Introducción a WebLogic Server

Índice

Hasta ahora hemos repasado las tecnologías más importantes de Java Enterprise. Para ello nos hemos apoyado en GlassFish un servidor de aplicaciones ligero, versátil y disponible tanto como aplicación de código abierto y de uso gratuito, como producto comercial soportado por Oracle.

Esta es una apuesta muy válida para una pyme o para aplicaciones cuyo funcionamiento no sea crítico, crítico en el sentido de pérdidas económicas millonarias en caso de fallo grave. Por este motivo las grandes empresas necesitan ir un paso más allá: se necesita una solución que pueda crecer a la par que el negocio, una arquitectura capaz de integrar las distintas tecnologías con las que se trabaje en la empresa (mainframes, bases de datos, pasarelas de pago, etc.) y que permita una monitorización detallada de su desempeño.

Junto con la licencia de uso se demanda del fabricante un soporte técnico resolutivo (en forma y plazos), independientemente de que la empresa pueda contar con personal técnico cualificado. En definitiva, siempre tiene que haber alguien a quien acudir en caso de problemas.

En las siguientes sesiones estudiaremos uno de los servidores comerciales más populares: WebLogic Server.

Características generales:

- Los requisitos de hardware son elevados comparados con un servidor ligero tipo GlassFish o JBoss (7 en adelante). Un puesto de desarrollo típico (IDE más servidor WebLogic en local) necesita un mínimo de 2GB de RAM
- Certificado para múltiples plataformas: HP-UX, AIX, Linux, Solaris y Windows
- Certificado Java EE 5 hasta la versión 11g, y en su última versión, WebLogic 12c (c de Cloud), consigue la certificación Java EE 6 y compatibilidad con Java SE 7, sin embargo, con apenas un año vida, no goza de una gran difusión. La release actual de la familia 11g es la 10.3.6 que adelanta alguna de las especificaciones de Java EE 6 como JPA 2.0 (Persistencia) JAX-RS (Servicios RESTful) y JSF 2.0.
- Se distribuye junto con su propia JVM, denominada JRockit. Cuando se configura un nuevo dominio en WebLogic se nos da la opción de escoger entre el JDK de Oracle u otro existente en el sistema operativo. JRockit es un producto que no se ofrece para todas las plataformas que pueden trabajar con WebLogic. Como ejemplo en el caso de AIX sólo podremos utilizar la JVM de IBM.
- Salvando las distancias, es una plataforma muy parecida a GlassFish, así que con la experiencia que hemos adquirido durante el curso no nos deberíamos sentir "extraños" administrando este servidor. De hecho Oracle se está esforzando en alinear ambos productos y no es descabellado que GlassFish acabe siendo una versión "descafeinada" de WebLogic.

1. Instalación de WebLogic

La instalación de WebLogic en un puesto de desarrollo es un proceso relativamente

sencillo, basta con descargar un paquete autoejecutable de la web de Oracle (hace falta registrarse como usuario previamente).

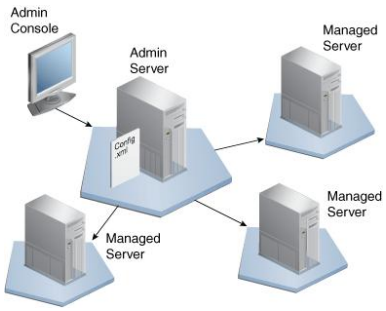
Este instalador nos preguntará dónde queremos ubicar la instalación de WebLogic y qué opciones queremos instalar (debidamente explicadas). Una vez contestadas todas las preguntas, comenzará la instalación de la aplicación.

En una máquina de producción, el proceso se complica, no tanto por los pasos, que son los mismos, sino por la cantidad de factores que hay que tener en cuenta para configurar un dominio: recursos disponibles, ancho de banda de los discos, configuración de red y un largo etcétera

Una vez instalado el producto, de forma automática se enlazará con el asistente de configuración, pero claro, antes de pasar a ese punto debemos estudiar unos cuantos conceptos...

2. Estructura de un dominio WebLogic

Todos los servidores de aplicaciones provienen de los mismos estándares, por tanto los conceptos con los que trabajaremos serán muy parecidos a los vistos en las sesiones de GlassFish. Por ejemplo:



Un dominio de WebLogic está formado habitualmente por un servidor de administración "admin", que hace las veces del DAS de GlassFish: coordina el trabajo de otros servidores, denominados servidores gestionados (managed servers) que son los que verdaderamente alojarán nuestras aplicaciones empresariales. Toda la configuración del dominio se puede mantener desde una consola de administración, que interactúa con el admin, al igual que en GlassFish.

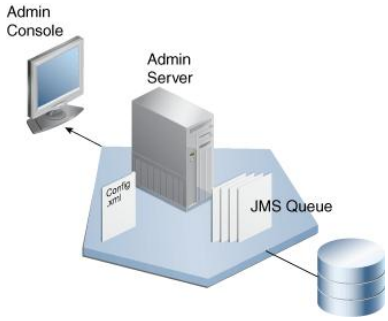
En este supuesto se trabaja con varias máquinas, pero podemos encontrarnos con una sola máquina que ejecute varios servidores simultáneamente.

Máquinas y Servidores

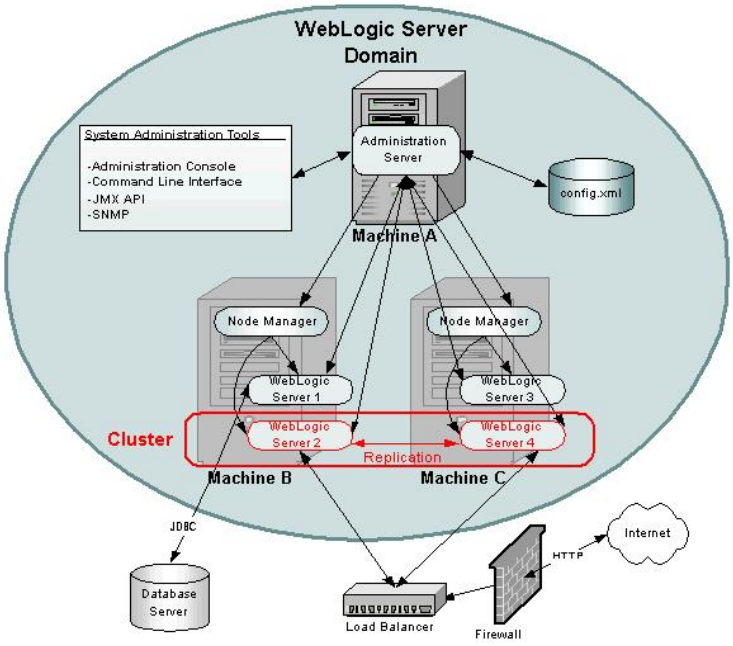
Para evitar confusiones utilizaremos la denominación *máquina* para referirnos al hardware sobre el cual se ejecute el servidor de aplicaciones y que no se confunda con el concepto *Servidor* de Java EE.

Aunque el servidor admin es un servidor convencional con funciones administrativas, no se suele utilizar para desplegar aplicaciones, salvo la propia consola de administración. Una excepción podría ser un dominio de pruebas en un puesto de desarrollo.

En este supuesto, la configuración "mínima" necesaria para desplegar aplicaciones en local sería la siguiente:



Si os fijáis, esta es la configuración que establece GlassFish por defecto. Teniendo claros los esquemas anteriores, vamos a entrar un poco más en detalle...



Esta es una representación mucho más realista de un dominio de WebLogic y de ella podemos observar los siguientes elementos:

- Para empezar, el dominio está compuesto por tres máquinas distintas: una máquina que albergará el servidor Admin y dos máquinas presumiblemente más potentes que albergarán 4 servidores gestionados, dos en cada máquina.
- Toda la configuración del dominio se centraliza en un archivo principal, denominado **config.xml** que es custodiado por el admin server. Aunque se podrían editar ciertas cosas manualmente está totalmente desaconsejado ya que podemos quedarnos sin dominio...
- En las máquinas B y C hay un proceso nuevo, denominado Node Manager que es el encargado de controlar los servidores que se ejecutan en cada máquina.
- Los servidores 1 y 3 se ejecutan de forma independiente, sin embargo los servidores 2 y 4 forman un clúster activo-activo.
- Las peticiones HTTP que recibe un clúster se pueden repartir de varias maneras entre los nodos que lo conforman, aunque lo habitual es utilizar hardware dedicado (balanceador de carga). Estos dispositivos monitorizan constantemente la salud de los servidores e implementan múltiples algoritmos de reparto de peticiones.

A continuación revisaremos con más detalle cada uno de estos elementos:

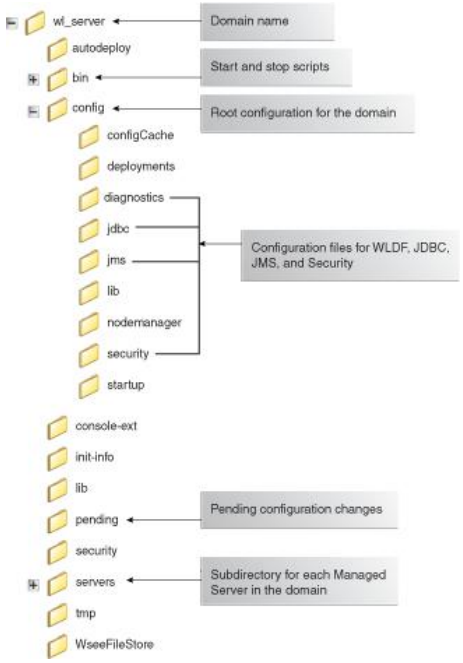
2.1. Dominio

En GlassFish veíamos que los dominios se creaban y se iniciaban con la herramienta asadmin, y todo su contenido se almacenaba dentro de la carpeta domains de la instalación de GlassFish.

En el caso de WebLogic, los dominios se definen como parte de la instalación del producto o simplemente ejecutando un asistente de configuración (bien de forma gráfica, o bien "espartana" desde un sh).

Una vez creado, este dominio se puede ubicar físicamente en la carpeta de la máquina donde más nos interese, y es el asistente el que se encarga de configurar los scripts apropiados para que se pueda arrancar sin problemas.

La estructura de directorios de un dominio es la siguiente:



Veréis que tiene cierto parecido con lo visto en GlassFish:

- Tenemos una carpeta autodeploy, para instalar aplicaciones con sólo copiar los empaquetados.
- Una carpeta **bin** con los scripts de arranque/parada del dominio, y de los servidores gestionados : `startWebLogic.sh`, `stopWebLogic.sh`, `startManagedWebLogic.sh`, `ystopManagedWebLogic.sh`
- La carpeta **config** contendrá el config.xml más una serie de carpetas adicionales con definiciones de recursos.
- La carpeta **lib** nos permitirá, como en GlassFish, añadir librerías que de forma

automática se cargarán en el CLASSPATH.
- La carpeta **pending**, almacenará los cambios pendientes de aplicar realizados a través de la consola de administración. Más adelante veremos esto con más detalle.
- Por último, la carpeta **servers** contendrá una carpeta por cada servidor definido en el dominio. En ellas, cada servidor almacenará su información de trabajo (archivos temporales y logs principalmente).

jugar con librerías

No podemos quitar/poner librerías en la carpeta `/lib` "en caliente". El procedimiento es parar el dominio, hacer los cambios y volver a iniciarlo.

Los dominios en WebLogic tienen dos modos de trabajo diferenciados: el **modo desarrollo** y el **modo producción**. En función de la configuración del dominio, las diferencias más importantes son las siguientes:

Modo Desarrollo

- La JVM por defecto es la HotSpot, del JDK estándar.
- Se permite utilizar certificados del almacén Demo.
- El despliegue automático de aplicaciones (autodeploy) está habilitado.
- El servidor admin, utiliza una configuración de arranque generada de forma automática.
- Se activan opciones para permitir la depuración de código.
- Los cambios en la configuración desde la consola se aplican de forma inmediata.

Modo Producción

- La JVM por defecto es JRockit.
- Se produce un warning si una aplicación quiere utilizar los certificados de ejemplo.
- Se desactiva el autodeploy, únicamente se pueden desplegar aplicaciones mediante las herramientas de administración.
- Se solicitan credenciales de usuario para lanzar los servidores (admin y gestionados).
- Se desactivan las opciones de debug.
- La consola de administración trabaja mediante bloqueos: Un administrador que quiera hacer cambios, primero debe bloquear de forma exclusiva la consola y luego aplicarlos. Se pide confirmación ante cualquier cambio. El config.xml con los cambios pendientes se almacena en la carpeta pending.

Los dominios están pensados para migrar de desarrollo a producción (esto se puede hacer fácilmente desde la consola), pero al revés es más complicado. Hay que modificar el fichero `setDomainEnv.sh` de la carpeta bin para añadir el parámetro `-DWebLogic.ProductionModeEnabled=false`

Sólo acudiremos a este archivo para ajustar ciertos parámetros del arranque del dominio en WebLogic (memoria, JVM, etc.), siendo recomendable utilizar la consola en la medida de lo posible.

2.2. Servidor de administración (Admin Server)

Como hemos comentado el Admin es un proceso Java cuya finalidad es controlar el resto de servidores pertenecientes al dominio, por una parte recibiendo instrucciones del administrador a través de la consola de administración y por otra se comunicándose con los servidores gestionados replicando la configuración del dominio y enviando instrucciones de control.

Para iniciar el servidor admin, tenemos el script `startWebLogic.sh` dentro de la carpeta bin. En la configuración por defecto, iniciar un dominio implica iniciar el servidor admin y posteriormente iniciar el resto de servidores gestionados.

Si por cualquier motivo se cae el servidor Admin, o hay que reiniciarlo para hacer algún cambio, los servidores gestionados podrán seguir funcionando sin problemas, y recuperarán la conexión con el admin en cuanto vuelva a estar disponible.

Por defecto el admin está configurado para escuchar peticiones a través del puerto 7001 mediante **t3/t3s** (basados en RMI) o **http/https**, que son los protocolos de comunicaciones que entiende WebLogic.

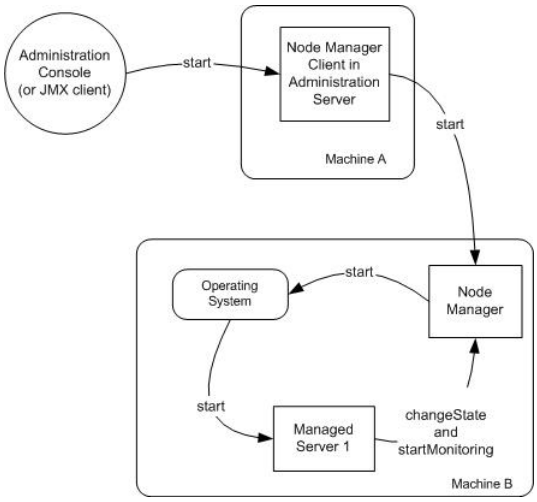
Más adelante comentaremos la consola de administración y allí veremos las posibilidades que tenemos de configuración y administración.

2.3. NodeManager

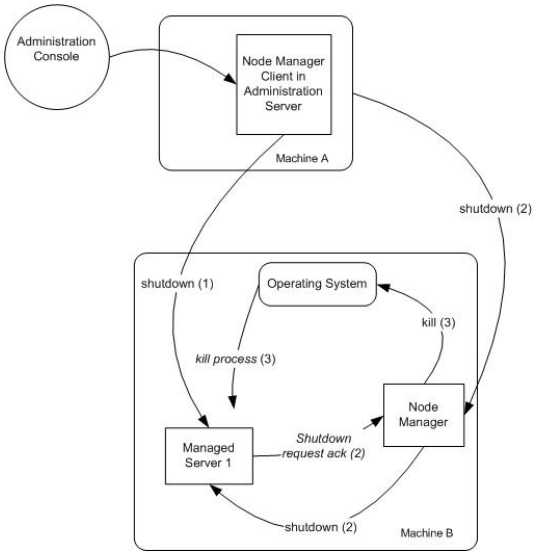
Como hemos visto en el diagrama, podemos tener todo nuestro dominio en una sola máquina, o repartido entre varios ordenadores. El proceso NodeManager es un "demonio" que se ejecuta en cada máquina donde residan servidores gestionados. Sus funciones son las siguientes:

- Parar y arrancar servidores gestionados remotos.
- Monitorizar el estado de los servidores gestionados.
- Matar y reiniciar de forma automática aquellos servidores que se encuentren en estado fallido.

Por ejemplo, si solicitamos iniciar un servidor desde la consola de administración, el Admin intentará conectarse con el NodeManager de la máquina donde se ejecute el servidor. Si se logra comunicar con él, se iniciará el servidor gestionado seleccionado. Si no, la consola mostrará un error informando del problema:



En el caso de que queramos detener un servidor, el Admin intentará conectarse directamente al servidor gestionado para detenerlo. Si esto no es posible, pasamos al plan "B" que es contactar con el Node Manager de la máquina y que éste comunique la instrucción de parada al servidor. En el caso de que esto tampoco funcione, nos queda el plan "C", que consiste en que el NodeManager solicite al sistema operativo que mate al proceso del servidor. Los tres flujos posibles son los siguientes:



La forma de lanzar manualmente este proceso es ejecutando el script `startNodeManager.sh` que reside en la carpeta server/bin de la instalación de WebLogic. Esto es así porque es un proceso común a nivel de máquina, independiente de los dominios definidos. Lo habitual es iniciar este proceso de forma automática con el arranque del sistema.

Recordad que no es necesario iniciar este proceso en aquellas máquinas donde sólo resida

el servidor de administración, aunque si en una misma máquina tenemos tanto servidor de administración como gestionados, lo vamos a necesitar.

Por último, comentar que la configuración de los dominios que debe gestionar el NodeManager reside en el archivo common/nodemanager/nodemanager.domains, enumerándose sus nombres junto con la ruta donde reside su configuración.

Error al iniciar NodeManager

Si al iniciar el NodeManager en Linux da un error de formato en el archivo nm_data.properties, basta con eliminar dicho archivo para solucionarlo.

2.4. Servidor gestionado (Managed Server)

Al configurar un dominio debemos conocer qué aplicaciones vamos a desplegar y qué necesidades van a tener en cuanto a recursos y tolerancia a fallos.

En función de esos parámetros vamos a tener que definir uno o varios servidores gestionados repartidos en una o varias máquinas que conformarán nuestro dominio.

Además, estos servidores pueden ser servidores independientes, o estar configurados en clúster de modo que se puedan repartir la carga de trabajo (clúster activo-activo).

También es posible utilizar la denominada **migración de servicios** por la cual, en caso de caída de una máquina podremos migrar los servicios que no sean replicables y que residen en dicha máquina (servidores JMS, JTA y servicios de usuario tipo Singleton) a otra perteneciente al mismo clúster.

Incluso podemos recurrir a herramientas de terceros para configurar nuestros servidores en clúster activo-pasivo (HA).

Resumiendo, hay muchas formas de hacer las cosas, pero no tenemos porqué ser clarividentes y crear desde el principio una estructura compleja. Basta con desplegar un servidor y dotarle de los recursos necesarios para ejecutar nuestras aplicaciones. Con el tiempo podremos adaptar la configuración del dominio a las nuevas necesidades del negocio.

2.4.1. Modos de arranque de un servidor gestionado

Anteriormente hemos explicado que en la configuración por defecto, se inicia en primer lugar el servidor admin, después el NodeManager y éste, posteriormente, "levanta" los servidores gestionados. El servidor admin comparte la configuración con el resto de servidores y éstos despliegan las aplicaciones que tienen asignadas. Pero, ¿Qué pasaría si el servidor admin no puede arrancar? Nuestro dominio quedaría totalmente inutilizado, es decir, el admin se convierte en un "punto único de fallo".

Afortunadamente hay mecanismos para evitarlo, como el denominado *Managed Server Independence*. Si un servidor tiene activada esta configuración y tiene en su sistema de archivos acceso al fichero config.xml (bien porque están en la misma máquina, o bien montando directorios por NFS) intentaría contactar durante un número de segundos (parametrizado) con el Admin, y si en ese periodo de tiempo no lo consigue arrancaría de forma independiente.

Un servidor arrancado de esta forma, podrá desplegar aquellas aplicaciones que tenga "cacheadas" pero no se podrá cambiar su configuración, hasta que el Admin no arranque de nuevo.

Otro posible "punto único de fallo" sería el NodeManager, por ejemplo, si el proceso se cuelga o aborta. Weblogic permite otra forma alternativa de iniciar un servidor: usar el script `startManagedWebLogic.sh`, que se encuentra en la carpeta bin del dominio. Como parámetro debemos indicar el nombre del servidor. Ejemplo:

```
startManagedWebLogic.sh "miServidor" "http://localhost:7001"
```

Si iniciamos un servidor de esta forma, no necesitaremos un NodeManager en nuestro sistema, pero hay que tener en cuenta que desde la consola de administración ya no vamos a poder iniciar servidores (dará error al no poder conectar con un NodeManager). Lo que sí que podremos hacer es pararlos (recordad que el shutdown es una orden que va directa del admin a los servidores gestionados).

2.4.2. Fichero de credenciales boot.properties

Anteriormente hemos comentado que un dominio puede iniciarse en modo desarrollo o modo producción, y esto variará el funcionamiento de los distintos componentes de WebLogic.

Cuando estamos en modo producción, todo servidor iniciado por script, solicita en su arranque un usuario válido del dominio de WebLogic, el admin. En el caso de un dominio en modo desarrollo, el servidor admin omite esta validación (pero no los servidores gestionados).

Una forma de automatizar el arranque, es invocar los scripts con las credenciales como parámetros, creando un script que invoque los propios de WebLogic. Esto supondría un problema de seguridad pues tendríamos un fichero con la password de un usuario en texto claro.

Para evitar estos problemas, WebLogic define un fichero denominado `boot.properties` que se encuentra dentro de la carpeta `security` de cada servidor (`dominio/servers/nombreServ/security`) Su contenido debe ser el siguiente:

```
username=usuario
password=clave
```

Si creamos este fichero, cuando arranquemos el servidor mediante script ya no nos va a

solicitar credenciales, y para mayor seguridad, el contenido del archivo se encriptará de forma automática.

2.5. Herramientas administrativas

WebLogic incluye una serie de herramientas de administración, en cierta medida parecidas a las estudiadas en GlassFish. Son las siguientes:

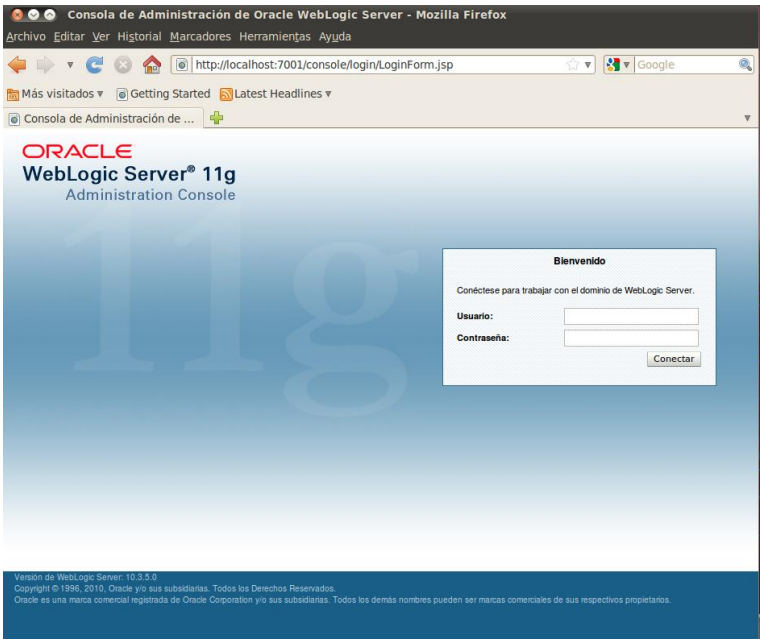
- Consola de administración del sistema
- Herramienta de scripting weblogic.WLST
- Herramienta de despliegue weblogic.Deployer
- WebLogic Maven plugin (12c)
- JMX, Mbeans
- Asistente de configuración
- *** edición manual config.xml ***

En versiones anteriores de WebLogic, se acudía directamente al archivo config.xml para realizar labores de administración (ejemplo: cambiar de un plumazo usuarios/passwords de conexiones a bases de datos). En la actualidad esta práctica está totalmente desaconsejada salvo situaciones excepcionales, que personalmente no me he encontrado...

2.5.1. Consola de administración del Sistema

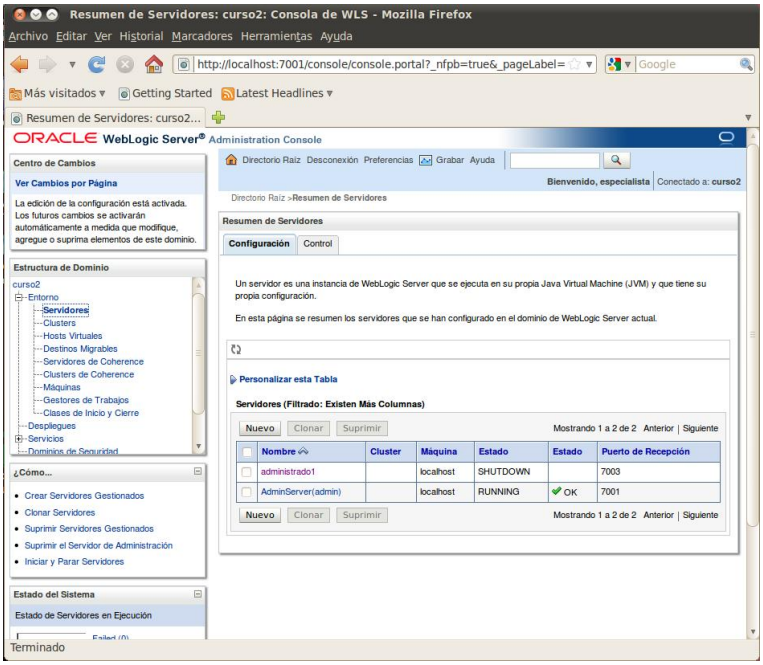
Si recordáis la consola de GlassFish, veréis que esta es una consola bastante más elaborada y cómoda de utilizar. Se trata de una aplicación basada en JSF, que se ejecuta en el servidor Admin. Para acceder a la consola basta con conectarnos a la dirección http://localhost:7001/console (asumiendo que el admin se encuentre en nuestra máquina y escuche por el puerto estándar).

Lo primero que veremos es la pantalla de bienvenida solicitándonos un usuario WebLogic:



Si os fijáis, en la parte inferior izquierda aparece la versión del producto (puede ser de utilidad para identificar una versión ya instalada).

Una vez introducido un usuario válido, el aspecto de la consola es el siguiente:



A la izquierda de la consola tenemos una representación de la estructura del dominio

donde "cuelgan" todos los elementos que lo conforman. En la parte superior se nos muestra la ruta de navegación (al estilo del explorador de Windows 7) en la que podemos ver en todo momento donde nos encontramos y volver a las ventanas anteriores. En la parte central tendremos la configuración del elemento que hemos seleccionado.

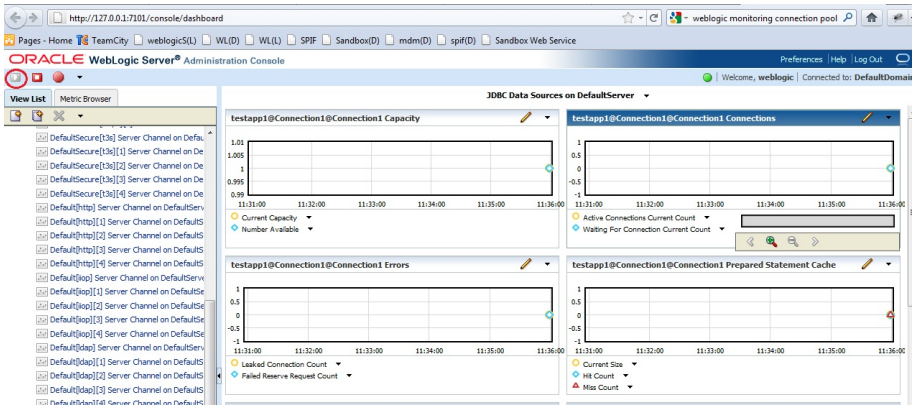
En la siguiente sesión entraremos más en detalle con ella.

2.5.2. Dashboard

Aunque esté incluida en la aplicación consola, se puede considerar una herramienta de monitorización independiente.

Para acceder a ella, debemos abrir en el navegador la siguiente dirección http://localhost:7001/console/dashboard, suponiendo que el servidor admin escuche por el puerto 7001.

El aspecto del Dashboard es el siguiente:



A la izquierda de la pantalla hay una serie de vistas predefinidas con información sobre el Heap, las conexiones JMS, JDBC etc. El Dashboard nos da la oportunidad de definir vistas personalizadas que contendrán las métricas que consideremos más importantes, y representarlas con diferentes tipos de gráficos.

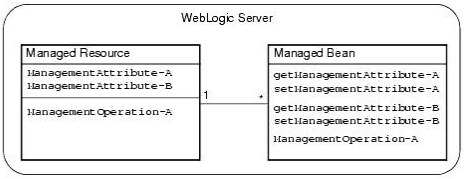
La idea es configurar un cuadro de mando que nos permita conocer de un vistazo el funcionamiento de una aplicación, servidor o incluso el dominio entero.

2.5.3. JMX y los Mbeans

Como la mayoría de los servidores, WebLogic permite explicar y modificar su configuración mediante servicios JMX (Java Management Extensions), que se utilizan tanto internamente para la comunicación entre los procesos, como interfaz para aplicaciones de terceros.

Trabajar con JMX es relativamente sencillo, pero hay que definir un par de conceptos:

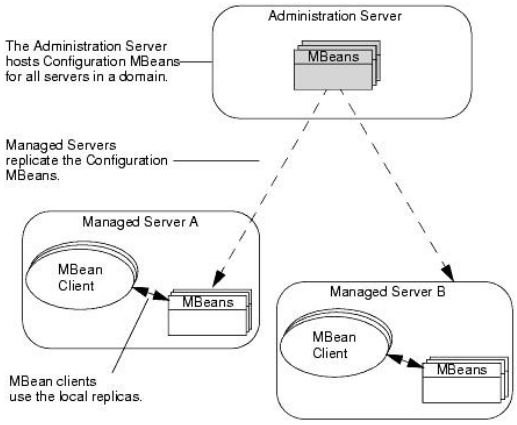
- **WebLogic Server managed resources**: La configuración de un dominio se agrupa en diferentes subsistemas: Orígenes de datos, JMS, Servidores, etc. Se denominan recursos administrados de WebLogic, (o habitualmente recursos) a cada elemento con entidad propia definido en cada uno de estos subsistemas. Por ejemplo, un pool de conexiones sería un recurso. Un recurso suele tener una serie de propiedades configurables (en el caso del pool, la conexión a la BD, número de conexiones, etc.), pero también puede tener una serie de operaciones definidas (por ejemplo, liberar conexiones del pool).
- **MBeans**: Los MBeans son clases Java definidas por la especificación JMX cuya única misión es exponer las propiedades (mediante getters y setters), y las operaciones definidas en cada recurso, para que sean accesibles a través de dicha interfaz. Salvando las distancias, trabajar con JMX se podría asemejar a trabajar con Transfer Objects.



Según la función que desempeñe un MBean, se pueden clasificar en:

- MBeans de configuración: Aquellos que exponen atributos y configuraciones de un recurso, tal y como hemos visto hasta ahora.
- MBeans de ejecución (Runtime MBeans). Estos beans se utilizan para acceder a propiedades que nos indiquen cómo está funcionando un componente, y se utilizan fundamentalmente en **monitorización**. En el caso del pool, podremos ver el número de conexiones creadas, cuantas están en ejecución, colas de espera, etc.

En cada servidor gestionado tendremos los Mbeans correspondientes para acceder a su configuración y sus propiedades de ejecución, pero además, éstos se replicarán en el servidor de administración, donde tendremos acceso a la configuración de todo el dominio. Gráficamente:



Desde la consola de WebLogic trabajaremos con Mbeans de forma transparente, pero si queremos una monitorización más detallada, o una herramienta de configuración más específica tendremos que recurrir a productos de terceros, o definirnos nuestras propias aplicaciones explotando JMX.

2.5.4. Herramientas de scripting weblogic.WLST

WLST es la "navaja suiza" de WebLogic. Se trata de una herramienta de administración de línea de comandos, al estilo de asadmin, pero que además es un cliente JMX que admite programación en un lenguaje script, denominado Jython (Python para Java).

Ejemplo de creación de clusters WebLogic mediante scripting:

```
from java.util import *
from javax.management import *
import javax.management.Attribute
print 'starting the script .... '
connect('username','password','t3://localhost:7001')
clusters = "cluster1","cluster2"
ms1 = {'managed1':7701,'managed2':7702,'managed3':7703, 'managed4':7704,
'managed5':7705}
ms2 = {'managed6':7706,'managed7':7707,'managed8':7708, 'managed9':7709,
'managed10':7710}
clustHM = HashMap()
edit()
startEdit()
for c in clusters:
  print 'creating cluster '+c
  clu = create(c,'Cluster')
  clustHM.put(c,clu)
cd('..\..')
clus1 = clustHM.get(clusters[0])
clus2 = clustHM.get(clusters[1])
for m, lp in ms1.items():
  managedServer = create(m,'Server')
  print 'creating managed server '+m
  managedServer.setListenPort(lp)
  managedServer.setCluster(clus1)
for m1, lp1 in ms2.items():
  managedServer = create(m1,'Server')
```

```
  print 'creating managed server '+m1
  managedServer.setListenPort(lp1)
  managedServer.setCluster(clus2)
save()
activate(block="true")
disconnect()
print 'End of script ...'
exit()
```

Como asadmin, puede trabajar en modo *interactivo* o *línea de comandos*, siendo necesario un fichero de entrada con las tareas a realizar. Además, tiene un tercer modo de trabajo, modo integrado, ya que se trata de una aplicación Java con un API que podemos utilizar para desarrollar nuestras propias herramientas de Administración.

Asimismo, en cuanto a la conexión o no con un servidor, se definen dos modos de trabajo: online y offline.

2.5.4.1. Modo online

En este modo, la herramienta nos proporciona una forma sencilla de interactuar con los MBeans del servidor utilizando de forma transparente el API JMX. El comportamiento es equivalente a modificar la configuración desde la consola, es decir nos tenemos que conectar al servidor, realizar los cambios en la configuración que necesitemos, indicar al servidor que aplique las modificaciones (en modo producción) y cerrar la conexión.

Con esta herramienta podemos conectarnos tanto a un servidor admin como a un gestionado, con la diferencia que en este último sólo podemos examinar las propiedades (no modificarlas).

2.5.4.2. Modo offline

En este modo de trabajo únicamente tendremos acceso a la información persistente de la configuración de un dominio (ejemplo config.xml), y podremos realizar tareas del tipo crear/modificar/borrar dominios, o hacer ingeniería inversa de la configuración, como veremos más adelante.

2.5.4.3. Pasos para utilizar WLST

1. Acceder a la carpeta:

```
cd /home/expertojava/Oracle/Middleware/wlserver_12.1/server/bin
```

2. Añadir las librerías de WebLogic al CLASSPATH y los binarios al PATH del sistema. Esto lo podemos hacer ejecutando:

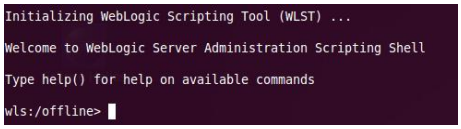
```
.  ./setWLSEnv.sh
```

Cuidado, si no utilizamos el operador "." no estableceremos las variables en la sesión del shell.

3. Ejecutar WLST. Para ello escribiremos:

```
java weblogic.WLST
```

Con este comando entraremos en el modo interactivo e inicialmente en offline:



Desde esta línea de comandos, podemos lanzar uno a uno las instrucciones del script de Jython que acabamos de ver, por ejemplo podemos conectarnos a un servidor mediante el comando:

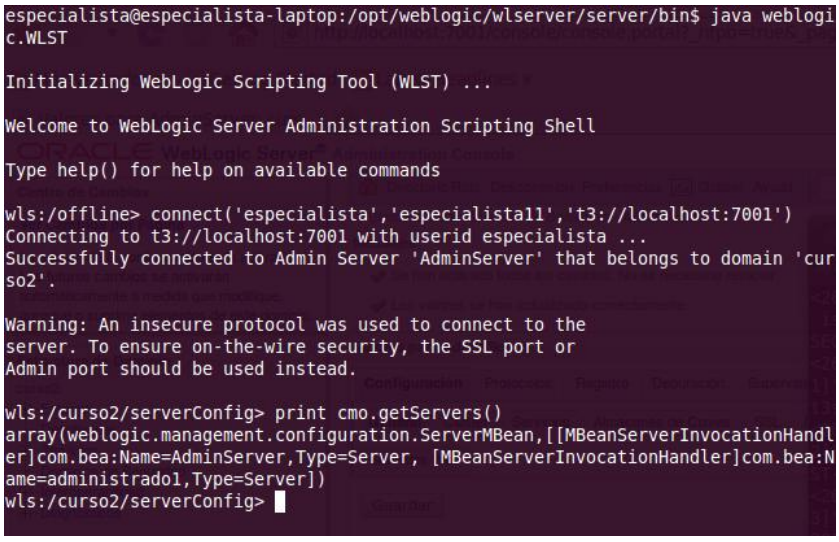
```
connect('username','password','t3://localhost:7001')
```

Fijaos que el protocolo especificado es t3 "a secas", es decir, no estamos utilizando encriptación SSL para ejecutar nuestros comandos. Oracle recomienda trabajar con t3s en Producción para mayor seguridad

Protocolos con encriptación

Para que funcione el t3s hay que habilitarlo en el servidor y configurar los certificados correspondientes en cliente y servidor, por tanto, en esta sesión nos conformaremos con t3.

De esta forma pasaríamos a modo online, y podríamos interrogar al sistema a través de sus Mbeans. Por ejemplo, sacar la lista de servidores del dominio.

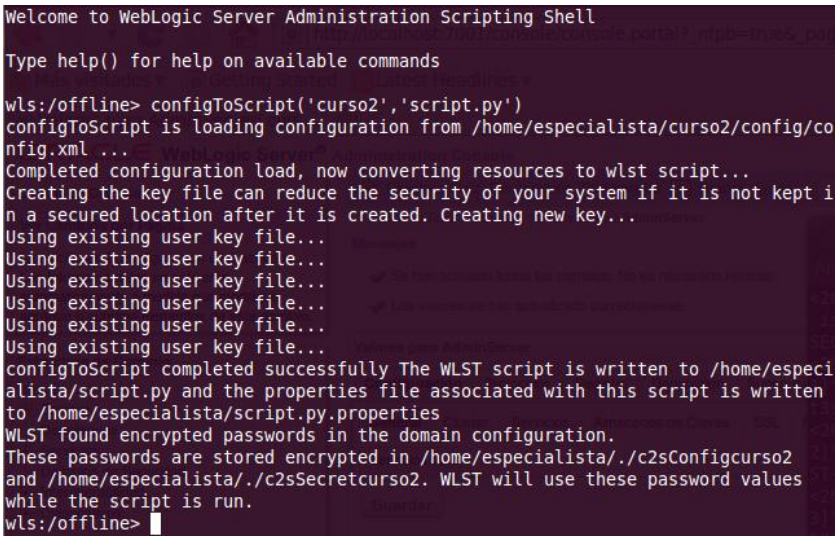


Acabamos de obtener los MBeans asociados a los servidores que componen el dominio. Evidentemente la gracia estará en hacer más cosas, a parte de un simple print...

La mejor forma de aprender es buscar en Internet ejemplos, como por ejemplo los de esta

página: http://wlstbyexamples.blogspot.com/2010/02/server-state-using-wlst.html

Otra opción es recurrir a la ingeniería inversa. Si tenemos un dominio ya configurado, podemos generar de forma automática el script de Jython que serviría para configurar de nuevo el dominio desde cero. El comando se denomina `configToScript` y recibe dos parámetros, la ubicación en disco del dominio, y el fichero en el que queremos volcar la salida:



Ahora basta con ejecutar `exit()` para salir de WLST y ya podremos editar el fichero script.py

Nota

Además del fichero de salida, se genera información adicional, como por ejemplo todas las credenciales encriptadas. Esta información se vuelca en los archivos que se indican en el proceso de generación del script.

El script generado contiene las funciones básicas en Jython para generar los distintos recursos pertenecientes al dominio, más una función "main" donde se ejecutan con los valores concretos de configuración. Estudiar estos scripts ayuda a comprender cómo trabaja la herramienta y qué cosas podemos hacer con ella.

Por último, si tuviésemos que crear un domino desde cero, bastaría con ejecutar:

```
java weblogic.WLST script.py
```

2.5.5. Herramienta de despliegue weblogic.Deployer

Se trata de otra herramienta Java, (se invoca igual que WLST), pero especializada en la gestión del ciclo de vida de las aplicaciones Java. Con esta herramienta podremos:

- Desplegar/replegar aplicaciones en un servidor.
- Iniciar/detener aplicaciones.
- Actualizar la versión de una aplicación.
- Listar las aplicaciones de un dominio.
- Distribuir aplicaciones.

Es muy habitual utilizar scripts de Unix que invoquen a esta herramienta para automatizar el despliegue y actualización de las aplicaciones Java Enterprise. De hecho, un sh configurado con el comando unix sudo, puede permitir que los desarrolladores puedan desplegar aplicaciones en WebLogic sin tener un usuario propio administrador.

Como ejemplo, para desplegar una nueva aplicación, la sintaxis sería la siguiente:

```
java weblogic.Deployer -adminurl http://localhost:7001
        -user weblogic -password weblogic -deploy c:\localfiles\myapp.ear
```

La línea de comandos habitual se compone por la conexión al admin, el comando y los parámetros adicionales que necesite. Los comandos más importantes son:

- deploy: Despliega una nueva aplicación a un servidor.
- undeploy: Elimina una aplicación de un servidor.
- redeploy: Redespliega una aplicación ya existente en ejecución.
- distribute: este comando sirve para hacer llegar una nueva aplicación a un servidor, pero sin iniciarla, a diferencia de deploy
- start/stop: activa/desactiva una aplicación.
- listapps: muestra la lista de aplicaciones desplegadas en nuestro dominio.

Uno de los parámetros más importantes es el de **staging mode**. Este parámetro define la forma en la que un servidor va a localizar y cargar una aplicación. Habitualmente se utilizan los valores `-stage` o `-nostage`, aunque existe un tercero:

- stage: En esta modalidad, cada servidor mantiene una carpeta "caché" denominada stage donde almacenará los EAR o WAR de las distintas aplicaciones que contiene. El servidor admin o la herramienta deployer, al desplegar una aplicación, acabarán dejando en la carpeta stage del servidor destino, una copia de la aplicación. Es el modo por defecto en los servidores gestionados.
- nostage: Este modo prescinde de la carpeta stage, lo que indicaremos en la consola será una ruta de disco donde poder localizar el binario. Hay que tener en cuenta que la ruta indicada debe ser visible tanto por el servidor Admin como el servidor o servidores destino. En caso contrario puede fallar incluso el arranque de estos servidores. Es el modo por defecto de un servidor Admin.
- external_stage: En este caso el administrador define una ruta concreta, relativa a cada servidor que se utilizará como carpeta stage. A diferencia del modo stage normal, es responsabilidad del usuario administrador el copiar las aplicaciones a dicha carpeta para que el servidor pueda desplegarlas. Se podría entender como una combinación de las dos opciones anteriores.

Lo habitual es utilizar el modo stage en dominios no demasiado complejos. Sin embargo

los modos nostage y external_stage exigen más trabajo por parte del administrador, pero agilizan el despliegue de las aplicaciones más pesadas.

Nota

Los comandos, descritos con más detalle los podéis consultar en este enlace: http://docs.oracle.com/cd/E13222_01/wls/docs90/deployment/wldeployer.html#1003230

2.5.6. WebLogic Maven plugin (12c)

Desde la versión 12c, WebLogic incluye un plugin para Maven que hace de envoltura de las herramientas anteriores, deployer y WLST. Este plugin expone como *goals* comandos de control de servidores, despliegue de aplicaciones y ejecución de scripts WLST, de modo que se puedan integrar con el ciclo de vida de una compilación con Maven.

En la sección de Referencias tenéis un enlace donde se explica cómo instalarlo y algunos ejemplos.

2.5.7. El asistente de instalación

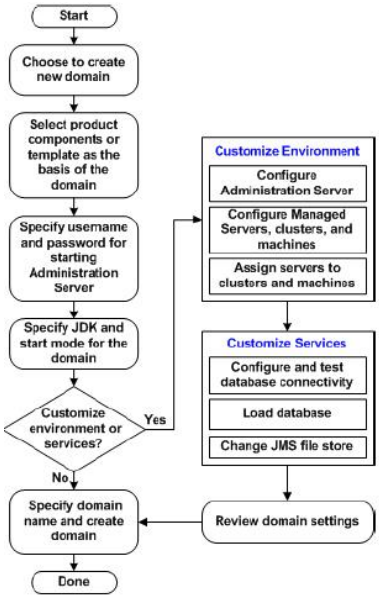
La última herramienta que vamos a ver es la primera que usaremos. Se trata de un asistente que nos permite en pocos pasos definir los diferentes servidores que compondrán nuestro dominio así como su configuración básica.

Podemos utilizar este script en cualquier momento, no sólo al instalar el producto. Para ello iremos a la carpeta `Oracle/MiddleWare/wlserver_12.1/common/bin/` y ejecutaremos:

```
sh config.sh
```

Nos aparecerá la siguiente ventana:

Con esta herramienta modelaremos nuestro nuevo dominio, siguiendo estos pasos:



Definiremos la ubicación del dominio, las credenciales de un usuario adminstrador, el JRE que utilizará para ejecutarse, y por último los distintos elementos que conforman el dominio: servidores, clústeres y máquinas. Con esta herramienta podemos configurar un dominio desde cero de forma rápida y sencilla.

2.5.7.1. Instalación en modo texto

Para configurar un dominio desde una sesión de telnet, ejecutaremos el mismo script, pero con parámetros adicionales:

```
sh config.sh -mode=console
```

2.5.7.2. Instalación desatendida

En este caso debemos tener preparado previamente un script con la configuración, y la orden sería:

```
sh config.sh -mode=silent [-silent_script=scriptfile1;scriptfile2;...]
[-log=logfile]
```

Lentitud de arranque

Al parecer la combinación Weblogic + Ubuntu + VirtualBox tiene ciertos problemas. Concretamente Weblogic se ralentiza notablemente al iniciar ciertos procesos (comunicación entre el admin y otros servidores, y el arranque en sí de la consola de administración). La causa parece estar en un bug de la JVM que se da en las circunstancias arriba descritas. La solución pasa por modificar el archivo `/jre/lib/security/java.security` y cambiar esta línea: `securerandom.source=file:/dev/urandom` por `securerandom.source=file:/dev/./urandom`. Más información en este enlace: http://www.itonguard.com/20090313/weblogic-starts-slow/

3. Referencias

- Resumen Administración WebLogic Server: http://docs.oracle.com/cd/E13222_01/wls/docs81/adminguide/overview.html
- Despliegue de aplicaciones: http://docs.oracle.com/cd/E15586_01/fusionapps.1111/e15524/adv_wls_e.htm
- Iniciar/parar Servidores: http://docs.oracle.com/cd/E13222_01/wls/docs81/ConsoleHelp/startstop.html
- WebLogic Scripting Tool: http://docs.oracle.com/cd/E13222_01/wls/docs90/config_scripting/using_WLST.html
- Ejemplos WLST: https://forums.oracle.com/forums/thread.jspa?threadID=895416
- Más ejemplos WLST: http://wlstbyexamples.blogspot.com/2010/05/weblogic-server-weblogic-scripting-tool.html
- JMX: http://docs.oracle.com/cd/E13222_01/wls/docs81/jmx/overview.html#1029146
- Novedades WebLogic 12c: http://docs.oracle.com/cd/E24329_01/web.1211/e24494/toc.htm
- Instalación y uso del plugin de Maven: https://blogs.oracle.com/arungupta/entry/wls_maven_plugin_weblogic_12c