: *upload it through your browser*.

Locate Application or Component to configure    

Connected to rvg7.i3a.ua.es:7001 Active Domain: J2EE Dec 23, 2002 11:59:13 AM

This wizard will guide you through the process of configuring and deploying a J2EE application or module. This can be any one of the following types of files:

- A **.jar** containing EJBs (Enterprise Java Beans)
- A **.war** (Web Application Archive) containing JSPs and Servlets
- A **.rar** (Resource Adapter Archive) containing a JCA Connector module
- An **.ear** (J2EE Enterprise Application Archive) containing any of the above

Step 1. The `.ear`, `.war`, `.jar`, or `.rar` file you wish to configure and deploy must be available in the Admin servers file system. It may already be there; if it is not, you can either copy it there yourself or [upload it through your browser](#) into the directory selected below.

Step 2. Select the `.ear`, `.war`, `.jar`, or `.rar` file you would like to configure and deploy. Note that you may also configure an 'exploded' application or component by simply selecting its root directory. Click on the [select] link to the left of the desired directory or file to choose it and proceed to the next step.

Listing of [rvg7.i3a.ua.es/home/weblogic/bea/user_projects/J2EE](#)

Nos aparece la siguiente figura en la que se nos permite buscar el fichero en nuestra máquina local. Seleccionamos el fichero y pinchamos en *Upload*.

Install or Update an Application

Connected to rvq7.i3a.ua.es:7001
Active Domain: J2EE
Dec 23, 2002 1

Upload and Install an Application

Click on the 'browse' button below to locate an application archive on the machine from which you are browsing. When you have located the file, click 'upload' to install it on this WebLogic Administration Server. The following types of application files can be uploaded:

- A **.jar** containing EJBs (Enterprise Java Beans)
- A **.war** (Web Application Archive) containing JSPs and Servlets
- A **.rar** (Resource Adapter Archive) containing a JCA Connector module
- An **.ear** (J2EE Enterprise Application Archive) containing any of the above

Note: if you browse for the file, you may have to adjust the file-type filter to 'All' in order to find .jar, .war, .rar and .ear files.

Nos vuelve a aparecer la anterior página, en la cual debemos realizar el paso 2 que consiste en seleccionar la aplicación que queremos desplegar. Pinchamos en la aplicación recién descargada.

- A **.rar** (Resource Adapter Archive) containing a JCA Connector module
- An **.ear** (J2EE Enterprise Application Archive) containing any of the above

Step 1. The .ear, .war, .jar, or .rar file you wish to configure and deploy must be in the Admin server's file system. It may already be there; if it is not, you can either upload it yourself or [upload it through your browser](#) into the directory selected below.

Step 2. Select the .ear, .war, .jar, or .rar file you would like to configure and deploy. You may also configure an 'exploded' application or component by simply selecting the root directory. Click on the [select] link to the left of the desired directory or file and proceed to the next step.

Listing of rvq7.i3a.ua.es/home/weblogic/bea/user_projects/J2EE

[\[Up to parent directory\]](#)
[\[select\] .wlstaging](#)
[\[select\] NodeManagerClientLogs](#)
[\[select\] admin](#)
[\[select\] applications](#)
[\[select\] logs](#)
[\[select\] proxy](#)
[\[select\] servidor1](#)
[\[select\] userConfig](#)
[\[select\] benefits.war](#)
[\[select\] fags.war](#)
[\[select\] proxyApp.war](#)
[\[select\] shoppingcart.war](#)

Pasamos entonces al paso 3. Se nos advierte qué fichero estamos configurando y nos permite asignar la aplicación a uno o varios servidores o a

todo un cluster. Por último le asignamos un nombre a la aplicación. Este nombre tiene un carácter informativo. Una vez introducidos los datos pinchamos en *Configure and Deploy* para que se realice el despliegue.

Step 3. You have chosen to configure

`/home/weblogic/bee/user_projects/J2EE/faq.s.war`

Select the Servers and/or Clusters on which you would like to deploy this application initially.
(You can reconfigure deployment targets later if you wish).

The screenshot shows a deployment configuration window with two main sections. The top section is for servers, with 'Available Servers' on the left containing a list: 'admin', 'proxy', 'servidor2', and 'servidor3'. In the center are two arrow buttons (one pointing right, one pointing left). On the right is 'Target Servers' with a list containing 'servidor1'. The bottom section is for clusters, with 'Available Clusters' on the left containing a list: 'clusterJ2EE'. In the center are two arrow buttons. On the right is 'Target Clusters' which is empty.

Step 4. Enter a name for this application.

faq.s

Step 5. Press 'Configure and Deploy' to configure and deploy the application, or 'Cancel' to leave the Domain unchanged.

The screenshot shows the bottom of the deployment configuration window with two buttons: 'Configure and Deploy' and 'Cancel'.

Una vez hemos pinchado en *Configure and Deploy* nos aparecerá una ventana como la siguiente. Se nos muestra información de la actividad de despliegue. Al cabo de un tiempo nos aparecerá el resultado del despliegue. Si hemos conseguido desplegar la aplicación aparecerá *true* en la columna *Deployed* del servidor correspondiente. Si ha habido algún problema podemos consultar el log del servidor correspondiente.

J2EE> Web Applications> fags

Connected to rvg7.i3a.ua.es:7001 Active Domain: J2EE Dec 23, 2002 12:03:05 PM

[Edit Web Application Deployment Descriptors...](#)

Configuration

Targets

Deploy

Monitoring

Notes

Deployment Status by Target:

Target	Target Type	Deployed	
servidor2	Server	false	<input type="button" value="Deploy"/>
servidor1	Server	false	<input type="button" value="Deploy"/>

Deploy or redeploy this component to all targets

Deployment Activity:

Description	Status	BeginTime	End Time	
Activate application fags on servidor1	Running	Mon Dec 23 12:03:02 CET 2002		<input type="button" value="Cancel"/>

Note: To configure additional deployment targets for this component, please move to the 'Targets' tab.

3.3.1. Despliegue de una aplicación en un directorio

En modo desarrollo y cuando estamos implementando una aplicación es normal que tengamos que modificar muy a menudo el código de la aplicación. Por ello es conveniente tener dicha aplicación en un directorio, no en un fichero. La aplicación sin comprimir la podemos dejar en cualquier directorio del sistema, pero es conveniente dejarla en el directorio *applications* del dominio. Dentro de ese directorio dejaremos uno nuevo que contendrá nuestra aplicación. Para configurar esta aplicación debemos realizar los siguientes pasos. En el paso que realizamos anteriormente cuando desplegábamos una aplicación en un fichero, seleccionamos el paso 2 el directorio *applications* (que será donde habremos dejado nuestra aplicación) pinchando sobre su nombre.

Step 1. The .ear, .war, .jar, or .rar file you wish to configure an available in the Admin servers file system. It may already be there either copy it there yourself or [upload it through your browser](#) if below.

Step 2. Select the .ear, .war, .jar, or .rar file you would like to configure. Note that you may also configure an 'exploded' application or selecting its root directory. Click on the [select] link to the left of the file to choose it and proceed to the next step.

Listing of rvq7.i3a.ua.es/home/weblogic/bea/user_project

[\[Up to parent directory\]](#)
[\[select\]](#) .wlstaging
[\[select\]](#) NodeManagerClientLogs
[\[select\]](#) admin
[\[select\]](#) applications
[\[select\]](#) logs
[\[select\]](#) proxy
[\[select\]](#) servidor1
[\[select\]](#) userConfig
[\[select\]](#) HelloWorld.ear
[\[select\]](#) benefits.war
[\[select\]](#) faqs.war
[\[select\]](#) jaxrpc-hello.war
[\[select\]](#) proxyApp.war
[\[select\]](#) shoppingcart.war

Nos aparecerá el nombre de nuestra aplicación (*faqs*) que es un directorio. Ahora pinchamos en *select* para seleccionar dicho directorio.

Locate Application or Component to

Connected to rvq7.i3a.ua.es:7001
Active Dor

This wizard will guide you through the process c application or module. This can be any one of th

- A **.jar** containing EJBs (Enterprise Java E
- A **.war** (Web Application Archive) contain
- A **.rar** (Resource Adapter Archive) contain
- An **.ear** (J2EE Enterprise Application Arc

Step 1. The .ear, .war, .jar, or .rar file you wish t available in the Admin servers file system. It ma either copy it there yourself or [upload it through](#) below.

Step 2. Select the .ear, .war, .jar, or .rar file you Note that you may also configure an 'exploded' selecting its root directory. Click on the [select] file to choose it and proceed to the next step.

Listing of [rvq7.i3a.ua.es/home/weblogic/be](#)

[\[Up to parent directory\]](#)

[\[select\] .wlnotdelete](#)

[\[select\] .wlnotdelete_proxy](#)

[\[select\] .wlnotdelete_servidor1](#)

[\[select\] DefaultWebApp](#)

[\[select\] fags](#)

[\[select\] certificate.war](#)

Ya podemos seguir los pasos vistos en la sección anterior para configurar la aplicación (selección de *targets*, nombre de la aplicación, etc.).

Una herramienta bastante útil es el editor del descriptor de la aplicación. Nos permite editar el fichero *web.xml* y modificar los datos allí contenidos. Para acceder a esta herramienta pinchamos en el nombre de una aplicación y luego pinchamos en el enlace *Edit Web Application ...* mostrado en la figura.

J2EE> Web Applications> faqs

Connected to rvq7.i3a.ua.es:7001 Active Domain: J2EE Dec

[Edit Web Application Deployment Descriptors...](#)

Configuration | Targets | **Deploy** | Monitoring | Notes

Deployment Status by Target:

Target	Target Type	Deployed	
servidor1	Server	true	<input type="button" value="Undeploy"/> <input type="button" value="Redeploy"/>

Undeploy this component from all targets
 Deploy or redeploy this component to all targets

La siguiente figura muestra un ejemplo en la edición de un descriptor.

Aplicacion de FAQs> WebAppDescriptor-8> Active Servlets> faqcentral

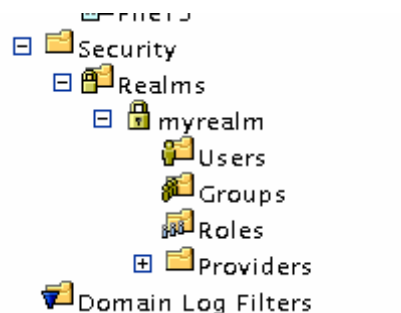
Connected to rvq7.i3a.ua.es:7001 Active Domain: J2EE

Configuration

⚠?	Description:	<input type="text"/>
⚠?	Display Name:	<input type="text"/>
⚠?	Small Icon File Name:	<input type="text"/>
⚠?	Large Icon File Name:	<input type="text"/>
⚠?	Servlet Name:	<input type="text" value="faqcentral"/>
⚠?	Servlet Class:	<input type="text" value="faqs.servlets.FAQCentralServ"/>
⚠?	Jsp File:	<input type="text"/>
⚠?	Load On Startup:	<input type="text" value="-1"/>
⚠?	Run As:	<input type="text" value="(none)"/> ▼

3.4. Gestión de seguridad

En la carpeta *Security* del applet de la izquierda tenemos todo lo referente a la gestión de seguridad. Tenemos creado un *Realm* por defecto que es el que se utiliza para guardar los usuarios que vamos creando. Podemos crear nuevos usuarios, grupos y roles. En Weblogic 7.0 la seguridad se maneja mediante políticas de seguridad. Las políticas de seguridad permiten definir *quién* tiene acceso a un determinado recurso. Adicionalmente podemos definir una restricción de tiempo de acceso a un recurso (de qué hora a qué hora se tiene acceso a un recurso). Por defecto un recurso no tiene protección hasta que el administrador le asigna una determinada política de seguridad. Los recursos a los que se pueden imponer políticas de seguridad son: la consola de administración, recursos de aplicación (web, EJB, ear, jar, etc.), JDBC, JNDI, EIS, JMS.



Un usuario puede ser una persona o una entidad software (cliente java). El usuario es único dentro del sistema y se identifica con su nombre y una contraseña. No existe el usuario invitado (*Guest*). Puede ser creado pero se recomienda no hacerlo por riesgo en la seguridad del sistema. Para crear un nuevo usuario pinchamos en *Users* y nos aparecerá una página como la de la siguiente figura. Se nos muestra los usuarios creados y tenemos un enlace para crear un nuevo usuario. También se nos informa de si un usuario está bloqueado y podemos eliminar el usuario.

Select Users

Connected to porta-rvg:7001 Active Domain: MIDominio

[Configure a new User...](#)

Filter By:

User	Description	Provider	Locked	
system	system	DefaultAuthenticator		

Al pinchar en *Configure a new User* nos aparece la siguiente figura. Los datos indicados son el nombre del usuario, una breve descripción y la contraseña. Una vez introducidos estos datos pinchamos en *Create*.

myrealm> Create User

Connected to porta-rvg:7001 Active Domain: MiDominio Dec 24, 2017

General Groups Details

? **Name::** miguel

? **Description:** Miguel Cazorla

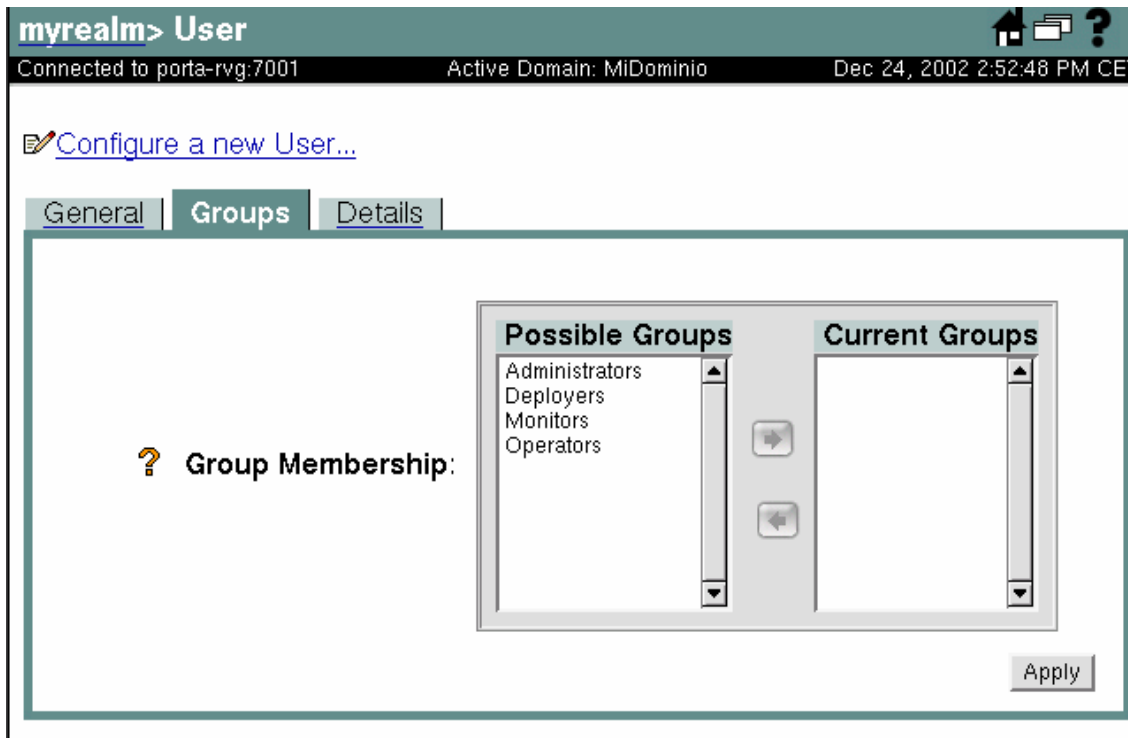
? **Password::** xxxxxxx

? **Confirm Password::** xxxxxxx

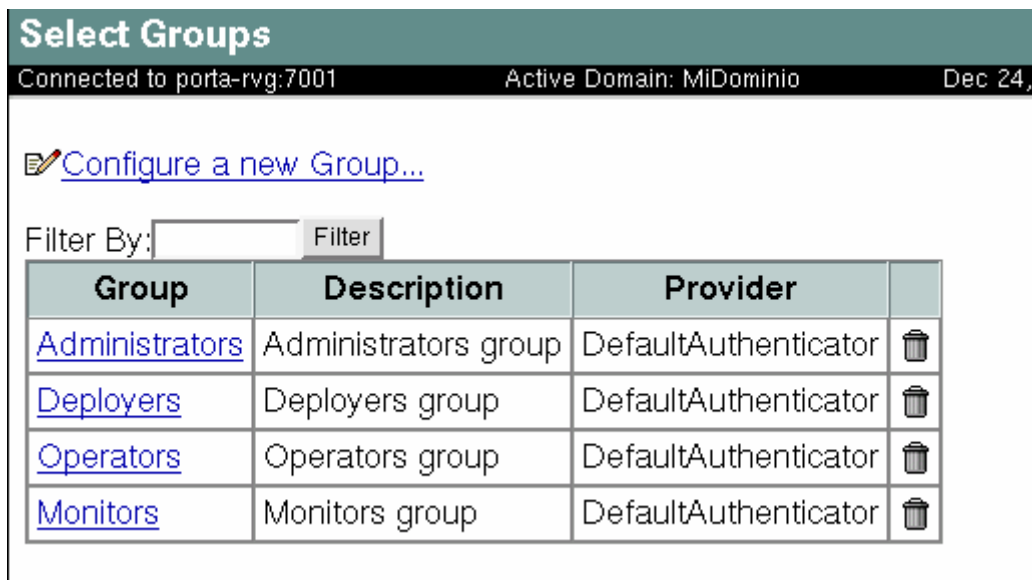
Una vez creado el usuario podemos asignarlo a un grupo. Un grupo es una agrupación de usuarios con alguna característica en común. Un usuario puede pertenecer a más de un grupo. Si asignamos una determinada política de seguridad a un grupo, dicha política es asignada a todos los usuarios del grupo. Por defecto existen seis grupos:

- **Users:** todos los usuarios identificados pertenecen a este grupo.
- **Everyone:** incluye el anterior más los usuarios no identificados.
- **Administrators:** es el grupo que permite administrar el dominio: arrancar y parar el sistema, visualizar y modificar todos los recursos.
- **Deployers:** los miembros de este grupo pueden desplegar aplicaciones.
- **Monitors:** sólo se permite la monitorización del sistema.
- **Operators:** permite cualquier operación con los servidores.

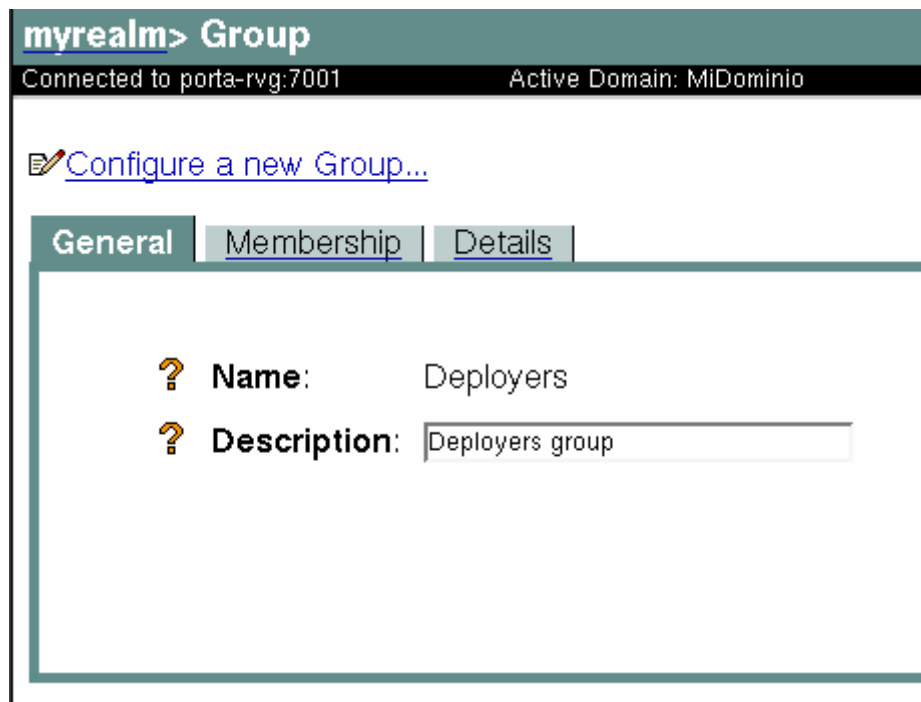
Al crear un usuario se nos permite asignarlo a un grupo.



También podemos crear y configurar nuevos grupos. Para crear un nuevo grupo pinchamos en el icono *Groups* y nos aparece la siguiente figura en la que podemos modificar un grupo ya existente o bien crear uno nuevo pinchando en *Configure a new Group*.



Damos el nombre y una breve descripción al grupo y pinchamos en *Apply*. En la solapa *Membership* podemos incorporar otros grupos como miembros de este.



myrealm> Group

Connected to porta-rvg:7001 Active Domain: MiDominio

[Configure a new Group...](#)

General Membership Details

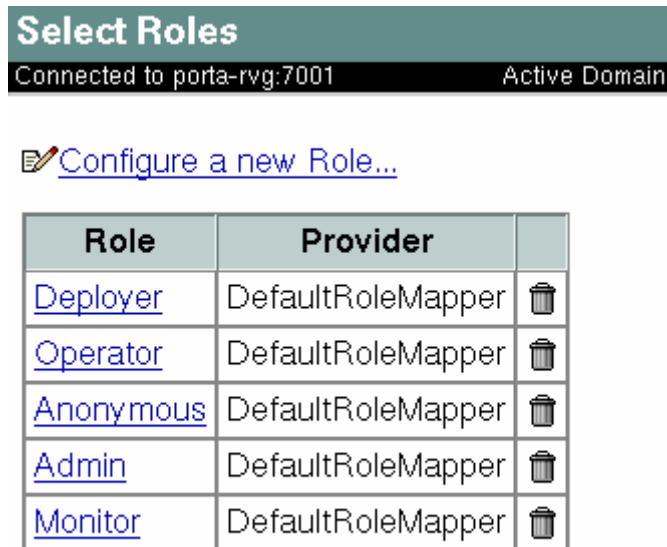
? Name: Deployers

? Description: Deployers group

El último apartado de la seguridad son los roles. Un rol es una asociación entre usuarios y recursos que permite establecer qué usuario tiene acceso a qué recurso. Un grupo es una agrupación estática (los usuarios están agrupados siempre de la misma forma). Un rol es dinámico, porque dependiendo de quién esté accediendo a un recurso y dependiendo de a qué recurso quiera acceder, tendrá permiso o no. Se recomienda agrupar usuarios en grupos y crear roles para luego asignar los roles a los grupos. Por defecto existen los siguientes roles:

- **Admin:** permite realizar cualquier operación dentro del sistema: despliegues, modificar recursos, etc.
- **Deployer:** permite ver la configuración del sistema y realizar despliegues de aplicaciones.
- **Operator:** permite ver la configuración del sistema y realizar operaciones con los servidores.
- **Monitor:** sólo permite visualizar la configuración de los servidores.
- **Anonymous:** es el rol más restrictivo y es asignado a todos los usuarios.

Cada uno de estos roles está asignado al grupo correspondiente (*Admin* al de *Administrator*, *Deployer* al de *Deployers*, etc.). Para crear un nuevo rol pinchamos en *Configure a new role*.



Las políticas de seguridad por defecto asociadas a los recursos de WebLogic se muestran en la siguiente tabla:

Recurso	Política de seguridad
WebLogic	
Recursos administrativos	Rol <i>Admin, Deployer, Operator, Monitor</i>
EIS, EJB, JMS, JDBC, JNDI, MBean	Grupo <i>Everyone</i>
Recursos servidor	Rol <i>Admin, Operator</i>
Recursos servicios web	Grupo <i>Everyone</i>

WebLogic, en su versión 7.0, sigue manteniendo la asignación de seguridad en los ficheros *web.xml* y *weblogic.xml*. La secuencia a seguir para asignar seguridad (restringir el acceso a determinados usuarios) a un recurso es el siguiente:

- Declarar un patrón URL en el fichero *web.xml*.
- Declarar un rol.
- Asociar el patrón con el rol.
- Mapear el rol con usuarios o grupos en el fichero *weblogic.xml*

Un ejemplo de la información que podemos poner en el fichero *web.xml* (además de la de mapeado utilizada hasta ahora) es la siguiente:

```

<security-constraint>
  <web-resource-collection>
    <web-resource-name>nombre aplicacion</web-resource-
name>
    <url-pattern>url a proteger</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>nombre del rol</role-name>
  </auth-constraint>
  <user-data-constraint>
    <description>SSL not required</description>
    <transport-guarantee>NONE</transport-guarantee>
  </user-data-constraint>
</security-constraint>

<security-role>
  <role-name>nombre del rol</role-name>
</security-role>
<login-config>
  <auth-method>BASIC</auth-method>
</login-config>

```

En *nombre aplicación* debemos poner el nombre de nuestra aplicación que queremos proteger. En *url a proteger* especificamos la url (directorios, jsp, html, etc.) Por ejemplo, */manager/** protegería todo el directorio *manager* de nuestra aplicación (cualquier fichero por debajo del directorio). Lo anterior define un recurso web y lo asociamos a un rol. Debemos especificar el nombre del rol y la etiqueta *security-role* crea el rol. La última etiqueta, *login-config*, permite especificar el método de autenticación: BASIC pide el nombre de usuario y contraseña mediante una ventana; FORM permite pedir la autenticación mediante un formulario y CLIENT-CERT mediante un certificado.

Por último debemos asociar el rol creado con un grupo o usuario de WebLogic. Para ello disponemos del fichero *weblogic.xml*.

```

<weblogic-web-app>
  <security-role-assignment>
    <role-name>nombre del rol</role-name>
    <principal-name>nombre de grupo o
usuario</principal-name>
  </security-role-assignment>
</weblogic-web-app>

```

En *nombre del rol* pondremos el rol previamente definido en el fichero *web.xml*. Como nombre de grupo o usuario debemos poner uno válido (definido) dentro de WebLogic.

3.5. Administración desde línea de comandos

La administración desde línea de comandos es una herramienta útil para el control del sistema. Nos permite realizar varias tareas de monitorización y comprobación del sistema. También es útil en modo producción, pues normalmente se deshabilita la consola de administración por motivos de seguridad.

El comando a ejecutar es el siguiente:

```
java -cp $BEA_HOME/weblogic700/server/lib/weblogic.jar
weblogic.Admin -url URL -username usuario -password contraseña
COMANDO argumentos
```

El comando no es más que una llamada a la clase *weblogic.Admin* que se encuentra en el fichero *weblogic.jar* proporcionado por BEA. El parámetro URL especifica la dirección URL del servidor de administración o bien del servidor contra el que vayamos a realizar el comando. Debemos especificar también el puerto en la dirección. El usuario y su contraseña asociada deben ser válidos para el comando que vamos a ejecutar. Los posibles comandos a utilizar pueden ser algunos de los siguientes:

CONNECT	Realiza el especificado número de conexiones y devuelve el tiempo (en milisegundos) total y medio de conexión.
ejemplo	<i>java weblogic.Admin -url http://localhost:7001 -username system -password weblogic CONNECT 25</i>
FORCESHUTDOWN	Termina de forma inmediata un proceso servidor pasado como argumento. Como URL se suele utilizar la del servidor de administración.
ejemplo	<i>java weblogic.Admin -url http://localhost:7001 -username system -password weblogic FORCESHUTDOWN Servidor1</i>
GETSTATE	Devuelve el estado del servidor pasado como argumento. Como URL se suele utilizar la del servidor de administración.
ejemplo	<i>java weblogic.Admin -url http://localhost:7001 -username system -password weblogic GETSTATE Servidor1</i>

HELP	Muestra ayuda de un comando.
ejemplo	<code>java weblogic.Admin -url http://localhost:7001 -username system -password weblogic HELP GETSTATE</code>
LICENSES	Lista las licencias instaladas en la máquina.
ejemplo	<code>java weblogic.Admin -url http://localhost:7001 -username system -password weblogic LICENSES</code>
PING	Envía un mensaje para verificar que un servidor está disponible y aceptando peticiones. Opcionalmente podemos pasarle dos argumentos: el número de veces intentos y el tamaño de cada paquete.
ejemplo	<code>java weblogic.Admin -url http://localhost:7001 -username system -password weblogic PING 10 1000</code>
RESUME	Mueve un servidor del estado STANDBY a RUNNING.
ejemplo	<code>java weblogic.Admin -url http://localhost:7001 -username system -password weblogic RESUME Servidor1</code>
SERVERLOG	Muestra el fichero log de un servidor. Se puede especificar un intervalo de tiempo a mostrar. En el ejemplo se muestra desde las 14:00 horas del 24 de diciembre a las 10:00 horas del 31 de diciembre.
ejemplo	<code>java weblogic.Admin -url http://localhost:7001 -username system -password weblogic SERVERLOG "2002/12/24 14:00" "2002/12/31 10:00"</code>
SHUTDOWN	Para la ejecución de un servidor.
ejemplo	<code>java weblogic.Admin -url http://localhost:7001 -username system -password weblogic SHUTDOWN Servidor1</code>
START STARTINSTANDBY	Arranca un servidor si tenemos disponible el Node Manager. El comando STARTINSTANDBY lo arranca en modo STANDBY
ejemplo	<code>java weblogic.Admin -url http://localhost:7001 -username system -password weblogic START Servidor1</code>

VERSION	Muestra la versión del software Weblogic.
ejemplo	<code>java weblogic.Admin -url http://localhost:7001 -username system -password weblogic VERSION</code>

Existen otros comandos y utilidades adicionales que se detallan a continuación. En todos ellos hace falta incluir el fichero *weblogic.jar* en el classpath.

dbping	Realiza una conexión a la base de datos especificada utilizando una clase de las proporcionadas por WebLogic.
	<pre>java utils.dbping ORACLE_THIN system oracle localhost:1521:j2eebd **** Success!!! **** You can connect to the database in your app using: java.util.Properties props = new java.util.Properties(); props.put("user", "system"); props.put("password", "oracle");</pre>
system	Obtiene información del sistema: la versión y el desarrollador de Java, el classpath, el nombre, arquitectura y versión del sistema operativo
ejemplo	<pre>java utils.system * * * * * java.version * * * * * 1.3.1_03 * * * * * java.vendor * * * * * Sun Microsystems Inc. * * * * * java.class.path * * * * * bea/weblogic700/server/lib/weblogic.jar * * * * * os.name * * * * * Linux * * * * * os.arch * * * * * i386 * * * * * os.version * * * * * 2.4.18-14</pre>
myip	Obtiene la dirección IP y el nombre DNS de la máquina
ejemplo	<pre>\$ java utils.myip Host rvq7.i3a.ua.es is assigned IP address: 192.168.2.17</pre>
Deployer	Controla el despliegue de aplicaciones
ejemplo: desplegar una	<code>java weblogic.Deployer -adminurl http://miguel.dccia.ua.es:7001 -source faqs.war -targets adminServer -user system -activate</code>

aplicación	<p>Enter a password for the user "system":weblogic Operation started, waiting for notifications...</p> <pre>#TaskID Action Status Target Type Application Source 0 Activate Success adminServer Server faqs faqs.war</pre>
ejemplo: eliminar una aplicación	<pre>java weblogic.Deployer -adminurl http://miguel.dccia.ua.es:7001 - name faqs -targets adminServer -user system -deactivate</pre> <p>Enter a password for the user "system":weblogic Operation started, waiting for notifications...</p> <pre>#TaskID Action Status Target Type Application Source 1 Deactivate Success adminServer Server faqs null</pre>
ejemplo: reactivar una aplicación	<pre>java weblogic.Deployer -adminurl http://miguel.dccia.ua.es:7001 - name faqs -targets adminServer -user system -activate</pre> <p>Enter a password for the user "system":weblogic Operation started, waiting for notifications...</p> <pre>#TaskID Action Status Target Type Application Source 2 Activate Success adminServer Server faqs null</pre>

Tema 4: Creación de un cluster

Este tema contiene las instrucciones básicas para obtener las características más potentes de un servidor de aplicaciones: la escalabilidad y la recuperación ante fallos. Para ello aprenderemos a configurar un cluster y determinadas características adicionales como un servidor proxy y la replicación de memoria.

Un cluster es una asociación de servidores WebLogic que actúan como si fueran uno sólo. Una aplicación desplegada en un cluster es respondida por cada servidor dentro del cluster. Si nuestro sistema observa un aumento en el número de peticiones, podemos incorporar nuevos servidores para soportar dicho aumento. Otra característica, la recuperación ante fallos, es muy importante en sistemas de alta disponibilidad. WebLogic nos va a permitir replicar las sesiones HTTP e incluso los servicios (por ejemplo, JDBC) para que se permita realizar copias de las sesiones en otros servidores. De esta forma, si el servidor que está sirviendo actualmente tiene algún problema o no responde, el servidor que contiene la copia de la sesión puede seguir respondiendo sin necesidad de comenzar una nueva sesión.

4.1. Configuración básica de un cluster

Para configurar un cluster pinchamos en el icono de cluster y nos aparecerá una figura como la siguiente. Pinchamos en *Configure a new Cluster*.



En la siguiente figura debemos empezar a configurar el cluster. Comentamos las distintas opciones:

- **Name:** el nombre que identificará el cluster.

- **Cluster Address:** son las distintas direcciones que participarán en el cluster, separadas por comas.
- **Default Load Algorithm:** es el algoritmo de carga a utilizar. Este algoritmo permite el balanceo de la carga (peticiones). Los posibles algoritmos son:
 - *Round-Robin.* La primera petición se asigna al primer servidor, la segunda al segundo y así sucesivamente hasta que se sobrepasa el último y se vuelve a empezar.
 - *Weight-based.* Este algoritmo permite balancear la carga ponderando el peso de cada servidor. Utilizando el campo **Cluster Weight** (ver siguiente figura) podemos asignar un determinado peso al servidor, para así permitir que servidores en máquinas más potentes respondan a más peticiones. Si, por ejemplo, asignamos a un servidor un peso 1, a otro 2 y a otro 3, el algoritmo asigna una petición al primer servidor, las dos siguientes al segundo y las tres siguientes al tercero.

The screenshot shows the 'J2EE> Servers> servidor1' configuration window. The 'Cluster' tab is selected, showing fields for 'Replication Group' (grupo1), 'Preferred Secondary Group' (grupo2), 'Cluster Weight' (100), and 'Interface Address' (empty). The window also shows 'Connections', 'Monitoring', 'Control', 'General', 'Memory', 'Deployment', and 'Tuning' tabs.

- *Random.* Elige el siguiente servidor de manera aleatoria.
- **Weblogic Plug-In Enabled:** no habilitado por defecto. Seguridad.
- **Service Age Threshold:** el número de segundos que tienen que diferir dos servicios para que uno sea considerado más viejo que el otro.
- **Client Cert Proxy Enabled:** no habilitado por defecto. Seguridad.

Connected to miguel.dccia.ua.es:7001 Active Domain: MiDominio Dec 27,

Configuration Servers Monitoring Notes

General Multicast

⚠️? **Name:** MiCluster

⚠️? **Cluster Address:** miguel.dccia.ua.es:6001,miguel.dc

⚠️? **Default Load Algorithm:** round-robin ▾

⚠️? ☐ **WebLogic Plug-In Enabled**

 ? **Service Age Threshold:** 180

⚠️? ☐ **Client Cert Proxy Enabled**

En la siguiente solapa debemos chequear la dirección de *multicast*. El *multicast* permite la comunicación entre los servidores del cluster. Esta dirección se configura desde el sistema operativo. Por defecto es la mostrada en la figura (el puerto por defecto es también el 7777). Si queremos chequear si funciona la dirección podemos hacer uso de una utilidad de WebLogic. Desde dos sesiones distintas del sistema operativo tecleamos el siguiente comando:

```
java -cp $BEA_HOME/weblogic700/server/lib/weblogic.jar
utils.MulticastTest
-n mensaje -a dirección
```

donde *mensaje* es el mensaje que se enviará desde a la dirección de multicast y *dirección* es la dirección multicast a utilizar. Ponemos mensajes distintos en cada sesión y debemos recibir los dos mensajes.

Connected to miguel.dccia.ua.es:7001 Active Domain: MiDominio Dec 27, 2002

Configuration Servers Monitoring Notes

General **Multicast**

⚠️ Multicast Address:

⚠️ Multicast Port:

⚠️ Multicast Send Delay:

⚠️ Multicast TTL:

⚠️ Multicast Buffer Size: k

Pasamos a la solapa *Servers* donde debemos indicar los servidores que participarán en el cluster. Los servidores deben estar parados para poder asignarlos al cluster.

Connected to miguel.dccia.ua.es:7001 Active Domain: MiDominio Dec 27, 2002 10:38:58 AM C

Configuration **Servers** Monitoring Notes

⚠️ Choose Servers for this Cluster:

Available		Chosen
adminServer	⇌	
proxy		
Servidor1		
Servidor2		

Apply


En la solapa *Monitoring* podemos saber el número de servidores configurados para el cluster y los activos en este momento. Si pinchamos en *Monitor server participation...* nos aparece la figura mostrada más abajo donde se nos muestra información de los distintos servidores.

Connected to miguel.dccia.ua.es:7001 Active Domain: MiDominio Dec 27, 2002 11:11:5


[Configuration](#) [Servers](#) **Monitoring** [Notes](#)

Number of Servers configured for this cluster: 2

Number of Servers currently participating in this cluster: 2

 [Monitor server participation in this cluster](#)

Connected to miguel.dccia.ua.es:7001 Active Domain: MiDominio Dec 27, 2002

 [Customize this view...](#)

Name	Machine	State	Servers	Resend Requests	Fragments Received	Log Mu Me
Servidor1		RUNNING	2	0	176	0
Servidor2		RUNNING	2	0	176	0

Para desplegar una aplicación al cluster (y así que todos los servidores del cluster respondan a la aplicación) debemos asignar la aplicación al cluster. Para ello, en la solapa *Targets* seleccionamos *Clusters* en vez de *Servers*. Es muy importante **no** desplegar la misma aplicación a un cluster y además a un servidor que forme parte del cluster.

[Edit Web Application Deployment Descriptors...](#)

[Configuration](#) **Targets** [Deploy](#) [Monitoring](#) [Notes](#)

[Servers](#) **Clusters** [Virtual Hosts](#)

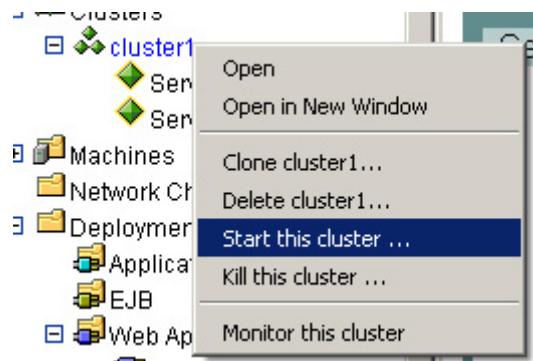
? **Targets-Cluster:**

Available

Chosen

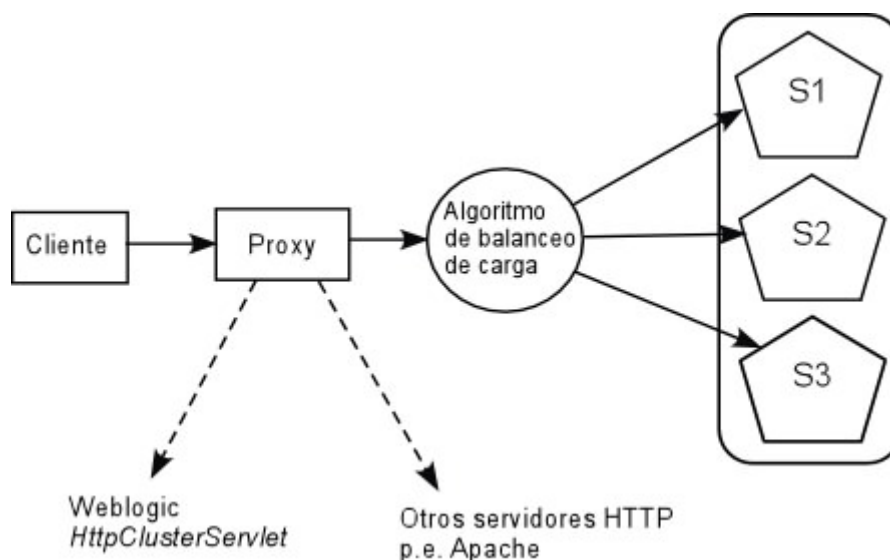
cluster1

También podemos arrancar todos los servidores que formen parte de un cluster a la vez, si tenemos configurado el NodeManager. Pinchamos con el botón derecho en el icono del cluster y seleccionamos *Start this cluster...*



4.2. Configuración de un servidor proxy

Una vez creado el cluster como se indicaba en el apartado anterior ya lo tenemos disponible para su utilización. Sin embargo, cada servidor tiene su propia dirección IP, por lo que si tenemos una aplicación desplegada en el cluster, ¿a qué dirección IP debe direccionar el cliente su petición? Podemos pedir a un servidor (que pertenezca al cluster) en concreto y éste responderá, pero perderemos el balanceo de carga. Para solucionar este problema se suele insertar un servidor proxy HTTP entre el cluster y el cliente. Este servidor proxy será un servidor de aplicaciones que tendrá asociada una aplicación que se encargará de realizar el balanceo de carga. También se puede utilizar otro servidor proxy (como Apache) o incluso un proxy hardware. En esta sección vamos a ver cómo podemos configurar un servidor proxy haciendo uso de una utilidad que incorpora Weblogic. Esta utilidad no es más que una clase que implementa un servlet para realizar el balanceo de carga.



Lo primero a realizar es la creación de un servidor de aplicaciones, al que llamaremos proxy. Vamos a asociar una aplicación a este servidor de aplicaciones. Para ello vamos a crear una aplicación vacía, que contendrá sólo el fichero de descripción de aplicación (*web.xml*) el cual utilizará un servlet de Weblogic. Creamos un fichero web.xml que contendrá la siguiente información:

```
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc. //DTD Web
Application 2.2//EN" "http://java.sun.com/j2ee/dtds/web-
App_2_2.dtd">
<web-app>

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

        <init-param>
            <param-name>WebLogicCluster</param-name>
            <param-value>
miguel.dccia.ua.es:7736:7737|miguel.dccia.ua.es: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>
```

Este fichero de descripción consta de las siguientes partes:

- El nombre del servlet y la clase que lo implementa. Hacemos referencia a la clase *weblogic.servlet.proxy.HttpClusterServlet*.
- El primer parámetro inicial identifica los servidores que forman parte del cluster. Debemos indicar cada uno de los servidores que forman parte del cluster de la siguiente manera: *nombre:puerto:puerto_seguro* *nombre* puede ser la dirección DNS o IP del servidor y a continuación ponemos el puerto y el puerto seguro del servidor. Para poner varios servidores los separamos con el símbolo | . El siguiente parámetro *DebugConfigInfo* es útil en modo desarrollo y proporciona información para depuración de errores.
- Debemos indicar el mapeado del servlet a patrones URL. En este caso hemos mapeado las extensiones jsp, htm, html, así como "/" que indica que cualquier petición que no pueda resolver el proxy se reenvía hacia los servidores del cluster.

Una vez creado este fichero creamos un directorio WEB-INF y movemos el fichero web.xml dentro de este directorio. Para crear la aplicación web utilizamos el siguiente comando:

```
jar cf proxyApp.war WEB-INF/*
```

donde *proxyApp.war* es el nombre que le hemos dado a la aplicación. Debemos desplegar la aplicación dentro de nuestro dominio y asociarla al servidor proxy. El último paso es hacer que esta aplicación sea la aplicación por defecto del servidor proxy. Para ello pinchamos en el servidor proxy y nos posicionamos en la solapa *Connections HTTP*. Debemos seleccionar *proxyApp* en la opción *Default Web Application*. A partir de este momento ya podemos disponer de balanceo de carga en el cluster creado. Ahora podemos atacar cualquier aplicación desplegada en el cluster solicitando la dirección del proxy.

Connected to miguel.dccia.ua.es:7001 Active Domain: MiDominio Dec 27, 2002 11:30:01 A

Configuration | **Connections** | Monitoring | Control | Logging | Deploy

SSL | SSL Ports | **HTTP** | jCOM | Tuning | Protocols | Summary

? Default Web Application: (none) ▾

! ? Frontend Host:

! ? Frontend HTTP Port:

! ? Frontend HTTPS Port:

! ? ☒ Send Server Header Enabled

! ? Post Timeout Secs:

! ? Max Post Time: seconds

! ? Max Post Size: bytes

! ? ☒ Enable Keepalives

! ? Duration: seconds

! ? HTTPS Duration: seconds

! ? ☐ Accept Context Path In Get Real Path

4.3. Configuración de la replicación de memoria

La última característica por configurar es la recuperación ante fallos. Cuando un cliente realiza una petición a un servidor, se crea una instancia de la sesión. Si un servidor se viene abajo (ya sea por problemas técnicos o por desconexión por mantenimiento de la máquina) y está dando servicio a un determinado cliente, la sesión HTTP, los servicios EJB y toda la memoria asociada a ese cliente se pierde. Para solucionar este problema, WebLogic nos permite configurar la replicación de memoria. La replicación de memoria nos permite especificar donde van a ser almacenadas las copias de las sesiones. Vamos a trabajar con grupos de replicación, que son una agrupación lógica de servidores relacionados en un cluster. Lo recomendable es que los servidores en la misma máquina estén en el mismo grupo de replicación. Cuando se crea una sesión, WebLogic crea una réplica de la sesión y la envía a otro servidor siguiendo este orden de preferencia:

1. Primero trata de encontrar un servidor que no esté en su misma máquina y que pertenezca a su grupo secundario preferido.

2. Si ningún servidor cumple lo anterior, trata de buscar un servidor que pertenezca a su grupo secundario preferido aunque no esté en otra máquina.
3. La tercera opción es que el servidor no pertenezca a su grupo secundario preferido, pero resida en otra máquina.
4. La última opción es que ni pertenezca a su grupo preferido ni resida en otra máquina.

Para definir los servidores en un grupo de replicación y en el secundario, debemos definir, en cada servidor, a qué grupo pertenecen. El nombre de los grupos nos definimos nosotros. En la siguiente figura podemos observar los nombres elegidos para un servidor.

The screenshot shows a web-based configuration interface for a server named 'Servidor1' under the domain 'MiDominio'. The interface has a top navigation bar with 'Servers' and 'Servidor1'. Below this, a status bar indicates 'Connected to miguel.dccia.ua.es:7001', 'Active Domain: MiDominio', and 'Dec 2'. The main content area has a tabbed interface with 'Configuration' selected. Under 'Configuration', there are sub-tabs: 'General', 'Cluster' (which is active), 'Memory', 'Deployment', 'Tuning', and 'Co'. The 'Cluster' tab contains four configuration items, each with a yellow warning icon and a question mark:

- Replication Group:** The text input field contains 'grupo1'.
- Preferred Secondary Group:** The text input field contains 'grupo2'.
- Cluster Weight:** The text input field contains '100'.
- Interface Address:** The text input field is empty.

Tema 5: JNDI

JNDI (Java Naming and Directory Interface) es un API para el acceso a diferentes servicios de nombres y directorios de una manera uniforme. Proporciona un mecanismo para enlazar programas Java con, por ejemplo, sistemas de ficheros, recursos de red, recursos de bases de datos o servicios de directorios (LDAP). El API de JNDI permite encontrar objetos y datos registrados en estos servicios y así mismo registrar sus propios objetos y datos para que sean usados por otros usuarios.

JNDI suele ser utilizado para lo siguiente:

- Servicio de nombres: asocia nombres lógicos a recursos. Se detalla en la siguiente sección. Este servicio es muy similar al servicio DNS de la web. Cuando solicitamos una dirección web, el DNS se encarga de buscar la dirección IP asociada y la devuelve.
- Servicio de directorio: haciendo uso de otro servicio (LDAP, sistema de ficheros, etc.) JNDI proporciona todas las funcionalidades que permiten estos servicios. JNDI puede ser visto como un driver JDBC en el sentido de que se encarga de "traducir" las llamadas. En el momento de que un EJB, por ejemplo, pide un recurso a JNDI, éste pasa la petición al servicio correspondiente (LDAP, por ejemplo) y devuelve el recurso. El servicio de directorio es muy parecido al servicio X.500.

5.1. JNDI: búsqueda de objetos mediante su nombre lógico

Un servicio de nombres proporciona un método para mapear nombres lógicos (por ejemplo, *databd*) con entidades u objetos (un recurso DataSource, un EJB, JMS, etc.). De esta manera, no tenemos que buscar un determinado objeto, sino que buscaremos su nombre lógico. Pensad cuando trabajábamos con las bases de datos. Obteníamos una conexión a partir de un driver y nos conectábamos a una base de datos en concreto, que estaba alojada en una determinada dirección. Si la base de datos cambiaba de nombre o cambiaba su dirección debíamos reflejar dichos cambios en nuestro código. Si utilizamos JNDI y asociamos un nombre lógico, por ejemplo *databd*, a un objeto DataSource el objeto DataSource es el que manejará los datos de la conexión con la base de datos. Nuestro código Java accede a JNDI y obtiene una referencia al objeto DataSource asociado con el nombre lógico. Si cambian los parámetros de conexión, debemos cambiar el objeto DataSource, pero no nuestro código Java, puesto que el nombre lógico no ha cambiado.

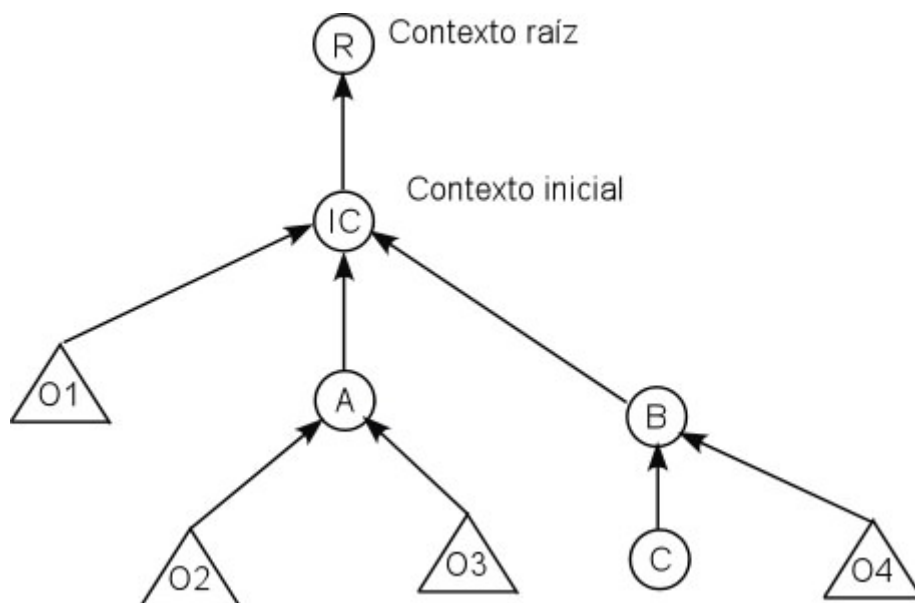
Vamos a definir un par de conceptos:

- **Contexto:** un contexto es similar a una conexión en JDBC. Cuando obtenemos un contexto de JNDI tenemos un flujo de información entre nuestra aplicación y el servicio deseado (de nombres o directorios).

Podemos entender un contexto como un directorio del sistema operativo. Dentro de ese directorio podremos tener más contextos u objetos, de la misma forma que en un directorio podemos tener más directorios u objetos (ficheros, enlaces, etc.) Cuando creamos un contexto en nuestro código primero deberemos especificar una serie de propiedades.

- **Enlace:** un enlace es una asociación entre un nombre atómico y un objeto.

JNDI suele tener asociado un árbol. En la siguiente figura se muestra un posible árbol JNDI. Todo árbol tiene un contexto raíz, sin embargo el que se utiliza para trabajar es el contexto inicial. A partir de este contexto podemos acceder a los objetos enlazados con este contexto (representados con un triángulo) o descender a subcontextos (los contextos se representan mediante círculos). De esta forma podemos agrupar objetos y organizarlos a nuestra manera. Dentro de JNDI podemos hacer referencia a subcontextos utilizando el "." como delimitador.



5.2. Programar con JNDI

Para acceder al contexto inicial debemos utilizar un código similar al mostrado a continuación:

```
Context miContexto = null;
Hashtable ht = new Hashtable ();
ht.put(Context.INITIAL_CONTEXT_FACTORY,
        "weblogic.jndi.WLInitialContextFactory");
ht.put(Context.PROVIDER_URL,
        "t3://localhost:7001");
miContexto = new InitialContext (ht);
```

En la primera línea hemos creado un contexto. La tabla *Hash* creada sirve para pasar unos cuantos parámetros iniciales. El primer parámetro es quién nos va a proporcionar el árbol JNDI, en este caso Weblogic. El segundo parámetro es

la URL del servidor que nos proporcionará el árbol. Una vez asignados los parámetros iniciales creamos un contexto inicial. En todo código JNDI debemos capturar la excepción *NamingException*.

Cuando terminemos de utilizar el contexto debemos cerrarlo llamando al método **close** de Context.

Para asociar un objeto en el árbol utilizaremos el siguiente código:

```
Persona persona = new Persona();
miContexto.bind ("objeto persona", persona);
// miContexto.rebind ("objeto persona", persona);
```

Hemos creado un objeto cualquiera, en este caso el objeto *persona*. Utilizamos el contexto para asociar (*bind*) el nombre "*objeto persona*" al objeto. Si utilizamos el método *bind* y ya existe una asociación con este nombre en el árbol, se producirá una excepción. Por ello se puede utilizar la llamada al método *rebind* que, caso de existir, reemplaza la asociación anterior.

También podemos crear subcontextos para organizar mejor nuestra información. Para crear un subcontexto podemos utilizar el siguiente código:

```
Context subcontexto = miContexto.createSubContext
("empleados");
Persona persona = new Persona();
subcontexto.bind ("contable", persona);
```

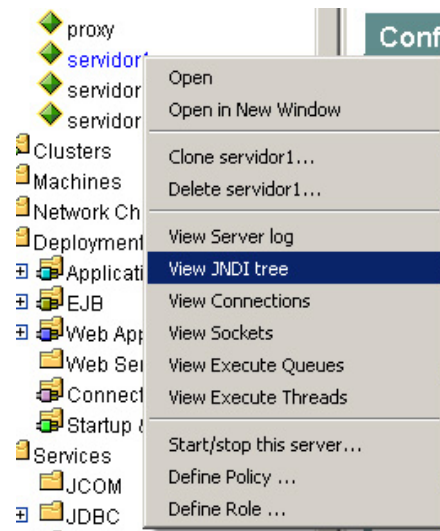
Hemos creado un subcontexto enlazado con el contexto inicial y dentro de ese subcontexto hemos asociado un objeto.

Por último, queda recuperar un objeto dentro de un contexto. El siguiente código devuelve el objeto introducido en el ejemplo anterior. Observad que es necesario realizar una conversión al objeto que esperamos que se devuelva.

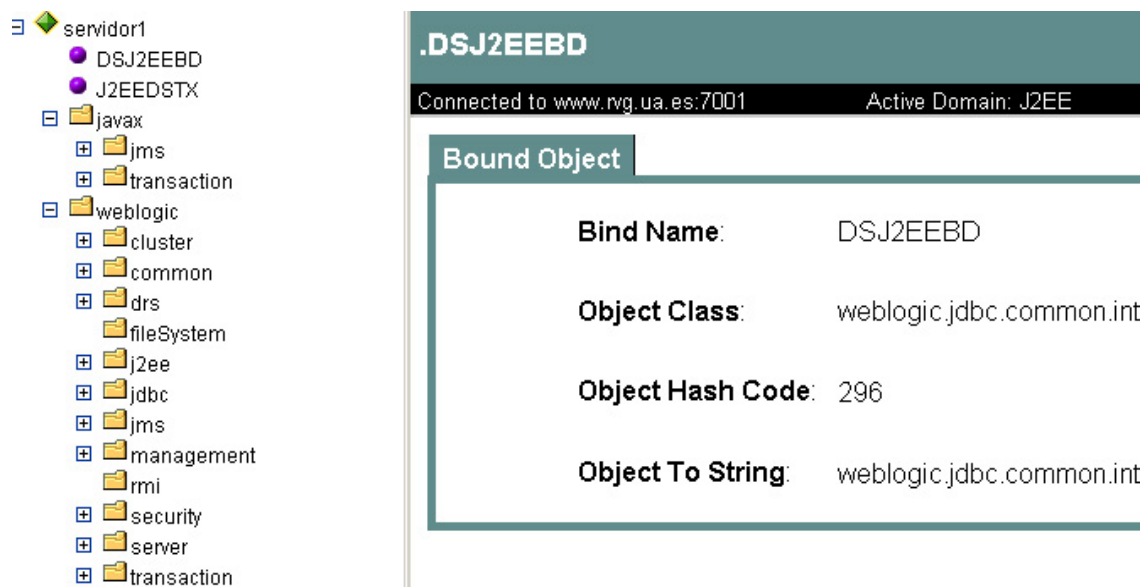
```
Persona pers = (Persona) miContexto.lookup
("empleados/contable");
```

5.3. Weblogic y JNDI

Weblogic gestiona un árbol JNDI propio, donde las aplicaciones pueden asociar objetos para que puedan ser utilizados por otras aplicaciones. Podemos ver el árbol JNDI de un determinado servidor. Para ello pinchamos con el botón derecho sobre el servidor seleccionado y pinchamos en la opción *View JNDI tree*.



Se nos abrirá una nueva ventana que nos mostrará el árbol JNDI. Esta ventana es muy similar a la consola, puesto que está dividida en dos y tiene un applet en la parte izquierda. El applet muestra el árbol y podemos movernos por él para ver los objetos presentes. En la figura se muestra dos objetos, DSJ2EEBD y J2EEDSTX, que son dos fuentes de datos a las que tendremos acceso mediante JNDI.



5.4. Clases de arranque y parada

Algunas veces se hace necesario el uso de clases para realizar ciertas tareas tanto en el arranque como en la parada de los servidores. Un posible ejemplo son las asociaciones que se han creado en el árbol JNDI dentro de WebLogic. Una vez que paremos los servidores estas asociaciones no persisten. Podemos hacer uso de clases que carguen estas asociaciones y que las almacenen cuando paremos los servidores. Para utilizar una clase de arranque o parada pinchamos en el icono *Startup & Shutdown* y creamos una clase

nueva. Debemos especificar el identificador de la clase (elegido de forma arbitraria) y el nombre de la clase. Tened en cuenta que la clase debe estar incluida en el classpath



MiDominio> Startup & Shutdown> Ejecutivos

Connected to miguel.dccia.ua.es:7001 Active Domain: MiDominio



Configuration



Targets



Notes



  **Name:**



Ejecutivos

  **ClassName:**

  **Deployment Order:**

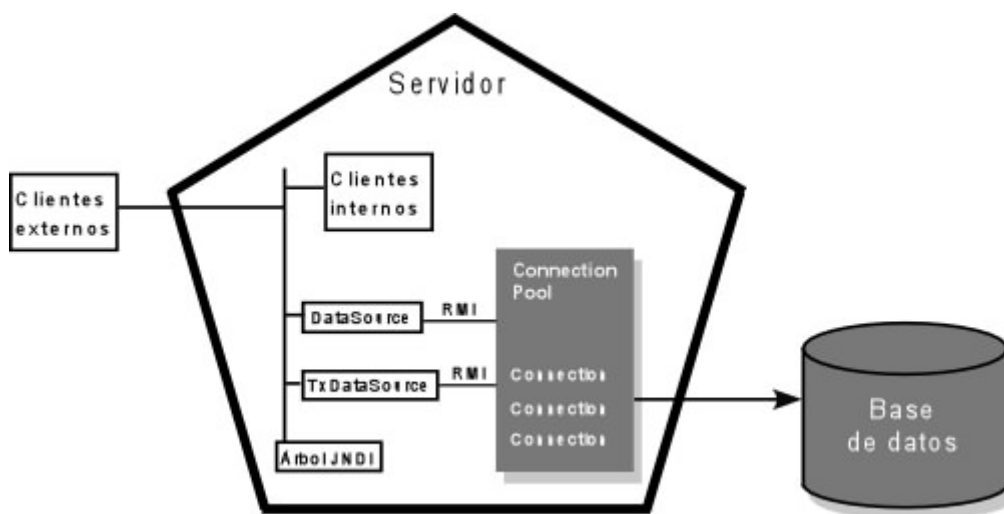
  **Arguments:**

  ☐ **Failure is fatal**

  ☐ **Run before application deployments**

Tema 6: Acceso a bases de datos con el servidor de aplicaciones

Hasta ahora la conexión a una base de datos se realizaba mediante un código JDBC que utilizaba un driver y obtenía una conexión. Una aplicación que utilice este código puede seguir haciéndolo dentro de WebLogic, es decir, si desplegamos esa aplicación en WebLogic la aplicación funcionará correctamente. En este tema vamos a configurar el servidor de aplicaciones para que nos permita trabajar con JDBC, aprovechando las características de pool de conexiones y fuentes de datos. También veremos cómo podemos conseguir una conexión controlada por el servidor. Un esquema general de cómo maneja WebLogic las bases de datos es el siguiente:



Tanto los clientes externos como los internos obtienen un objeto DataSource o TxDataSource mediante el árbol JNDI gestionado por el servidor. Una vez obtenido este objeto, se solicita una conexión a través de él, que a su vez la obtiene de un Connection Pool gestionado por el servidor.

6.1. Configuración de las fuentes de datos y el pool de conexiones

6.1.1 Pool de conexiones

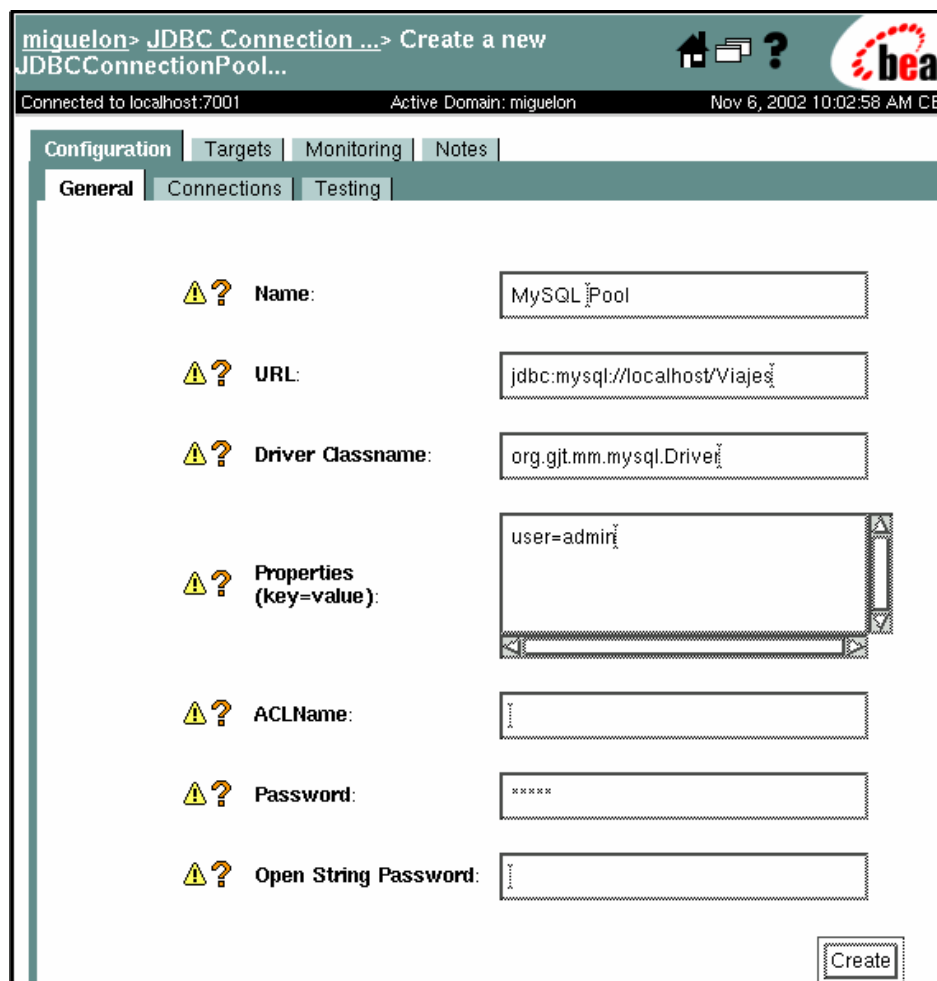
El primer paso a seguir es la configuración de un pool de conexiones. Para ello seleccionad la opción JDBC del menú *Services*. Nos aparecerá una pantalla como la mostrada a continuación. Pinchando en el enlace podemos configurar un nuevo pool de conexiones.



 [Configure a new JDBC Connection Pool...](#)

En la siguiente pantalla definimos el pool de conexiones. Indicamos los siguientes datos:

- El nombre del nuevo pool. Este nombre lo utilizaremos después para hacer referencia al pool de conexiones.
- La URL de la base de datos. Podemos incluir parámetros adicionales en la URL.
- El nombre de la clase que gestiona la conexión. Tened en cuenta que la clase debe estar disponible en el CLASSPATH.
- En el apartado de *Properties* incluimos las propiedades que queramos enviar en la conexión. En este caso hemos puesto el nombre del usuario para la conexión.
- Por último indicamos la contraseña y pulsamos *Create*.



miguelon> JDBC Connection ...> Create a new JDBCConnectionPool...

Connected to localhost:7001 Active Domain: miguelon Nov 6, 2002 10:02:58 AM CET

Configuration Targets Monitoring Notes

General Connections Testing

Name: MySQL Pool

URL: jdbc:mysql://localhost/Viajes

Driver Classname: org.gjt.mm.mysql.Driver

Properties (key=value): user=admin

ACLName:

Password: *****

Open String Password:

Create

Pasamos a la solapa de *Connections*. Repasamos el significado de cada valor:

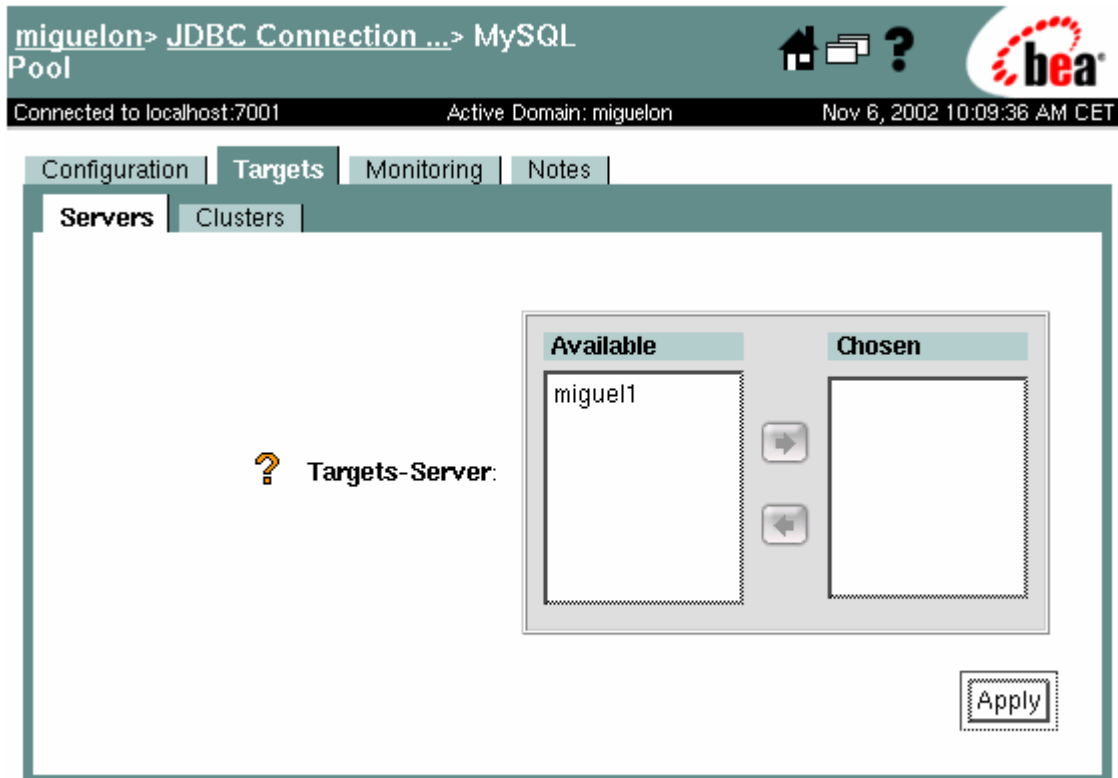
- La capacidad inicial indica el número de conexiones que se crearán en el instante inicial.
- La capacidad máxima es el número máximo de conexiones que se abrirán en este pool, independientemente de las solicitadas.
- El incremento de capacidad maneja el número de conexiones que se abrirán cuando llega una nueva solicitud de conexión y todas están ocupadas. Es recomendable crear varias conexiones a la vez, pues es un recurso crítico.
- El tiempo de retraso en el login indica el tiempo de espera entre la apertura de conexiones.

The screenshot shows the 'JDBC Connection ...> MySQL Pool' configuration window. The top bar indicates 'Connected to localhost:7001', 'Active Domain: miguelon', and the date 'Nov 6, 2002 10:09:19 AM CET'. The 'Configuration' tab is active, with sub-tabs for 'General', 'Connections', 'Monitoring', and 'Notes'. The 'Connections' sub-tab is selected, showing various settings with warning icons (yellow triangles with question marks) next to them:

- Initial Capacity:** 2
- Maximum Capacity:** 10
- Capacity Increment:** 2
- Login Delay Seconds:** 0 seconds
- Refresh Period:** 0 minutes
- ☐ **Supports Local Transaction**
- ☐ **Allow Shrinking**
- Shrink Period:** 15 minutes
- Prepared Statement Cache Size:** 0

La última opción a modificar (por el momento) está en la solapa de *Targets*. Con esta opción especificamos en cuál o cuáles de los servidores está activo este pool de conexiones. Podemos asociarlo a uno o varios servidores o bien a todo un cluster. Para ello pinchamos en la solapa correspondiente (*Servers* o *Cluster*) y pasamos la elección a *Chosen*. En nuestro caso sólo tenemos un servidor, por lo que lo seleccionamos, lo pasamos al cuadro de *Chosen* y pulsamos en *Apply*. Si se ha producido un error (no se encuentra la clase del

driver, la base de datos no está disponible, etc.) se mostrará un error en la línea de comandos por la que hemos arrancado el servidor. Una cuestión muy importante a tener en cuenta es que si un pool lo asociamos a un cluster, el número de conexiones se multiplica por el número de servidores que forman el cluster.



Si pulsamos de nuevo en *Services -> JDBC* aparecerá la siguiente información, indicando que el pool de conexiones está activo. Desde esta ventana podemos crear una copia del pool o eliminar el pool, pulsando los iconos de la derecha.

The screenshot shows the JDBCAccess console interface. At the top, the title bar reads 'miguelon> JDBC Connection ...'. Below the title bar, a status bar indicates 'Connected to localhost:7001', 'Active Domain: miguelon', and the date/time 'Nov 6, 2002 10:13:11 AM CET'. The main content area has tabs for 'Configuration', 'Targets', 'Monitoring', and 'Notes'. The 'Configuration' tab is selected, showing a table of JDBC connection pools. Below the table are links for 'Configure a new JDBC Connection Pool...' and 'Customize this view...'. The table has columns for Name, URL, Driver Classname, Initial Capacity, Capacity Increment, and Maximum Capacity. The 'MySQL Pool' is listed with a URL of 'jdbc:mysql://localhost/Viajes', driver 'org.gjt.mm.mysql.Driver', initial capacity of 2, capacity increment of 2, and maximum capacity of 10. Action icons (copy and delete) are shown for the pool.

Name	URL	Driver Classname	Initial Capacity	Capacity Increment	Maximum Capacity	
MySQL Pool	jdbc:mysql://localhost/Viajes	org.gjt.mm.mysql.Driver	2	2	10	

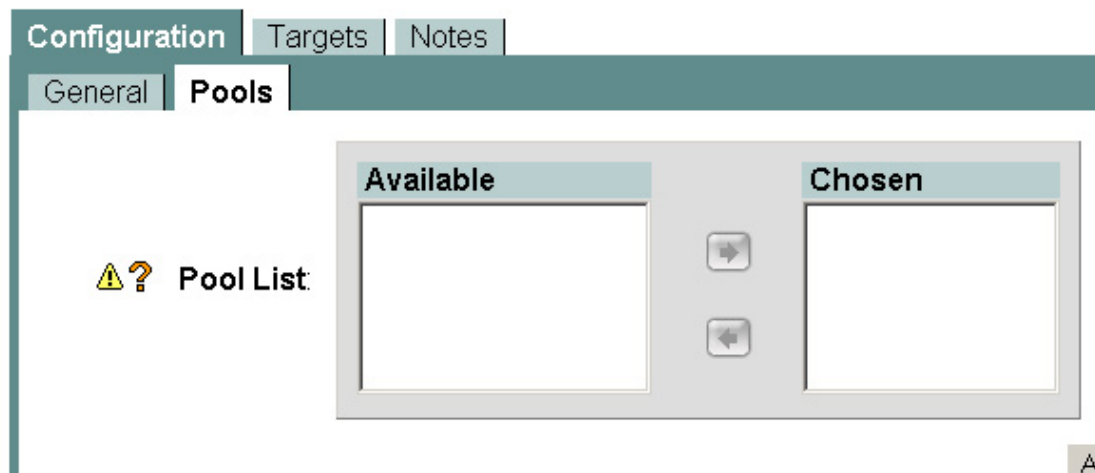
Dentro de WebLogic también podemos definir un Multipool. Un Multipool permite agrupar varios pool de conexiones para proporcionar alta disponibilidad (debido a posibles errores o fallos de las bases de datos) y balanceo de carga (para sistemas con una carga excesiva). Pinchamos en el icono *Multipools* y nos aparecerá la siguiente figura que nos permite configurar un nuevo Multipool.



En la siguiente pantalla damos el nombre al multipool y seleccionamos el tipo de algoritmo a aplicar: *High-Availability* (alta disponibilidad) o *Load-Balancing* (balanceo de carga).

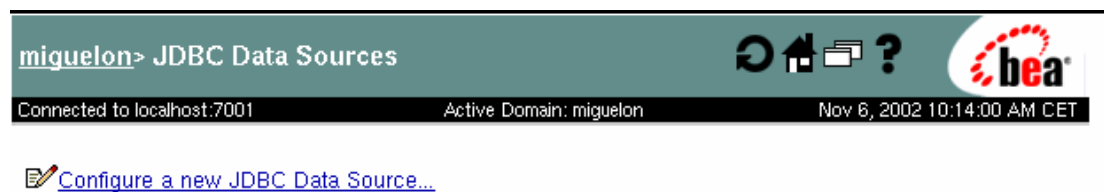


Por último tenemos que elegir los pools (ya creados, en este caso no tenemos ninguno) que formarán parte de este multipool. No tenemos que olvidarnos de seleccionar el servidor o cluster al cual asignaremos el multipool.



6.1.2 Fuentes de datos

Pasamos ahora a definir una fuente de datos. Pinchamos en *Services* -> *DataSources*. Pinchamos en configurar una nueva fuente de datos.



Damos nombre a esta nueva fuente de datos. Los siguientes datos son los siguientes:

- El nombre JNDI es el identificador que nos servirá para obtener el objeto DataSource usando JNDI.
- El nombre del pool es el nombre del pool asociado a este DataSource. Usamos el pool creado anteriormente.

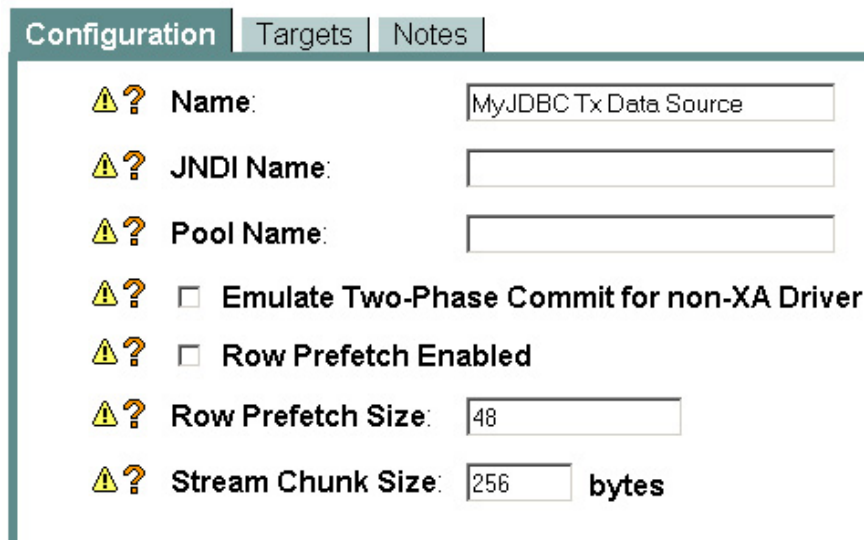
Pinchamos el botón *Apply* para crear la fuente de datos.

The screenshot shows the configuration page for a MySQL DataSource in the JBoss/JRun console. The breadcrumb path is 'miguelon > JDBC Data Sources > MySQL DataSource'. The console status bar indicates 'Connected to localhost:7001', 'Active Domain: miguelon', and the date/time 'Nov 6, 2002 10:14:33 AM CET'. The 'Configuration' tab is selected, showing a list of properties with warning icons (yellow triangle with a question mark). The properties are: Name (MySQL DataSource), JNDI Name (MySQLDataSource), Pool Name (MySQL Pool), Row Prefetch Enabled (unchecked checkbox), Row Prefetch Size (48), and Stream Chunk Size (256 bytes). An 'Apply' button is located at the bottom right of the configuration area.

Configuration	Targets	Notes
<p> Name: MySQL DataSource</p> <p> JNDI Name: <input type="text" value="MySQLDataSource"/></p> <p> Pool Name: <input type="text" value="MySQL Pool"/></p> <p> <input type="checkbox"/> Row Prefetch Enabled</p> <p> Row Prefetch Size: <input type="text" value="48"/></p> <p> Stream Chunk Size: <input type="text" value="256"/> bytes</p> <p><input type="button" value="Apply"/></p>		

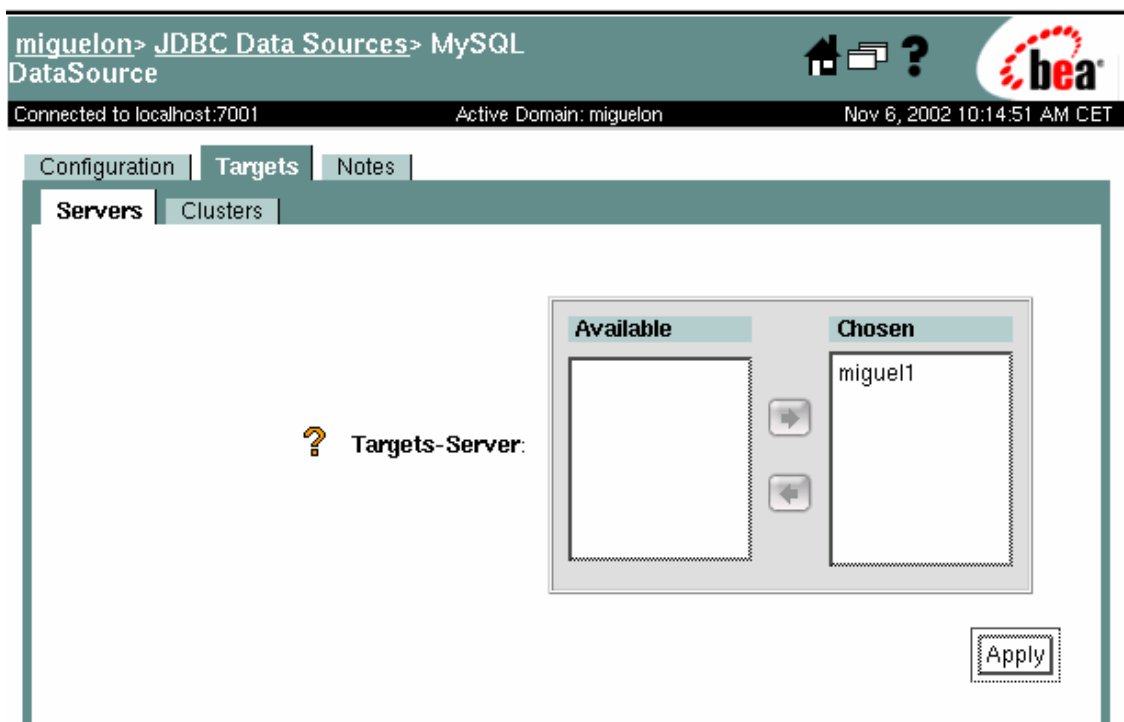
En el caso de las fuentes de datos transaccionales (Tx Data Source), la única diferencia con las fuentes de datos normales es un parámetro: *Emulate Two-Phase Commit* ... Si marcamos esta opción permitirá emular el método *Two-Phase commit* con drivers que no permitan transacciones distribuidas. Se recomienda que se utilice este tipo de fuentes de datos cuando se cumpla algunos de los siguientes criterios:

- Hagamos uso de transacciones distribuidas.
- Utilicemos el contenedor EJB para realizar transacciones.
- Se incluya más de una base de datos en una transacción.
- Se acceda a más de un recurso (una base de datos y un elemento JMS) dentro de la misma transacción.



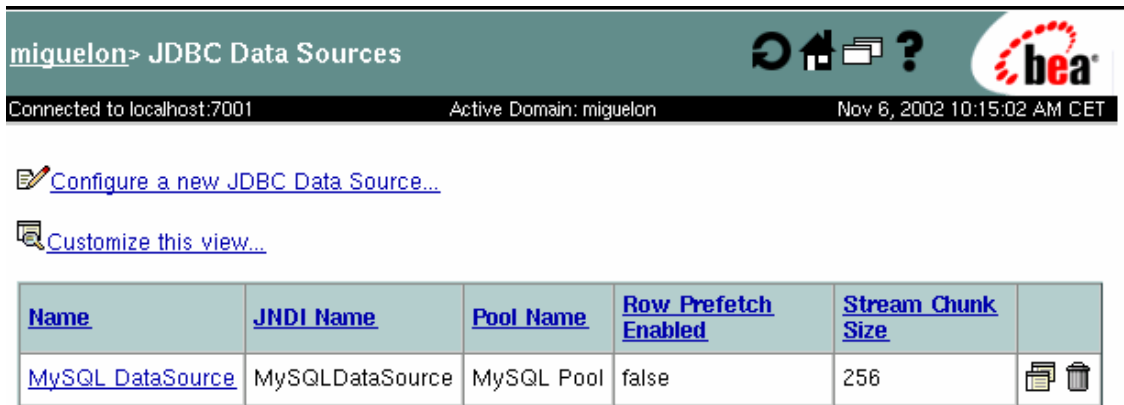
The image shows a configuration window for a JDBC Data Source. It has three tabs: 'Configuration', 'Targets', and 'Notes'. The 'Configuration' tab is active. It contains several fields and checkboxes, each preceded by a yellow warning icon with a question mark. The fields are: 'Name' (MyJDBC Tx Data Source), 'JNDI Name' (empty), 'Pool Name' (empty), 'Row Prefetch Size' (48), and 'Stream Chunk Size' (256 bytes). There are two checkboxes: 'Emulate Two-Phase Commit for non-XA Driver' and 'Row Prefetch Enabled', both of which are unchecked.

Al igual que con el pool de conexiones, debemos indicar el servidor al que está asociado esta fuente de datos. Pinchamos en *Apply* para confirmar los datos. Si se produjo algún error aparecerá en la línea de comandos del servidor. Lo mismo que con el pool de conexiones, si asociamos una fuente de datos a un cluster, el número de conexiones se multiplica por el número de servidores que forman el cluster.



The image shows the same configuration window, but with the 'Targets' tab selected. The 'Configuration' tab is still visible at the top. The 'Targets' tab has two sub-tabs: 'Servers' and 'Clusters'. The 'Servers' sub-tab is active. It shows a 'Targets-Server' section with a yellow warning icon and a question mark. This section contains two lists: 'Available' (empty) and 'Chosen' (containing 'miguel1'). There are two arrows between the lists, one pointing right and one pointing left. An 'Apply' button is located at the bottom right of the 'Targets-Server' section. The top of the window shows the user 'miguelon' and the domain 'miguelon'.

Podemos consultar, copiar o eliminar la fuente de datos creada pinchando de nuevo en *Services -> DataSources*.



miguelon> JDBC Data Sources

Connected to localhost:7001 Active Domain: miguelon Nov 6, 2002 10:15:02 AM CET

[Configure a new JDBC Data Source...](#)

[Customize this view...](#)

Name	JNDI Name	Pool Name	Row Prefetch Enabled	Stream Chunk Size
MySQL DataSource	MySQLDataSource	MySQL Pool	false	256

Una vez realizados todos estos pasos debemos parar el servidor y ponerlo en funcionamiento de nuevo para que surtan efecto los cambios.

6.2. Uso de una fuente de datos en una aplicación

Para hacer uso de una fuente de datos definida por el servidor de aplicaciones debemos obtener una conexión haciendo uso de un objeto **DataSource**. Este objeto se obtendrá haciendo uso de JNDI. Dentro de un programa JDBC los pasos a seguir para obtener una conexión son los siguientes:

- Debemos importar las clases para el manejo de las fuentes de datos y JNDI.

```
import javax.sql.DataSource;
import javax.naming.*;
import java.util.Hashtable;
```

- Definimos un contexto inicial (JNDI). La tabla *Hash* sirve para definir algunas variables de JNDI. En este caso le decimos que el contexto inicial se debe obtener de WebLogic y que la URL se obtiene del servidor de aplicaciones. En esta URL debemos especificar la dirección del servidor que tenga asignado el recurso.

```
Context miContexto = null;
Hashtable ht = new Hashtable ();
ht.put(Context.INITIAL_CONTEXT_FACTORY,
        "weblogic.jndi.WLInitialContextFactory");
ht.put(Context.PROVIDER_URL,
        "t3://localhost:7001");
miContexto = new InitialContext (ht);
```

- Obtenemos la fuente de datos buscando por el nombre lógico asociado al pool de conexiones antes creado. Recordad que es el nombre lógico para JNDI, no el nombre de la fuente de datos.


```
DataSource ds = (DataSource) miContexto.lookup  
("MySQLDataSource");
```

- El último paso es obtener una conexión a la base de datos a partir de la fuente de datos.

```
Connection con = ds.getConnection ();
```

Para poder ejecutar debemos tener en el CLASSPATH el siguiente fichero:

\$HOME_BEA/weblogic700/server/lib/weblogic.jar

donde \$HOME_BEA es el directorio donde está instalado Bea. También, por supuesto, debe estar funcionando el servidor de aplicaciones y el de la base de datos.

Instalación de WebLogic 7.0

En este primer ejercicio vamos a instalar el servidor de aplicaciones WebLogic 7.0. Para instalar WebLogic ir a vuestro CD y en el directorio *Servidores -- Recursos* tenéis un fichero ejecutable (*.bin*). Ejecutadlo y seguid las instrucciones vistas en teoría. Como podéis comprobar ya existe un directorio de instalación creado en vuestro usuario (el directorio *bea*). Elegid como directorio de instalación *bea1*. Una vez instalado comprobad la estructura de directorios creada y editar el fichero *license.bea* y echadle un vistazo.

Una vez instalado borrad el directorio *bea1* recién creado. Utilizaremos el directorio *bea*. Vais a crear el primer dominio. Utilizad la utilidad *dmwiz.sh*. Creáis un dominio *WLS Domain* y lo llamáis *MiDominio*. Elegid la opción de servidor de administración con servidores administrados. Los datos de los servidores serán los siguientes:

Nombre servidor	Puerto de escucha
Admin	7001, 7002
serva	6001, 6002

La dirección de escucha es la de vuestro ordenador. Una vez creado el directorio echad un vistazo a la estructura de directorios creada. Arrancad el servidor de administración suministrando el usuario y la contraseña. Una vez arrancado el servidor de administración proceded con el servidor *serva*. Abrid la consola de administración y visitad todas las opciones vistas en clase de teoría.

Definición de dominio, servidores y Node Manager

En este ejercicio vamos a definir un nuevo dominio. El sistema a crear tiene la siguiente estructura:

Dominio: **Desarrollo**

Nombre del servidor	Dirección de escucha	Puerto de escucha
admin	La de vuestra máquina	7001, 7002
serva		5001, 5002
servb		4001, 4002

Los pasos a seguir son los siguientes:

- Proceded a crear primero el dominio con únicamente el servidor de administración utilizando la utilidad *dmwiz.sh*.
- Posteriormente utilizad la consola de administración para crear los otros dos servidores.
- Asignar valor a las variables dentro del ejecutable que arranca el servidor de administración y los administrados. Cread un ejecutable por cada servidor administrado.
- Cread una máquina dentro del dominio. Será una máquina de tipo Unix y podéis asignar el nombre *maquina1*.
- Configurar el NodeManager para poder arrancar los servidores desde la consola de administración. El NodeManager controlará los servidores *serva* y *servb*. Sabremos que está funcionando pinchando en la máquina correspondiente y en *Monitoring*. Parad los servidores administrados. Ahora, desde la consola de administración, pinchad en cada uno de los servidores e id a *Control->Start/Stop* y arrancad los dos servidores.
- Habilitar la conexión segura (el puerto SSL) en cada servidor. Entrad a la consola de administración pero utilizando el protocolo seguro: ***https://direccion:puerto_seguro/console***
- Proteger todos los servidores de posibles ataques DoS. Indicad un tamaño de paquete máximo de 512Kb y un tiempo de espera máximo de 60 segundos.

Tareas adicionales a realizar:

- Arrancad todo el sistema, primero el servidor de administración y después los administrados.
- Visualizad la cantidad de memoria que está usando cada servidor. Para ello primero pincháis en el servidor correspondiente y accedéis a *Monitoring->Performance*. Forzar el recolector de basura y observad qué pasa con la memoria.

- Id a *Logging->General* del servidor *serva*. Indicad que el umbral de seguridad (*Stdout severity Threshold*) sea el de *Info*. Pasad a la solapa *Rotation* y configurar las opciones para que el tipo de rotación sea por tamaño e indicad un tamaño mínimo de 100k. Parar el servidor *serva* en *Control->Start/Stop*. Una vez que nos aparezca un mensaje indicando que se ha parado volver a arrancarlo desde línea de comandos. Editar el fichero log para el servidor y comparar su contenido con el del otro servidor, *servb*.

Despliegue de aplicaciones, creación de usuarios y manejo de seguridad y utilidades desde línea de comandos

Despliegue de aplicaciones

Como primer ejercicio vais a desplegar la aplicación [benefits.war](#). Realizad el despliegue en el servidor *serva*. Probad que el servidor responde a la aplicación (podéis llamar a la aplicación con */benefits*). Echad un vistazo a la aplicación y eliminarla (*undeploy*). Volved a probar que el servidor ha dejado de responder a la aplicación.

Manejo de seguridad básica

Lo primero a realizar es la creación de dos grupos de usuarios que contendrán los usuarios descritos a continuación. Para simplificar podéis asignar la misma contraseña a todos los usuarios.

Nombre de usuario	Grupo
juan	faqs
miguel	faqs
antonio	faqs
otto	managers
francisco	managers
javier	managers
patricia	managers

Una vez creados todos los usuarios eliminad el usuario *otto*.

Ahora vais a desplegar una nueva aplicación. La aplicación se encuentra en el fichero [timeoff.zip](#). Descomprimidlo y cambiad los datos necesarios en los ficheros *web.xml* y *weblogic.xml* dentro del directorio WEB-INF. La aplicación se llama con el nombre *timeoff*. Las URL a proteger son */managers/** y */officeclosing/** que son los servlets a los que sólo tienen acceso los miembros del grupo *managers*. Debéis asignar un nombre de rol (el que queráis). En el fichero *weblogic.xml* el rol definido antes se asigna a un grupo o usuario (es preferible siempre asignarlo a un grupo). En nuestro caso lo hacéis para el grupo *managers*.

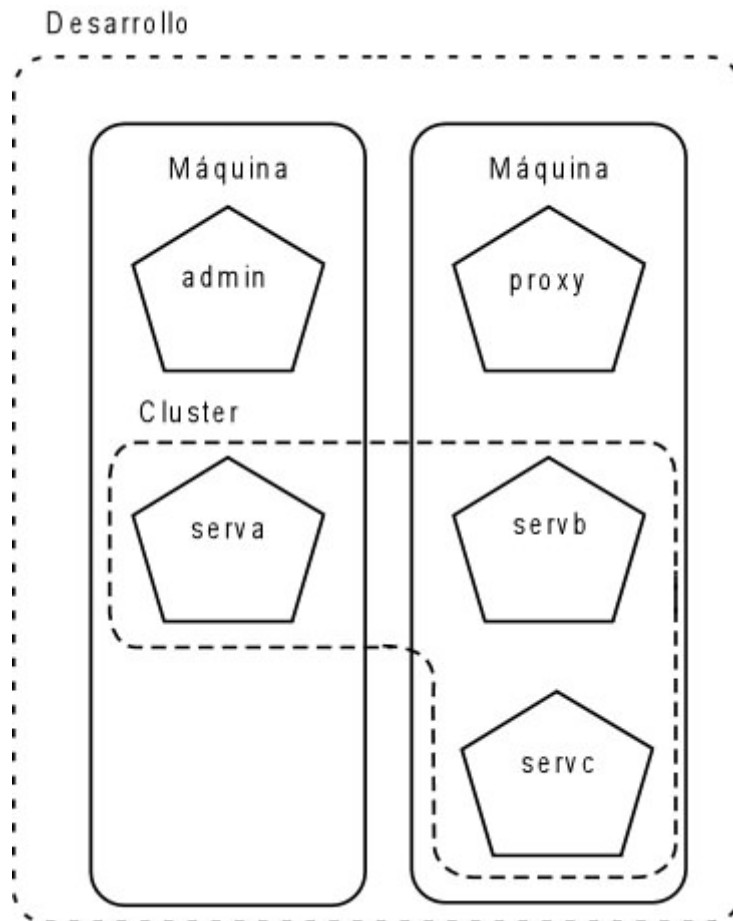
Opcionalmente, desplegad la aplicación *faqs* que hicisteis en el módulo de servlets y configurad WebLogic para que funcione con esta aplicación. Debéis tener en cuenta que el classpath de WebLogic debe contener el driver para la conexión a la base de datos.

Utilidades desde línea de comandos

Utilizad todas llamadas desde línea de comandos para familiarizaros con ellas: parad un servidor y volverlo a arrancar, eliminar una aplicación y volverla a desplegar y probad con todas las otras opciones.

Creación de un cluster

Este ejercicio será el más completo de los realizados hasta el momento. Vais a trabajar por parejas, utilizando dos máquinas para definir un sistema lo más real posible. El sistema a crear tendrá la siguiente estructura:



La máquina donde se defina el dominio será la que contenga el servidor de administración y un servidor adicional. La otra máquina contendrá el servidor proxy y dos servidores más. El servidor de administración y el proxy estarán fuera del cluster. Tened en cuenta que sólo se define un dominio, en la máquina donde tengamos el servidor de administración. Dentro de ese dominio se definen todos los servidores. Los servidores a definir son los siguientes:

Dominio: **Desarrollo**

Nombre del servidor	Dirección de escucha	Puerto de escucha	Nombre DNS	Grupo primario	Grupo secundario
admin		7001	Compañero1		
serva		4001	Compañero1	grupo1	grupo2

proxy		7001	Compañero2		
servb		4001	Compañero2	grupo2	grupo1
servc		5001	Compañero2	grupo2	grupo1

Configurad el NodeManager en las dos máquinas para poder arrancar los servidores desde la consola de administración. Definid la aplicación proxyApp tal como se explica en teoría, desplegadla y asignarla por defecto al servidor proxy. Definid la replicación de memoria, con los grupos definidos en la tabla anterior.

Para probar la tolerancia a fallos vais a utilizar la siguiente aplicación [shoppingcart.war](http://miguel.dccia.ua.es:4001/shoppingcart). Esta aplicación mantiene un carrito de la compra en memoria. Desplegad la aplicación en el cluster. Ahora probad el funcionamiento de la aplicación. Para ello dejad únicamente un servidor en el cluster y el servidor proxy. Indicad la dirección del servidor proxy y llamáis a la aplicación *shoppingcart* (por ejemplo, <http://miguel.dccia.ua.es:4001/shoppingcart>). Almacenad varios elementos en el carrito de la compra. Llamad ahora a la dirección del servidor que está funcionando en el cluster y comprobad que el carrito de la compra es el mismo. Arrancad otro de los servidores del cluster e introducid algún elemento más en el carro. Por último parar el primer servidor y comprobad que el carro sigue siendo el mismo en el último servidor arrancado.

Manejo de clases de arranque y JNDI

En este ejercicio vamos a crear una clase de arranque que cargue varios objetos y los enlace en el árbol JNDI del servidor. El objetivo de este ejercicio es familiarizaros con la configuración de una clase de arranque y visualizar el árbol JNDI. Debéis descargaros en vuestra máquina el fichero [arranque.zip](#). Descomprimidlo en el directorio de vuestro dominio. El directorio consta de dos subdirectorios. El primero, *objetos*, define las clases de los objetos a enlazar. El segundo, *startup*, contiene la clase que realizará el enlace. Para que se ejecute esta clase debéis crear una nueva clase de arranque con los siguientes parámetros:

Parámetro	Valor
Nombre	Ejecutivos
Nombre de la clase	arranque.startup.bindObjects
Target	adminServer

Una vez configurado parad el servidor de administración. Antes de volverlo a arrancar debéis asegurarnos que el directorio donde habéis dejado las clases está dentro del classpath. Para ello podéis editar el fichero:

```
$BEA_HOME\weblogic700\server\bin\startWLS.sh
```

e incluir el path hasta las clases. Hecho esto arrancad de nuevo el servidor de administración. Os debe aparecer el mensaje *Nombre de la clase de arranque: bindObjects*. Esto indicará que se ha realizado correctamente la carga de la clase.

Una vez arrancado el servidor, visualizar su árbol JNDI y comprobad que se han realizado los enlaces con los objetos. Los objetos enlazados son:

Contexto	Nombre del objeto enlazado
com.bea.ejecutivos	Jefe
com.bea.ejecutivos	Presidente
com.bea.ejecutivos	Propietario

Acceso a bases de datos con el servidor de aplicaciones

En este ejercicio se trata de configurar un pool de conexiones y una fuente de datos para después utilizarlos en una aplicación web. Utilizad la aplicación *faqs.war* que accede a una base de datos (acordaros de los ejercicios de JSP). Primero probad a desplegar la aplicación y comprobad que el servidor responde a la aplicación (acordaros de incluir el driver en el classpath de weblogic).

Vamos a proceder a crear el pool de conexiones. Debéis crear un nuevo pool de conexiones para acceder a MySQL, a la base de datos *faqs*. El pool se llamará *MysqlPool* y debéis asignar los parámetros URL; nombre de la clase; en *Properties* indicáis *user=root*; asignáis la contraseña de la base de datos. Creáis el pool y pasáis a la solapa *Connections*. Aquí indicáis una capacidad inicial de 5, una máxima de 10 y un incremento de 2. Por último, el destino del pool será uno de los servidores del dominio.

A continuación cread la fuente de datos. El nombre de la fuente de datos será *MysqlDS*, el nombre JNDI *faqsBD* y hacéis referencia al pool antes creado. Como destino seleccionáis el mismo servidor que para el pool.

Una vez realizados estos pasos tenéis que modificar el código de la aplicación *faqs* para que utilice la fuente de datos definida previamente. Sólo tenéis que modificar un fichero, *FAQBD.java* localizado en el directorio *WEB-INF/classes/faqs/bd*. La conexión en ese fichero se obtiene a partir de una conexión con el driver de mysql. Ahora debéis utilizar JNDI para obtener una referencia a la fuente de datos creada y que sea ésta la que nos devuelva la conexión.