BEA WebLogic: Introducción
Autor: Bea
Traductor: Juan Antonio Palos (Ozito)

- ¿Qué es BEA WebLogic Server
  - La solución WebLogic Server
  - Plataforma J2EE
  - Despliegue de Aplicaciones a través de Entornos Distribuidos y Heterogéneos
- ¿Qué es WebLogic Express?
- Arquitectura del Servidor de Aplicaciones WebLogic
- Capas de Componentes Software
  - Componentes de la Capa Cliente
  - Componentes de la Capa Media
  - Componentes de la Capa Backend
- Capas Lógicas de Aplicación
  - Capa Lógica de Presentación
    - Clientes de Navegador Web
    - Clientes No-Navegadores
  - Capa de Lógica de Negocio
    - Beans de Entidad
    - Beans de Sesión
    - Beans Dirigidos por Mensajes
  - Servicios de la Capa de Aplicación
    - HTTP
    - T3
    - RMI
    - RMI-IIOP
    - SSL
  - Datos y Servicios de Acceso
    - JNDI
    - JDBC
    - JTA
  - Tecnologías de Mensajería
    - JMS
    - Javamail
- WebLogic Server como Servidor Web
  - Cómo funciona el servidor WebLogic como un Servidor Web
  - Características del Servidor Web
    - Hosting Virtual
    - Usar Configuraciones de Servidor Proxy
    - Balance de Carga
    - Control de Fallos
- Servicios de Seguridad
  - Autentificación
  - Autorización
  - Alternartivas y Reinos Personalizados
  - Encriptación
- Clusters WebLogic
  - Ventajas de Usar Clusters
  - Arquitectura de un Cluster
  - Cómo se Define un Cluster WebLogic en una Red

- Servicios en Clusters
- Control del Servidor y Monitorización
  - Administrator Server
  - Consola de Administración

BEA WebLogic: Introducción
Autor: BEA
Traductor: Juan Antonio Palos (Ozito)

- ¿Qué es BEA WebLogic Server
  - La solución WebLogic Server
  - Plataforma J2EE
  - Despliegue de Aplicaciones a través de Entornos Distribuidos y Heterogéneos

¿Qué es BEA WebLogic Server

La solución WebLogic Server

El entorno de negocio de hoy en día demanda aplicaciones Web y de comercio electrónico que aceleren nuestra entrada en nuevos mercados, nos ayude a encontrar nuevas formas de llegar y de retener clientes, y nos permita presentar rápidamente productos y servicios. Para construir y desplegar estas nuevas soluciones, necesitamos una plataforma de comercio electrónico probada y creíble que pueda conectar y potenciar a todos los tipos de usuario mientras integra nuestros datos corporativos, las aplicaciones mainframe, y otras aplicaciones empresariales en una solución de comercio electrónico fin-a-fin poderosa y flexible. Nuestra solución debe proporcionar el rendimiento, la escalabilidad, y la alta disponibilidad necesaria para manejar nuestros cálculos de empresa más críticos.

Como una plataforma lider en la industria de comerción electrónico, **WebLogic Server** nos permite desarrollar y desplegar rápidamente, aplicaciones fiables, seguras, escalables y manejables. Maneja los detalles a nivel del sistema para que podamos concentrarnos en la lógica de negocio y la presentación

Plataforma J2EE

WebLogic Server utiliza tecnologías de la plataforma Java 2, Enterprise Edition (J2EE). J2EE es la plataforma estándard para desarrollar aplicaciones multi-capa basadas en el lenguaje de programación Java. Las tecnologías que componente J2EE fueron desarrolladas colaborativamente entre Sun Microsystems y otros vendedores de software entre los que se incluye BEA Systems.

Las aplicaciones J2EE están basadas en componentes estandarizados y modulares. WebLogic Server proporciona un conjunto completo de servicios para esos componentes y maneja automáticamente muchos detalles del comportamiento de la aplicación, sin requerir programación.

Despliegue de Aplicaciones a través de Entornos Distribuidos y Heterogéneos

WebLogic Server proporciona características esenciales para desarrollar y desplegar aplicaciones críticas de comercio electrónico a través de entornos de computación distribuidos y heterogéneos. Entre estas características están las siguientes:

- Estándars de Liderazgo—Soporte Comprensivo de Java Enterprise para facilitar la implementación y despliegue de componentes de aplicación. WebLogic Server es el primer servidor de aplicaciones independientes desarrollado en Java en conseguir la certificación J2EE.
- Ricas Opciones de Cliente—WebLogic Server soporta nevegadores Web y otros clientes que usen HTTP; los clientes Java que usan RMI (Remote Method Invocation) o IIOP (Internet Inter-ORB Protocol); y dispositivos móviles que usan WAP (Wireless Access Protocol). Los conectores de BEA y otras compañias permiten virtualmente a cualquier cliente o aplicación legal trabajar con el Servidor de aplicaciones WebLogic.
- Escalabilidad de comercio electrónico empresarial—Los recursos críticos se usan eficientemente y la alta disponibilidad está asegurada a través del uso de componentes **Enterprise JavaBean** y mecanismos como el "clustering" de WebLogic Server para las páginas Web dinámicas, y los almacenes de recursos y las conexiones compartidas.
- Administración Robusta—WebLogic Server ofrece una Consola de Administración basada en Web para configurar y monitorizar los servicios del WebLogic Server. Se hace conveniente para la configuración un interface de línea de comandos para administrar el servidor WebLogic on Scripts.
- Seguridad Lista para el Comercio Electrónico—WebLogic Server proporciona soporte de Secure Sockets Layer (SSL) para encriptar datos transmitidos a través de WebLogic Server, los clientes y los otros servidores. La seguridad de WebLogic permite la autentificación y autorización del usuario para todos los sevicios de **WebLogic Server**. Los almacenes de seguridad externa, como servidores Lightweight Directory Access Protocol (LDAP), puede adaptarse para WebLogic, nos permiten una sola autentificación para empresas. El "Security Service Provider Interface" hace posible extender los servicios de seguridad de WebLogic e implementar estas características en aplicaciones.
- Máxima flexibilidad en el desarrollo y despliegue—WebLogic Server proporciona una estrecha integración y soporte con las bases de datos más importantes, herramientas de desarrollo y otros entornos.

BEA WebLogic: Introducción
Autor: BEA
Traductor: Juan Antonio Palos (Ozito)

- ¿Qué es WebLogic Express?

¿Qué es WebLogic Express?

BEA WebLogic Express : Es una plataforma escalable que sirve contenido y datos dinámicos a la Web y a las aplicaciones inhalámbricas. WebLogic Express incorpora la presentación y los servicios del acceso de base de datos de WebLogic Server, permitiendo a los desarrolladores crear rápidamente aplicaciones de negocio eléctronico interactivas y transaccionales y proporcionar los servicios de presentación para las aplicaciones existentes.

WebLogic Express ofrece muchos de los servicios y APIs disponibles con el WebLogic Server, incluyendo características de WebLogic JDBC, las JavaServer Pages (JSP), los Servlets, la Invocación Remotra de Métodos (RMI), y las funcionalidades de servidor web.

WebLogic Express se diferencia de WebLogic Server en que WebLogic Express no proporciona JavaBeans Enterprise (EJB), servicios de mensaje de Java (JMS), ni el protocolo de transacciones de dos fases.

BEA WebLogic: Introducción
Autor: BEA
Traductor: Juan Antonio Palos (Ozito)

- Arquitectura del Servidor de Aplicaciones WebLogic

Arquitectura del Servidor de Aplicaciones WebLogic

WebLogic Server es un servidor de aplicaciones: una plataforma para aplicaciones empresariales multi-capa distribuidas. WebLogic Server centraliza los servicios de aplicación como funciones de servidor webr, componentes del negocio, y acceso a los sistemas "backend" de la empresa. Utiliza tecnologías como el almacenamiento en memoria inmediata y almacenes de conexiones para mejorar la utilización de recursos y el funcionamiento de la aplicación. WebLogic Server también proporciona facilidades a nivel de seguridad empresarial y una administración poderosa.

WebLogic Server funciona en la capa media (o capa "n") de una arquitectura multi-capa. Una arquitectura multi-capa determina dónde se ejecutan los componentes software que crean un sistema de cálculo en relación unos con otros y al hardware, la red y los usuarios. Elegir la mejor localización para cada componente software nos permite desarrollar aplicaciones más rápidamente; facilita el despliegue y la administración; y proporciona un mayor control sobre el funcionamiento, la utilización, la seguridad, el escalabildad, y la confiabilidad.

WebLogic Server implementa J2EE, el estándar para la empresa de Java. Java es un lenguaje de programación, seguro ante la red, orientado a objetos, y J2EE incluye la tecnología de componentes para desarrollar objetos distribuidos. Estas funciones agregan una segunda dimensión architectura del servidor de aplicaciones WebLogic Server-- un capa de lógica de aplicación, con cada capa desplegada selectivamente entre las tecnologías J2EE de WebLogic Server.

Las dos secciones siguientes describen estas dos vistas de la arquitectura de WebLogic Server: capas de software y capas de la lógica de la aplicación.

BEA WebLogic: Introducción
Autor: BEA
Traductor: Juan Antonio Palos (Ozito)

- Capas de Componentes Software
    - Componentes de la Capa Cliente
    - Componentes de la Capa Media
    - Componentes de la Capa Backend

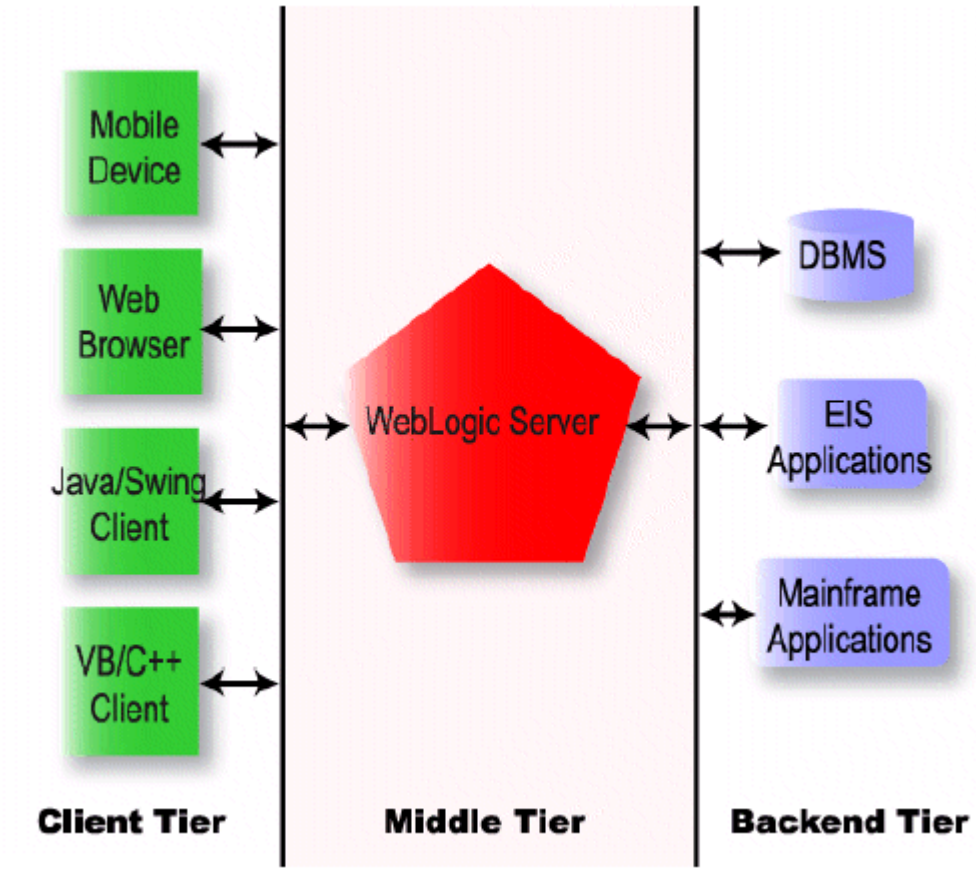
Capas de Componentes Software

Los componentes software de una arquitectura multi-capa constan de tres capas:

- La capa del **cliente** contiene los programas ejecutados por los usuarios, incluyendo navegadores Web y programas de aplicaciones de red. Estos programas se pueden escribir virtualmente en cualquier lenguaje de programación.
- La capa **media** contiene el servidor WebLogic y otros servidores que son direccionados directamente por los clientes, como servidores web existentes o servidores proxy.
- La capa **backend** contiene recursos de empresa, como sistemas de base de datos, aplicaciones de unidad central y legales, y aplicaciones de plannings de recursos de empresa empaquetados (ERP).

Las aplicaciones del cliente tienen acceso al servidor WebLogic directamente, o a través de un servidor web o un proxy. El servidor WebLogic conecta con servicios **backend** por cuenta de los clientes, pero los clientes no tienen acceso directamente a los servicios backend.

La figura 1-1 ilustra estas tres capas de la arquitectura del servidor **WebLogic Server**.



- **Componentes de la Capa Cliente**

Los clientes del servidor WebLogic utilizan interfaces estándares para acceder a servicios del servidor WebLogic. El servidor WebLogic tiene una completa funcionalidad de servidor web, así que un navegador web puede solicitar páginas al servidor WebLogic usando el protocolo estándar de la Web, HTTP. Los servlets de WebLogic Server y las JavaServer Pages (JSPs) producen páginas Web dinámicas, personalizadas requeridas para las aplicaciones avanzadas de comercio electrónico. Los programas del cliente escritos en Java pueden incluir interfaces gráficos de usuario altamente interactivos construidos con las clases de Java Swing. También se puede tener acceso a servicios del servidor WebLogic usando los APIs estándar del J2EE.

Todos estos servicios también están disponibles para los clientes de navegadores web desplegando servlets y páginas JSP en el servidor WebLogic. Los programas del cliente compatibles con CORBA escritos en Visual Basic, C++, Java, y otros lenguajes de programación pueden ejecutar JavaBeans Enterprise y RMI en el servidor WebLogic usando WebLogic RMI-IIOP. Las aplicaciones del cliente escritas en cualquier lenguaje que soporten el protocolo HTTP pueden acceder a cualquier servicio del WebLogic Server a través de un servlet.

- **Componentes de la Capa Media**

La capa media incluye el servidor WebLogic y otros servidores Web, cortafuegos, y servidores proxy que median en el tráfico entre los clientes y el servidor WebLogic. El servidor WAP de Nokia, parte de la solución de comercio móvil de BEA, es un ejemplo de otro servidor de la capa media que proporciona una conectividad entre los dispositivos inhalámbricos y el servidor WebLogic. Las aplicaciones basadas en una arquitectura multi-capa requieren confiabilidad, escalabilidad, y un alto rendimiento en la capa media. El servidor de aplicaciones que seleccionemos para la capa media es, por lo tanto, crítico para el éxito de nuestro sistema.

La opción **Cluster** del servidor WebLogic permite que distribuyamos peticiones de cliente y servicios **backend** entre varios servidores WebLogic cooperantes. Los programas en la capa del cliente acceder al cluster como si fuera un solo servidor WebLogic. Cuando la carga de trabajo aumenta, podemos agregar otros servidores WebLogic al cluster para compartir el trabajo. El cluster utiliza un algoritmo de balance de capa seleccionable para elegir el servidor WebLogic del cluster que es capaz de manejar la petición. Cuando una petición falla, otro servidor WebLogic que proporciona el servicio solicitado puede asumir el control. Los fallos son transparentes siempre que sea posible, lo que reduce al mínimo la cantidad de código que se debe escribir para recuperar incidentes. Por ejemplo, el estado de la sesión de un servlet se puede replicar en un servidor secundario WebLogic de modo que si el servidor WebLogic que está manejando una petición falla, la sesión del cliente se pueda reanudar de forma ininterrumpida desde el servidor secundario. Todos los servicios de WebLogic, EJB, JMS, JDBC, y RMI están implementados con capacidades de clustering.

- **Componentes de la Capa Backend**

La capa **backend** contiene los servicios que son accesibles a los clientes sólo a través del servidor WebLogic. Las aplicaciones en la capa backend tienden a ser los recursos

más valiosos y de misiones críticas para empresa. El servidor WebLogic los protege de accesos directos de usuarios finales. Con tecnologías tales como almacenes de conexiones y caches, el servidor WebLogic utiliza eficientemente los recursos **backend** y mejora la respuesta de la aplicación.

Los servicios **backend** incluyen bases de datos, sistemas de hojas de operación (planning) de recursos de la empresa (ERP), aplicaciones mainframe, aplicaciones legales de la empresa, y monitores de transacciones. Las aplicaciones existentes de la empresa se pueden integrar en la capa **backend** usando la especificación de configuración del conector Java (JCA) de Sun Microsystems. El servidor WebLogic hace fácil agregar un interface Web a una aplicación **backend** integrada. Un sistema de control de base de datos es el servicio **backend** más común, requerido por casi todas las aplicaciones del servidor WebLogic. WebLogic EJB y WebLogic JMS normalmente almacena datos persistentes en una base de datos en la capa **backend**.

Un almacen de conexiones JDBC, definido en el servidor WebLogic, abre un número predefinido de conexiones a la base de datos. Una vez que estén abiertas, las conexiones a la base de datos son compartidas por todas las aplicaciones del servidor WebLogic que necesiten acceder a esa base de datos. Sólo se incurre una sóla vez en la costosa sobrecarga asociada con el establecimiento de conexiones para cada conexión del almacén, por cada petición de cliente. El servidor WebLogic vigila las conexiones a la base de datos, refrescándolas cuando es necesario y asegurándose de la fiabilidad de los servicios de la base de datos para las aplicaciones.

WebLogic Enterprise Connectivity, que proporciona acceso a BEA WebLogic Enterprisesystems, y Jolt® para WebLogic Server que proporciona acceso a los sistemas Tuxedo® de BEA, también utilizan almacenes de conexiones para mejorar el funcionamiento del sistema.

BEA WebLogic: Introducción
Autor: BEA
Traductor: Juan Antonio Palos (Ozito)

- Capas Lógicas de Aplicación
  - Capa Lógica de Presentación
    - Clientes de Navegador Web
    - Clientes No-Navegadores
  - Capa de Lógica de Negocio
    - Beans de Entidad
    - Beans de Sesión
    - Beans Dirigidos por Mensajes
  - Servicios de la Capa de Aplicación
    - HTTP
    - T3
    - RMI
    - RMI-IIOP
    - SSL
  - Datos y Servicios de Acceso
    - JNDI
    - JDBC
    - JTA
  - Tecnologías de Mensajería

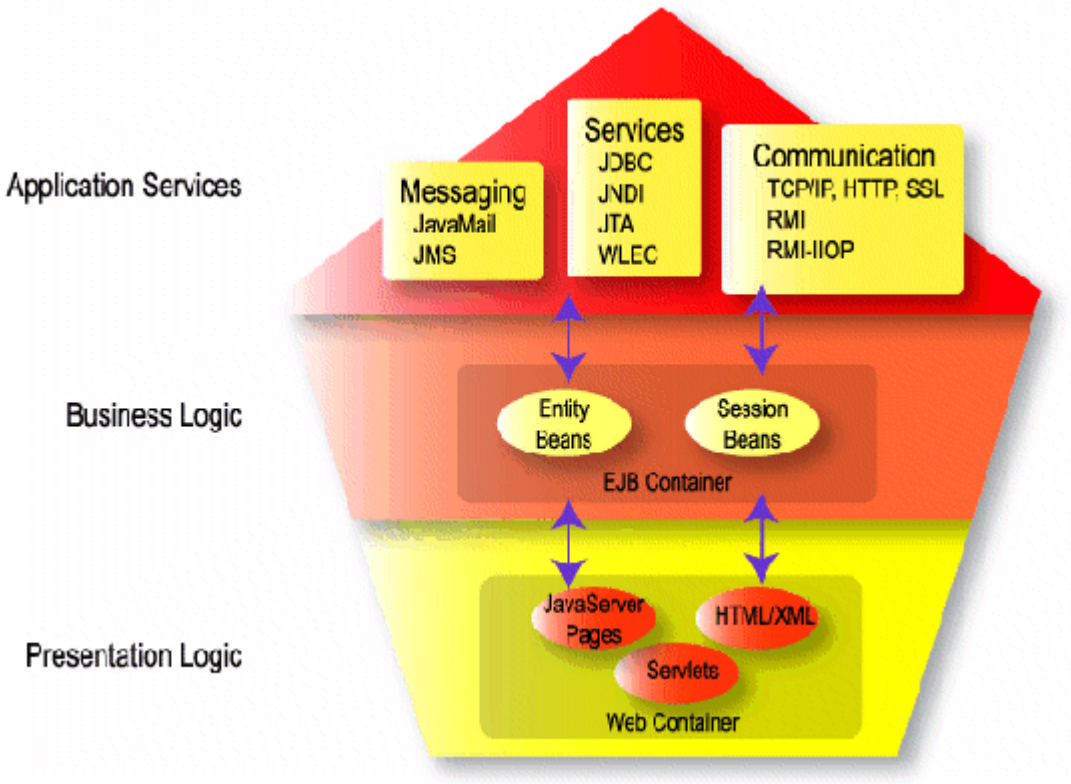
- JMS
- Javamail

Capas Lógicas de Aplicación

El servidor WebLogic implementa tecnologías de componentes y servicios J2EE. Las tecnologías de componentes J2EE incluyen servlets, páginas JSP, y JavaBeans Enterprise. Los servicios J2EE incluyen el acceso a protocolos de red, a sistemas de base de datos, y a sistemas estándares de mensajería. Para construir una aplicación de servidor WebLogic, debemos crear y ensamblar componentes, usando los APIs de sevicio cuando sean necesarios. Los componentes se ejecutan en contenedor Web del servidor WebLogic o el contenedor de EJB. Los contenedores proporcionan soporte para ciclo vital y los servicios definidos por las especificaciones J2EE de modo que los componentes que construyamos no tengan que manejar los detalles subyacentes.

Los componentes Web proporcionan la lógica de presentación para las aplicaciones J2EE basadas en navegador. Los componentes EJB encapsulan objetos y procesos del negocio. Las aplicaciones Web y los EJBs se construyen sobre servicios de aplicación de J2EE, como JDBC, JMS (servicio de mensajería de Java), y JTA (API de Transaciones de Java).

La Figura 1-2 ilustra los contenedores de componentes y los servicios de aplicación de WebLogic Server.



La siguientes secciones explican la capa de presetnación, la lógica del negocio y los servicios de aplicación.

- **Capa Lógica de Presentación**

La capa de presentación incluye una lógica de interface de usuario y de visualización de aplicaciones. La mayoría de las aplicaciones J2EE utilizan un navegador web en la máquina del cliente porque es mucho más fácil que programas de cliente que se despliegan en cada ordenador de usuario. En este caso, la lógica de la presentación es el contenedor Web del servidor WebLogic. Sin embargo, los programas del cliente escritos en cualquier lenguaje de programación, sin embargo, deben contener la lógica para representar el HTML o su propia lógica de la presentación.

- **Clientes de Navegador Web**

Las aplicaciones basadas en Web construidas con tecnologías web estándar son fáciles de acceder, de mantener, y de portar. Los clientes del navegador web son estándares para las aplicaciones de comercio electrónico. En aplicaciones basadas en Web, el interface de usuario esrá representado por los documentos HTML, las páginas JavaServer (JSP), y los servlets. El navegador web contiene la lógica para representar la página Web en el ordenador del usuario desde la descripción HTML.

Las JavaServer Pages (JSP) y los servlets están muy relacionados. Ambos producen el contenido dinámico de la Web ejecutando el código Java en el servidor WebLogic cada vez que se les invoca. La diferencia entre ellos es que JSP está escrito con una versión extendida de HTML, y los servlets se escriben con el lenguaje de programación Java.

JSP es conveniente para los diseñadores Web que conocen HTML y están acostumbrados al trabajo con un editor o diseñador de HTML. Los Servlets, escritos enteramente en Java, están más pensados para los programadores Java que para los diseñadores Web. Escribir un servlet requiere un cierto conocimiento del protocolo HTTP y de la programación de Java. Un servlet recibe la petición HTTP en un objeto request y escribe el HTML (generalmente) en un objeto result.

Las páginas JSP se convierten en servlets antes de que se ejecuten en el servidor WebLogic, por eso en última instancia las páginas JSP y los servlets son distintas representaciones de la misma cosa. Las páginas JSP se despliegan en el servidor WebLogic la misma forma que se despliega una página HTML. El fichero .jsp se copia en un directorio servido por WebLogic Server. Cuando un cliente solicita un fichero jsp, el servidor WebLogic controla si se ha compilado la página o si ha cambiado desde la última vez que fie compilada. Si es necesario llama el compilador WebLogic JSP, que genera el código del servlet Java del fichero jsp, y entonces compila el código Java a un fichero de clase Java.

- **Clientes No-Navegadores**

Un programa cliente que no es un navegador web debe suminstrar su propio código para represetnar el interface de usuario. Los clientes que no son navegadores generalmente contienen su propia presentación y lógica de la representación, dependiendo del servidor WebLogic solamente para la lógica y el acceso al negocio o a los servicios **backend**. Esto los hace más difícil de desarrollar y de desplegar y menos convenientes para las aplicaciones basadas en Internet de comercio electrónico que los clientes basados en navegador.

Los programas cliente escritos en Java pueden utilizar cualquier servicio del servidor WebLogic sobre Java RMI (Remote Method Invocatio). RMI permite que un programa cliente opere sobre un objeto del servidor WebLogic la misma forma que operaría sobre un objeto local en el cliente. Como RMI oculta los detalles de las llamadas a tavés de la red, el código del cliente J2EE y el código del lado del servidor son muy similares.

Los programas Java pueden utilizar las clases Swing de Java para crear interfaces de usuario poderosas y portables. Aunque usando Java podemos evitar problemas de portabilidad, no podemos utilizar los servicios del servidor WebLogic sobre RMI a menos que las clases del servidor WebLogic estén instaladas en el cliente. Esto significa que lo clientes RMI de Java no son adecuados para el comercio electrónico. Sin embargo, pueden usarse con eficacia en aplicaciones de empresariales en las cuales una red interna hace vieables la instalación y el mantenimiento. Los programas cliente escritos en lenguajes distintos a Java y los programas clientes Java que no utilizan objetos del servidor WebLogic sobre RMI pueden tener acceso al servidor WebLogic usando HTTP o RMI-IIOP.

HTTP es el protocolo estándar para la Web. Permite que un cliente haga diversos tipos de peticiones a un servidor y pase parámetros al servidor. Un servlet en el servidor WebLogic puede examinar las peticiones del cliente, extraer los parámetros de la petición, y preparar una respuesta para el cliente, usando cualquier servicio del servidor WebLogic. Por ejemplo, un servlet puede responder a un programa cliente con un documento de negocio en XML. Así una aplicación puede utilizar servlets como **gateways** a otros servicios del servidor WebLogic.

WebLogic RMI-IIOP permite que los programas compatibles con CORBA ejecuten Beans Enterprise del servidor WebLogic y clases RMI como objetos CORBA. El servidor RMI de WebLogic y los compiladores de EJB pueden generar IDL (Interface Definition Language) para las clases RMI y los Beans Enterprise. El IDL generado de esta manera se compila para crear los esqueletos para un ORB (Object Request Broker) y los trozos para el programa cliente. El servidor WebLogic analiza peticiones entrantes IIOP y las envía al sistema de ejecución RMI.

- **Capa de Lógica de Negocio**

Los JavaBeans Enterprise son componentes d ela lógica de negocio para aplicaciones J2EE. El contenedor EJB de WebLogic Server almacenca beans enterprise, proporcionan el control del ciclo de vida y servicios como el caheo, la persistenca, y el control de transaciones. Aquí tenemos tres tipos de beans enterprise: beans de entidad, beans de sesión y beans dirigidos por mensajes. Las siguientes secciones describen cada tipo en más detalle.

- **Beans de Entidad**

Un bean de entidad representa un objeto que contiene datos, como un cliente, una cuenta, o un item de inventario. Los beans de entidad contienen los valores de datos y los métodos que se pueden invocar sobre esos valores. Los valores se salvan en una base de datos (que usa JDBC) o algún otro almacén de datos. Los beans de entidad pueden participar en transacciones que implican otros beans enterprise y servicios transaccionales.

Los beans de entidad se asocian a menudo a objetos en bases de datos. Un bean de entidad puede representar una fila en un vector, una sola columna en una fila, o un resultado completo del vector o de la consulta. Asociado con cada bean de entidad hay una clave primaria única usada para encontrar, extraer, y grabar el bean.

Un bean de entidad puede emplear uno de los siguientes:

- **Persistencia controlada por el bean**. El bean contiene código para extraer y grabar valores persistentes.
- **Persistencia controlada por el contenedor.** El contenedor EJB carga y graba valores en nombre del bean.

Cuando se utiliza la persistencia controlada por el contenedor, el compilador WebLogic EJB puede generar clases de soporte de JDBC para asociar un bean de entidad a una fila de una base de datos. Hay disponibles otros mecanismos de persistencia controlada por el contenedor. Por ejemplo, TOPLink para BEAWebLogic Server, de "The Object People" (http://www.objectpeople.com), proporciona persistencia para una base de datos de objetos relacionales.

Los beans de entidad pueden ser compartidos por muchos clientes y aplicaciones. Un ejemplar de un bean de entidad se puede crear a petición de cualquier cliente, pero no desaparece cuando ese cliente desconecta. Continúa viviendo mientras cualquier cliente lo esté utilizando activamente. Cuando el bean ya no se usa, el contenedor EJB puede pasivizarlo: es decir, puede eliminar el ejemplar vivo del servidor.

Beans de Sesión

Un bean de sesión es un ejemplar transitorio de EJB que sirve a un solo cliente. Los beans de sesión tienden a implementar lógica de procedimiento; incorporan acciones en vez de datos. El contenedor EJB crea un bean de sesión en una petición del cliente. Entonces mantiene el bean mientras el cliente mantiene su conexión al bean. Los beans de sesión no son persistentes, aunque pueden salvar datos a un almacén persistente si lo necesitan.

Un bean de sesión puede ser con o sin estado. Los beans de sesión sin estado no mantienen ningún estado específico del cliente entre llamadas y pueden ser utilizados por cualquier cliente. Pueden ser utilizados para proporcionar acceso a los servicios que no dependen del contexto de una sesión, como enviar un documento a una impresora o extraer datos de sólo lectura en una aplicación. Un bean de sesión con estado mantiene el estado en nombre de un cliente específico.

Los beans de sesión con estado pueden ser utilizados para manejar un proceso, como ensamblar una orden o encaminar un documento con un flujo de proceso. Como pueden acumular y mantener el estado con interacciones múltiples con un cliente, los beans de sesión son a menudo la introducción a los objetos de una aplicación. Como no son persistentes, los beans de sesión deben terminar su trabajo en una sola sesión y utilizar JDBC, JMS, o beans de entidad para registrar el trabajo permanentemente.

- **Beans Dirigidos por Mensajes**

Los beans dirigidos por Mensaje, introducidos en la especificación EJB 2,0, son los beans enterprise que manejan los mensajes asíncronos recibidos de colas de mensaje JMS. JMS encamina mensajes a un bean dirigido por mensaje, que selecciona un ejemplar de un almacen para procesar el mensaje.

Los beans dirigidos por Mensajes se manejan en el contenedor del servidor EJB de WebLogic. Como las aplicaciones dirigidas al usuario no los llaman directamente, no pueden ser accedidas desdee una aplicación usando un EJB home. Sin embargo, una aplicación dirigida al usuario si puede ejemplarizar un bean dirigidopor mensajes indirectamente, enviando un mensaje a la cola de bean JMS.

- **Servicios de la Capa de Aplicación**

El servidor WebLogic proprociona los servicios fundamentales que permiten que los componentes se concentren en lógica del negocio sin la preocupación por detalles de implementación de bajo nivel. Maneja el establecimiento de red, la autentificación, la autorización, la persistencia, y el acceso a objetos remotos para EJBs y servlets. Los APIs Java estándar proporcionan acceso portable a otros servicios que una aplicación puede utilizar, por ejemplo base de datos y servicios de mensajería.

Las aplicaciones cliente de tecnologías de comunicación en la red conectan con el servidor WebLogic usando protocolos de establecimiento de una red estándar sobre TCP/IP. El servidor WebLogic escucha peticiones de conexión en una dirección de red que se pueda especificar como parte de un identificador de recuros uniforme (URI). Un URI es una cadena estandardizada que especifica un recurso en una red, incluyendo Internet. Contiene un especificador de protocolo llamado un esquema, la dirección de red del servidor, el nombre del recurso deseado, y los parámetros opcionales. La URL que introducimos en un navegador web, por ejemplo, http://www.bea.com/index.html, es el formato más familiar de URI.

Los clientes basados en Web se comunican con el servidor WebLogic usando el protocolo HTTP. Los clientes Java conectan usando Java RMI, que permite que un cliente Java ejecute objetos en servidor WebLogic. Los clientes CORBA tienen acceso a objetos RMI desde el servidor WebLogic usando RMI-IIOP, que le permite que ejecutar objetos del servidor WebLogic usando protocolos estándar de CORBA.

Esquema	Protocolo
HTTP	HyperText Transfer Protocol. Utilizado por los navegadores Web y los programas compatibles HTTP.
HTTPS	HyperText Transfer Protoco over Secure Layers (SSL). Utilizado por programas de navegadores Web y clientes compatibles HTTPS.
T3	Protocolo T3 de WebLogic para las conexiones de Java-a-Java, que multiplexa JNDI, RMI, EJB, JDBC, y otros servicios de WebLogic sobre una conexión de red.
T3S	Protocolo T3S de WebLogic sobre sockets seguros (SSL).
IIOP	Protocolo Internet de IIOP Inter-ORB, usado por los clientes de Java compatibles con CORBA para ejecutar objetos WebLogic RMI sobre IIOP.

Otros clientes de CORBA conectan con el servidor WebLogic con un CORBA que nombra contexto en vez de un URI para las capas de la lógica de WebLogic Server.

El esquema en un URI determina el protocolo para los intercambios de la red entre un cliente y el servidor WebLogic. Los protocolos de red de la tabla 1-1.

Las secciones siguientes proporcionan más información sobre estos protocolos.

- **HTTP**

HTTP, el protocolo estándar del World Wide Web, es un protocolo de petición-respuesta. Un cliente publica una petición que incluya ana URI. La URI comienza con **HTTP: / /** y la dirección del servidor WebLogic, y el nombre de un recurso en el servidor WebLogic, como una página html, un servlet, o una página JSP. Si se omite el nombre del recurso, el servidor WebLogic devuelve la página Web del valor por defecto, generalmente **index.html**. La cabecera de una petición HTTP incluye un comando, generalmente GET o POST. La petición puede incluir parámetros de datos y un mensaje de contenido.

WebLogic de BEA WebLogic responde siempre a una petición HTTP ejecutando un servlet, que devuelve resultados al cliente. Un servlet HTTP es una clase Java que puede tener acceso al contenido de una petición HTTP recibida sobre la red y devuelve al cliente un resultado compatible HTTP.

El servidor WebLogic dirige una petición de una página HTML al servlet File interno. Este servlet busca el fichero HTML en el directorio de documentos del sistema de ficheros del servidor WebLogic. Una petición para un servlet personalizado ejecuta la clase Java correspondiente en el servidor WebLogic. Una petición de una página JSP hace que el servidor WebLogic compile la página JSP en un servlet, si no se ha compilado ya, y después ejecuta el servlet, que devuelve los resultados al cliente.

- **T3**

T3 es un protocolo optimizado usado para transportar datos entre el servidor WebLogic y otros programas Java, incluyendo clientes y otros servidores WebLogic. El servidor WebLogic no pierde l apista de cada máquina virtual Java (JVM) con que conecta, y crea una sola conexión T3 para llevar todo el tráfico con una JVM.

Por ejemplo, si un cliente Java tiene acceso a un bean enterprise y a un almacen de conexiones JDBC en el servidor WebLogic, se establece una sola conexión de red entre el servidor WebLogic y la JVM del cliente. Los servicios de EJB y de JDBC pueden ser escritos como si tuvieran uso único de una conexión de red dedicada porque el protocolo T3 multiplexa invisiblemente los paquetes en la sola conexión. T3 es un protocolo eficiente para las aplicaciones Java-a-Java porque evita eventos innecesarios de conexión a red y utiliza pocos recursos del SO. El protocolo también tiene mejoras internas que reducen al mínimo el tamaño de los paquetes.

- **RMI**

RMI es el recurso estándar de Java para las aplicaciones distribuidas. RMI permite que un programa Java, llamado el servidor, publique objetos Java que otro programa Java, llamado cliente, puede ejecutar. En la mayoría de las aplicaciones, el servidor WebLogic es el servidor RMI y una aplicación cliente Java es el cliente. Pero los papeles pueden ser invertidos; RMI permite que cualquier programa Java desempeñe el papel de servidor. La configuración RMI es similar a la configuración de CORBA. Para crear un objeto remoto, un programador escribe un interface para una clase Java que defina los métodos que puede ejecutar un cliente remoto. El compilador RMI del servidor WebLogic, **rmic**, procesa el interface, produciendo clases Stubs RMI y esqueletos. La clase, los stubs, y los esqueletos remotos están instalados en servidor WebLogic.

Un cliente Java busca un objeto remotro en servidor WebLogic usando JNDI (Java Naming and Directory Interface), que se describe más adelante en esta sección. JNDI establece una conexión al servidor WebLogic, busca la clase remota , y devuelve los stubs al cliente.

El cliente ejecuta un método del stub como si ejecutara el método directamente en la clase remota. El método del stub prepara la llamada y la transmite sobre la red a la clase esqueleto en el servidor WebLogic.

Sobre el servidor WebLogic, la clase esqueleto desempaqueta la petición y ejecuta el método sobre el objeto del lado del servidor. Después empaqueta los resultados y los devuelve al stub en la lado del cliente. WebLogic EJB y varios otros servicios disponibles para los clientes Java se construye en RMI. La mayoría de las aplicaciones deben utilizar EJB en vez de usar RMI directamente, porque EJB proporciona una abstracción mejor para los objetos del negocio. Además, el envase del servidor EJB de WebLogic proporciona a mejroas como almacenamiento en memoria inmediata, persistencia, y el control del ciclo vital que no estén automáticamente disponibles para las clases remotas.

- **RMI-IIOP**

La Invocación Remotra de Métodos sobre Inter-ORBProtocol Internet (RMI-IIOP) es un protocolo que permite que los programas clientes CORBA ejecuten objetos de WebLogic RMI, incluyendo beans enterprise. RMI-IIOP se basa en dos especificaciones del "Object Manegement Group" (http://www.omg.com):

- Mapeo Java-a-IDL.
- Objetos-por-valor.

La especificación Java-a-IDL define cómo un IDL se deriva de un interface Java. Los compiladores del servidor WebLogic para RMI y EJB le dan la opción de producir IDL al compilar objetos RMI y EJB. Este IDL se puede compilar con un compilador de IDL para producir los stubs requeridos por un cliente CORBA.

La especificación del objeto-por-valor define cómo los tipos de datos complejos se mapean entre Java y CORBA. Para utilizar objeto-por-valor, un cliente CORBA debe

utilizar un ORB (Object Request Broker) con soporte de CORBA 2.3. Sin un ORB CORBA 2.3 los clientes CORBA sólo pueden utilizar tipos primitivos de datos Java.

- **SSL**

El intercambio de datos con los protocolos HTTP y T3 se pueden encriptar con el SSL. Usando SSL asegura al cliente que ha conectado con un servidor autentificado y que los datos transmitidos sobre la red son privados. SSL utiliza el cifrado con clave pública, que requiere que compramos una ID de servidor, que es un certificado para nuestro servidor WebLogic, de una "Certificate Authority" como **VeriSign**. Cuando un cliente conecta puerto SSL del servidor WebLogic, el servidor y el cliente ejecutan un protocolo que incluya la autentificación de las ID del servidor y la negociación de los algoritmos y de los parámetros de cifrado para la sesión. El servidor WebLogic también se puede configurar para requerir que el cliente presente un certificado, un arreglo que se llama autentificación mutua.

- **Datos y Servicios de Acceso**

El servidor WebLogic implementa tecnologías estándars J2EE para proporcionar servicos de datos y de acceso a las aplicaciones y a los componentes. Estos servicios incluyen los siguientes APIs:

- Java Naming and Directory Interface (JNDI)
- Java Database Connectivity (JDBC)
- Java Transaction API (JTA)

Las secciones siguientes explican estos servicios más detalladamente.

- **JNDI**

JNDI es un API estándar de Java que permite a las aplicaciones buscar un objeto por su nombre. El servidor WebLogic une los objetos Java que sirve a un nombre en un árbol de nombres. Una aplicación puede buscar objetos, como objetos RMI, JavaBeans Enterprise, colas JMS, y JDBC DataSources, obteniendo un contexto JNDI del servidor WebLogic y después llamando el método de operaciones de búsqueda JNDI con el nombre del objeto. Las operaciones de búsqueda devuelven una referencia al servidor de objetos WebLogic.

WebLogic JNDI soporta el balance de carga del cluster WebLogic. Todo servidor WebLogic en un cluster publica los objetos que sirve en un árbol de nombres replicado en un amplio cluster. Una aplicación puede obtener un contexto inicial JNDI del servidor fromany WebLogic en el cluster, realizar operaciones de búsqueda, y recibir una referencia del objeto desde cualquier servidor del cluster WebLogic que sirve el objeto. Se utiliza un algoritmo de balance de carga configurable separar la carga de trabajo entre los servidores del cluster.

- **JDBC**

La conectividad de la base de datos JDBC Java (JDBC) proporciona acceso a los recursos **backend** de base de datos. Las aplicaciones Java tienen acceso a JDBC usando un driver JDBC, que es un interface específico del vendedor de la base de

datos para un servidor de base de datos. Aunque cualquier aplicación Java puede cargar un driver JDBC, conectar con la base de datos, y realizar operaciones de base de datos, el servidor WebLogic proporciona una ventaja de rendimiento significativa ofreciendo almacenes de conexión JDBC.

Un almacen de conexiones JDBC es un grupo se conexiones JDBC con nombre manejadas a través del servidor WebLogic. En el momento de arranque el servidor WebLogic abre conexiones JDBC y las agrega al almacen. Cuando una aplicación requiere una conexión JDBC, consigue una conexión del almacen, la utiliza, y luego la devulve al almacen para su uso por otras aplicaciones. Establecer una conexión con la base de datos es a menudo una operación que consume mucho tiempo, y recrusos, un almacen de conexiones, que limita el número de operaciones de la conexión, mejora su funcionamiento.

Para registar un almacen de conexiones en el árbol de nombrado JNDIl, definimos un objeto DataSource para él. Las aplicaciones clientes Java pueden entonces conseguir una conexión del almacen realizando operaciones de búsqueda JNDI con el nombre del DataSource.

las clases de Java del lado del Servidor utilizan el driver de conexiones JDBC de WebLogic JDBC, que es un driver genérico de JDBC que llama a través al driver específico del vendedr al driver JDBC. Este mecanismo hace el código de la aplicación más portable, incluso si cambiamos la marca de la base de datos usada en la grada **backend**.

El driver JDBC del lado del cliente es el driver de WebLogic JDBC/RMI, que es un interface RMI al driver del alamacen. Utilizamos este driver de la misma manera que utilizamos cualquier driver JDBC estándar. Cuando se utiliza el driver JDBC/RMI, los programas Java pueden tener acceso a JDBC de una manera consistente con otros objetos distribuidos en el servidor WebLogic, y pueden mantener las estructuras de datos de la base de datos en la capa media.

WebLogic EJB y WebLogic JMS tatan con conexiones de un almacen de conexiones JDBC para cargar y salvar objetos persistentes. Usando EJB y JMS, podemos conseguir a menudo una abstracción más útil de la que podemos obtener usando JDBC directamente en una aplicación. Por ejemplo, usar un bean enterprise para representar un objeto de datos permite que cambiemmos el almacén subyacente más adelante sin la modificación del código JDBC. Si utilizamos mensajes persistentes JMS en vez de operaciones de base de datos con JDBC, será más fácil adaptar la aplicación a un sistema de mensajería de una tercera persona más adelante.

- **JTA**

El API de transacción de Java (JTA) es el interface estándar para el manejo de transacciones en las aplicaciones Java. Usando transacciones, podemos proteger la integridad de los datos en nuestras bases de datos y manejar el acceso a esos datos por aplicaciones o ejemplares simultáneos de la aplicación. Una vez que una transacción comienza, todas las operaciones transaccionales deben terminar con éxito o todas se deben deshacer.

El servidor WebLogic utiliza las transacciones que incluyen operaciones EJB, JMS, y JDBC. Las transacciones distribuidas, coordinadas con dos fases, pueden espandir múltiples bases de datos que son accedidas con los drivers XA-compliant JDBC, como BEA WebLogic jDriver para Oracle/XA.

La especificación EJB define transaciones controladas por el bean y por el contenedor. Cuando se desarrolla un bean con transaciones controladas por el contenedor, el servidor WebLogic coordina la transacción automáticamente. Si se despliega un bean enterprise con transacciones manejadas por el bean, el programador de EJB debe proporcionar un código de transacción.

El código de aplicación basado en los APIs JMS o JDBC puede iniciar una transacción, o participar en una transacción comenzada anteriormente. Un solo contexto de transacción se asocia con el thread de ejecución de WebLogic Server, que ejecuta una aplicación; todas las operaciones transaccionales realizadas en el thread participan en la transacción actual.

- **Tecnologías de Mensajería**

Las tecnologías de mensajería J2EE proporcionan un APIs estándar que las aplicaciones del servidor WebLogic pueden utilizar para comunicarse con una otra, así como con aplicaciones de servidores no-WebLogic. Los servicios de mensajería incluyen los siguientes APIs:

- Java Message Service (JMS)
- JavaMail

Las secciones siguientes describen estos APIs con más detalle:

- **JMS**

El servicio de mensajería JMS de Java (JMS) permite a las aplicaciones comunicarse unas con otra intercambiando mensajes. Un mensaje es una petición, un informe, y/o un evento que contiene la información necesaria para coordinar la comunicación entre diversas aplicaciones. Un mensaje proporciona un nivel de abstracción, permitiendo que separemos los detalles sobre el sistema de destino del código de la aplicación.

WebLogic JMS implementa dos modelos de mensajería: punto-a-punto (PTP) y publish/subscribe (pub/sub). El modelo PTP permite que cualquier número de remitentes envíe mensajes a una cola. Cada mensaje en la cola se entrega a un solo programa de lectura. El modelo pub/sub permite que cualquier número de remitentes envíe mensajes a un Topic. Cada mensaje en el Topic se envía a todos los programas de lectura con una suscripción al Topic. Los mensajes se pueden entregar a los programas de lectura síncrona o asíncronamente.

Los mensajes JMS pueden ser persistentes o no-persistentes. Los mensajes persistentes se salvan en una base de datos y no se pierden si se rearranca el servidor WebLogic. Los mensajes no-persistentes se pierden si se rearranca el servidor WebLogic. Los mensajes persistentes enviados a un Topic pueden conservarse hasta que todos los suscriptores interesados los hayan recibido.

JMS soporta varios tipos de mensaje que sonn útiles para diversos tipos de aplicaciones. El cuerpo de mensaje puede contener los texto arbitrario, secuencias del bytes, tipos de datos primitivos de Java, parejas de nombre/valor, objetos serializables Java, o contenido XML.

- **Javamail**

El servidor JavaMail WebLogic incluye implementación de referencia del Sun JavaMail. JavaMail permite que una aplicación cree mensajes de E-mail y los envíe a través de un servidor SMTP a la red.

BEA WebLogic: Introducción
Autor: BEA
Traductor: Juan Antonio Palos (Ozito)

- WebLogic Server como Servidor Web
  - Cómo funciona el servidor WebLogic como un Servidor Web
  - Características del Servidor Web
    - Hosting Virtual
    - Usar Configuraciones de Servidor Proxy
    - Balance de Carga
    - Control de Fallos

WebLogic Server como Servidor Web

El servidor WebLogic se puede utilizar como el servidor web primario para aplicaciones web avanzadas. Una aplicación Web de J2EE es una colección de páginas HTML o XML, de páginas JSP, de servlets, de clases Java, de applets, de imágenes, de ficheros multimedia, y de otros tipos de ficheros.

- **Cómo funciona el servidor WebLogic como un Servidor Web**

Una aplicación Web se ejecuta en el contenedor Web de un servidor web. En un entorno de servidor WebLogic, un servidor web es una entidad lógica, desplegada en uno o más servidores WebLogic en un cluster.

Los ficheros de una aplicación Web se graban en una estructura de directorios que, opcionalmente, puede empaquetarse en un solo fichero .war (Web ARchive) usando la utilidad **jar** de Java. Un conjunto de descriptores de despliegue XML definen los componentes y los parámetros de ejecución de una aplicación, como las configuraciones de seguridad. Los descriptores de despliegue permiten cambiar comportamientos durante la ejecución sin cambiar el contenido de los componentes de la aplicación Web, y hacen fácil desplegar la misma aplicación en varios servidores Web.

- **Características del Servidor Web**

Cuando se usa como un servidor web, WebLogic Server soporta las siguientes funcionalidades:

- Hosting Virtual.
- Soporte para configuraciones de servidores proxy
- Balance de Cargas
- Control de fallos

Esta sección describe cómo es soportada cada una de estas funciones por WebLogic Server.

Hosting Virtual

WebLogic Server soporta almacenamiento virtual, un arreglo que permite a un solo servidor WebLogic o a un Cluster WebLogic contener varios sitos Web. Cada servidor web virtual tiene su propio nombre de host, pero todos los servidores Web están mapeados en la DNS de la misma dirección IP del cluster. Cuando un cliente envía una petición HTTP a la dirección del cluster, se selecciona un servidor WebLogic para servir la petición. El nombre del servidor web se extrae de la cabecera de la petición HTTP y se mantiene en subsecuenes intercambios con el cliente para que el hostname virtual permanezca constante desde la perspectiva del cliente. Múltiples aplicaciones Web pueden desplegarse en un servidor WebLogic, y cada aplicación Web se puede mapear a un host virtual.

Usar Configuraciones de Servidor Proxy

WebLogic server se puede integrar con los servidores web existentes. Las peticiones pueden ser almacenadas desde un servidor WebLogic a otro servidor web o, usando un plug-in nativo provisto del servidor WebLogic, desde otro servidor web al servidor WebLogic. BEA proprociona los plug-ins para Apache Web Server, Netscape Enterprise Server, Microsoft Internet Information Server.

El uso de los servidores proxys entre clientes y un conjunto de servidores independientes WebLogic o de un cluster WebLogic permite realizar el balance de carga y el control de fallos para las peticiones Web. Para el cliente, solo parecerá un servidor web.

Balance de Carga

Podemos instalar varios servidores WebLogic detrás de un servidor proxy para acomodar grandes volúmenes de peticiones. El servidor proxy realiza el balance de cargas, distribuyendo las peticiones a través de los distintos servidores en la capa que hay detrás de él.

El servidor proxy puede ser un servidor WebLogic, o puede ser un servidor Apache, Netscape, o Microsoft. El servidor WebLogic incluye los plugs-in de código nativo para algunas plataformas que permitan estos servidores web de terceras partes a las peticiones del servidor proxy de WebLogic.

El servidor proxy se configura para redirigir ciertos tipos de peticiones a los servidores que hay detrás de él. Por ejemplo, un arreglo común es configurar el servidor proxy para manejar las peticiones para páginas HTML estáticas y redirigir los pedidos de servlets y páginas JSP a clusters WebLogic detrás del proxy.

Control de Fallos

Cuando un cliente web empieza una sesión servlet, el servidor proxy podría enviar las peticiones subsecuentes que son parte de la misma sesión a un servidor WebLogic distinto. El servidor WebLogic proporciona replicación de la sesión para asegurarse de que el estado de la sesión del cliente sigue estando disponible.

Hay dos tipos de réplica de sesión:

- Se puede usar la réplica de sesión JDBC con un cluster WebLogic o con un conjunto de servidores WebLogic independientes. No requiere la opción que CLustering del WebLogic Server.
- La réplica de sesión en-memoria requiere la opción de Clustering del WebLogic Server.

La réplica de sesión JDBC escribe datos de la sesión en una base de datos. Una vez que se haya comenzado una sesión, cualquier servidor WebLogic que seleccione el servidor proxy puede continuar la sesión recuperando los datos de la sesión desde la base de datos.

Cuando se despliega un Cluster WebLogic detrás de un servidor proxy, las sesiones de servlets se pueden replicar sobre la red a un servidor WebLogic secundario seleccionado por el cluster, para evitar la necesidad de acceder a la base de datos. La replicación en-memoria usa menos recursos y es mucho más rápida que la replicación de sesión JDBC, por eso es la mejor forma para proporcionar control de fallos para servlets cuando tenemos un Cluster WebLogic.

BEA WebLogic: Introducción
Autor: BEA
Traductor: Juan Antonio Palos (Ozito)

- Servicios de Seguridad
  - Autentificación
  - Autorización
  - Alternartivas y Reinos Personalizados
  - Encriptación

Servicios de Seguridad

WebLogic Server proporciona seguridad para las aplicaciones a través de un "**security realm**" (reino de seguridad). Un reino de la seguridad proporciona acceso a dos servicios:

- Un servicio de autentificación, que permite que el servidor WebLogic verifique la identidad de los usuarios.
- Un servicio de autorización, que controla el acceso de los usuarios a las aplicaciones.

Autentificación

Un reino tiene acceso a un almacén de usuarios y de grupos y puede autentificar a un usuario comprobando una credencial suminstrada por el usuario (generalmente una password) contra el nombre de usuario y la credencial en el almacén de seguridad. Los navegadores Web utilizan la autentificación solicitando un nombre de usuario y una password cuando un cliente web intenta acceder a un servicio protegido del servidor WebLogic. Otros clientes del servidor WebLogic proporcionan nombres de usuarios y credenciales programaticamente cuando establecen conexiones con el servidor WebLogic.

- **Autorización**

Los servicios de WebLogic Server se protegen con listas del control de acceso (ACLs). Una ACL es una lista de usuarios y grupos que están autorizados para acceder a un servicio. Una vez que se haya autentificado a un usuario, el servidor WebLogic consulta la ACL de un servicio antes de permitir que el usuario tenga acceso al servicio.

- **Alternartivas y Reinos Personalizados**

El reino de seguridad es enchufable--podemos utilizar el Fichero del reino por defecto, un reino alternativo suministrado con el servidor WebLogic, o podemos crear nuestro propio reino. El fichero de reino por defecto graba usuarios, passwords, grupos, y ACLs en un fichero cifrado. El fichero se maneja a través de la consola de administración.

El servidor WebLogic incluye reinos alternativos que acceden a un almacén externo de usuarios y grupos y a veces ACLs. Los reinos alternativos proporcionan estos sistemas externos de seguridad:

- Lightweight Directory Access Protocol (LDAP), Netscape Dirctory Server, Microsoft Site Server y Novell NDS.
- Servicio de Login de UNIX
- Dominio de Windows NT
- RDBMS, un reino que utiliza un sistema de base de datos para grabar usuarios, grupos, y ACLs.

Se puede desarrollar un reino personalizado para el uso en un servidor WebLogic implementando los interfaces Realm de WebLogic. El servidor WebLogic soporta reinos con un sistema de almacenamiento en memoria inmediata incorporado, para reducir al mínimo las llamadas al almacén externo. Cualquier característica que sea implementada reino personalizado cae dentro del fichero de reino por defecto. Por ejemplo, es común desarrollar aplicaciones personalizadas de un reino que use un almacén externo para los usuarios y los grupos, pero mantenga las ACLs en el fichero del reino.

- **Encriptación**

WebLogic Server soporta el protocolo Secure Sockets Layer (SSL), que protege los datos transferidos sobre un cable. SSL es el protocolo estándar para conexiones Web

seguras. El servidor WebLogic utiliza SSL en conexiones Web (HTTPS) y en los clientes independientes Java que usan servicios basados en RMI.

Para proporcionar SSL, primero debemos comprar una ID de servidor para nuestro servidor WebLogic a una "Certificate Authority", como **VeriSign** o **GTE Cybertrust**. Cuando un cliente establece una conexión segura, el servidor WebLogic envía su certificado al cliente, permitiendo que el cliente verifique que la conexión sea apropiada. Los clientes examinan el certificado para cerciorarse de que no ha expirado, que corresponde al servidor que lo envió, y que está firmado por una Certificate Authority reconocida. Después, el servidor y el cliente intercambian claves de cifrado y establecen una conexión segura.

WebLogic Server también puede ser configurado para requerir autentificación mutua, lo que requiere que un cliente envíe un certificado de cliente que el servidor deba validar antes de aceptar una conexión.

BEA WebLogic: Introducción
Autor: BEA
Traductor: Juan Antonio Palos (Ozito)

- Clusters WebLogic
  - Ventajas de Usar Clusters
  - Arquitectura de un Cluster
  - Cómo se Define un Cluster WebLogic en una Red
  - Servicios en Clusters

Clusters WebLogic

Un cluster WebLogic es un grupo de servidores WebLogic que funcionan juntos para proporcionar una poderosa y fiable plataforma de aplicación Web. Un cluster aparece ante sus clientes como un solo servidor pero es, de hecho, un grupo de servidores que actúa como uno sólo. Proporciona dos ventajas clave que no son proporcionadas por un solo servidor: escalabilidad y disponibilidad.

Ventajas de Usar Clusters

Los Clusters WebLogic traen escalabilidad y alta-disponibilidad a las aplicaciones J2EE de forma transparente para los desarrolladores de aplicaciones. La ventaja de la escalabilidad es que amplía la capacidad de la capa media más allá de un solo servidor WebLogic o de un solo ordenador. La única limitación en calidad de miembro del cluster es que todos los servidores WebLogic deben poder comunicarse mediante IP multicast. Los nuevos WebLogic Servers se pueden añadir a un cluster dinámicamente para aumentar la capacidad.

Un cluster WebLogic también garantiza la alta disponibilidad usando la redundancia de servidores múltiples para aislar a los clientes de los incidentes. El mismo servicio se puede proporcionar en múltiples servidores de un cluster. Si un servidor falla, otro puede asumir el control. La capacidad de hacer que un servidor en funcionamiento

asuma el control sobre un servidor que falla aumenta la disponibilidad de la aplicación a los clientes.

- **Arquitectura de un Cluster**

Un Cluster WebLogic consta de un número de servidores WebLogic desplegados en una red, coordinada con una combinación de Domain Name Service (DNS), replicación de árboles de nombrado JNDI, la réplica de los datos de la sesión, y WebLogic RMI. Los servidores proxy entre los clientes Web y el Cluster WebLogic coordinan los servicos de clustering para los servlets y las páginas JSP. Los servidores proxy pueden ser servidores WebLogic, o servidores de terceras partes; Netscape, Microsoft, o Apache, usados con un plug-in proporcionado por el servidor WebLogic.

Los clientes Web conectan con un cluster WebLogic dirigiendo peticiones a un servidor proxy. Los clientes Java basados en RMI conectan con un cluster WebLogic usando una dirección del cluster definida en la red.

El código del lado del Servidor también se beneficia del balance de carga y los servicios de control de fallos proporcionados por un cluster WebLogic. En aplicaciones J2EE, la mayoría del código de la aplicación se ejecuta en la capa media y puede utilizar los servicios distribuidos entre varios servidores WebLogic. Por ejemplo, un servlet que se ejecuta en el servidor **A** WebLogic podría utilizar un bean enterprise en el servidor **B** WebLogic y leer los mensajes de una cola JMS en el servidor **C**.

- **Cómo se Define un Cluster WebLogic en una Red**

A los servicios de un WebLogic Server se accede a través de DNS, el servicio de nombrado estándar para los recursos en red, incluyendo Internet. DNS asocia direcciones IP, como **170,0,20,1**, a nombres, como **mycomputer.mydomain.com** o **www.bea.com**. Cada servidor WebLogic se ejecuta en la red en una única dirección IP. Un cliente conecta con un servidor WebLogic codificando en una URL su nombre y el número de puerto donde está escuchando las conexiones.

Por ejemplo, a un servidor WebLogic que se ejecuta en un ordenador llamado **onyx**, configurado para escuchar en el puerto **7701**, se puede acceder con un navegador web usando la siguiente URL : **http://onyx:7701**. Para que esta conexión tenga éxito, el servidor de nombres de la red debe poder resolver el nombre **onyx** en el dominio local. Si el servidor de destino está en otro dominio sobre Internet, se debe proprocionar el nombre de dominio completo, por ejemplo **http://onyx.bea.com:7701**.

Una entrada adicional DNS mapea los nombres de todos los servidores WebLogic que participan en un cluster a un solo nombre del cluster. Los clientes conectan con el cluster usando el nombre del cluster o con un servidor proxy Web que dirija las peticiones hacia el cluster. Cuando DNS realiza operaciones de búsqueda en un nombre del cluster, devuelve una lista de todos los servidores que pertenecen al cluster. Un cliente selecciona el primer servidor de la lista y, si no consigue ninguna respuesta, lo intenta con el segundo servidor, trabajando de esa forma a través de la lista hasta que consigue una respuesta. DNS proporciona el servicio de balance de carga inicial que distribuye peticiones a través de los servidores del cluster. Cada DNS responde a operaciones de búsqueda con el nombre del cluster, rotando la lista de

servidores de uno en uno, de modo que cada servidor consigue eventualmente su turno.

Un router inteligente, un servidor proxy, un firewal, u otros software que operen sobre la red, podrían sobreescribir el DNS y seleccionar el servidor inicial basándose en la carga de la máquina, el tráfico de la red, u otro criterio de balance de carga dinámico.

La conexión inicial del servidor WebLogic proporciona el servicio de nombrado para el cliente. Busca el servicio pedido por el cliente y elige un servidor del cluster para manejar la petición, usando un algoritmo de balance de carga configurado en el servidor WebLogic.

Los servidores WebLogic de un cluster se comunican unos con otros usando IP multicast para replicar ciertas clases de información a todos los servidores del cluster. Se configrua una dirección común multicast para cada servidor del cluster. Cuando un servidor envía un mensaje a la dirección multicast del cluster, todos los servidores reciben el mensaje. Este proceso es mucho más eficiente que tener servidores enviando mensajes de punta a punta. Sin embargo, requiere que todos los servidores de un cluster estén en una red que soporte multicast. El multicast no trabaja sobre Internet, así que un cluster no puede atravesar Internet.

Para algunos servicios, el cluster selecciona los servidores WebLogic primarios y secundarios. Si el servidor primario WebLogic comienza a procesar una petición y después llega a ser inasequible, el servidor secundario puede asumir el control del proceso de la petición sin interrupción. El servidor primario replica su estado al servidor secundario usando una conexión de servidor-a-servidor.

La mayoría de los servicios se pueden desplegar en cualquier número de servidores WebLogic de un cluster. Mientras que se despliega cada servicio, el servidor WebLogic utiliza IP multicast para añadir el servicio a un árbol de nombrado de cluster-ancho. Cualquier servidor del cluster puede encontrar un servidor WebLogic para proporcionar un servicio dado buscando el servicio en el árbol de nombrado de cluster-ancho. Cuando más de un servidor pueden proporcionar un servicio, el cluster utiliza un algoritmo de balance de carga configurable para elegir un servidor.

- **Servicios en Clusters**

La mayoría de los servicios de WebLogic Server pueden ponerse en un cluster; es decir, pueden ser desplegados en un número ilimitado de servidores del cluster. El cluster selecciona el servidor WebLogic que proporcionará un servicio. Una vez que se haya seleccionado ese servidor y hayan sido ejemplarizados los objetos con estado en el servidor, fijan al cliente a ése servidor WebLogic hasta que hayan acabado con el servicio. Si un servidor WebLogic que recibe un objeto fijado falla, el cliente debe detectar el incidente y crear otro ejemplar en otro servidor del cluster. Para proporcionar un control de fallos más resistente, un cluster WebLogic evita fijar un objeto a un servidor a menos que absolutamente sea necesario. En algunos casos el cluster replica el estado del objeto en otro servidor de reserva para permitir el control de fallos para el servicio.

Las aplicaciones Web se pueden poner en clusters, según lo descrito anteriormente. Las sesiones de Servlet se replican en un servidor secundario, permitiendo que el cluster se recupere de un incidente de forma transparente.

Todos los JavaBeans enterprise se pueden poner en clusters. Pueden ser desplegados en un número ilimitado de servidores de un cluster WebLogic. Sin embargo, no todos los ejemplares de EJB pueden ponerse en un cluster. Una aplicación puede obtener el interface home de un EJB desde cualquier servidor donde se haya desplegado el bean, y puede utilizar ese interface home para crear ejemplares del bean. Si el servidor que proporciona al interface home falla, se puede extraer el interface home de otro servidor sin interrumpir la aplicación.

Algunos tipos de ejmplares de EJB, incluyendo los beans de sesión sin estado y los beans de entidad de sólo lectura, siempre pueden ponerse en clusters. Los beans de sesión con estado pueden ponerse en clusters usando la réplica en-memoria para proporcionar al control de fallos. Los beans de entidad de lectura-escritura se fijan siempre al servidor donde están ejemplarizados. Si el servidor que recibe un bean de entidad de lectura-escritura falla, la aplicación debe crear un nuevo ejemplar.

Un "metapool" JDBC proporciona un cluster para almacenes de conexiones JDBC desplegadas en los múltiples servidores de un cluster WebLogic. Cuando un cliente solicita una conexión del metapool, el cluster selecciona el servidor que proporcionará la conexión, permitiendo el balance de cargas y la protección contra incidentes del servidor. Una vez que un cliente tiene una conexión, el estado mantenido por el driver de JDBC hace necesario fijar el cliente al servidor WebLogic.

Los objetos JMS pueden distribuirse entre los servidores de un cluster. Cada destino (cola de mensajes o Topic) es controlado por un sólo servidor WebLogic del cluster. Pero las factorías de conexiones, cuyos clientes se usan para establecer conexiones con un destino, pueden desplegarse en varios servidores de un cluster. Distribuyendo los detinos y las factorías de conexiones a lo largo de un cluster, los administradores pueden manualmente hacer un balance de carga de los servicos JMS.

BEA WebLogic: Introducción
Autor: BEA
Traductor: Juan Antonio Palos (Ozito)

- Control del Servidor y Monitorización
    - Administrator Server
    - Consola de Administración

Control del Servidor y Monitorización

La administración del servidor WebLogic se lograda fijando los atributos para los servidores en un dominio, usando la Consola de Administración o el interface de línea de comandos. La consola de administración es una aplicación de navegador web que nos permite configurar los servicios del servidor WebLogic, manejar la seguridad, desplegar aplicaciones, y vigilar los servicios dinámicamente.

La consola de administración y el interface de la línea de comandos conectan con el Administration Server.

- **Administrator Server**

El servidor de administración es el servidor de WebLogic usado para configurar y para manejar todos los servidores WebLogic en su dominio. Un dominio puede incluir múltiples clusters WebLogic y servidores WebLogic independientes. Si un dominio sólo contiene un servidor WebLogic, entonces ese servidor es el servidor de administración. En un dominio con varios ejemplares del servidor WebLogic, el primer ejemplar que arranca debe ser el servidor de administración.

- **Consola de Administración**

La consola del servidor de administración WebLogic se ejecuta en un navegador web. Muestra los componentes del dominio que administra, incluyendo clusters y servidores WebLogic independientes, en un árbol gráfico en el panel izquierdo. En el panel derecho visualiza los detalles sobre el objeto seleccionado en el panel izquierdo. La Figura 2-1 es una imagen tomada de una sesión de la consola de administración.

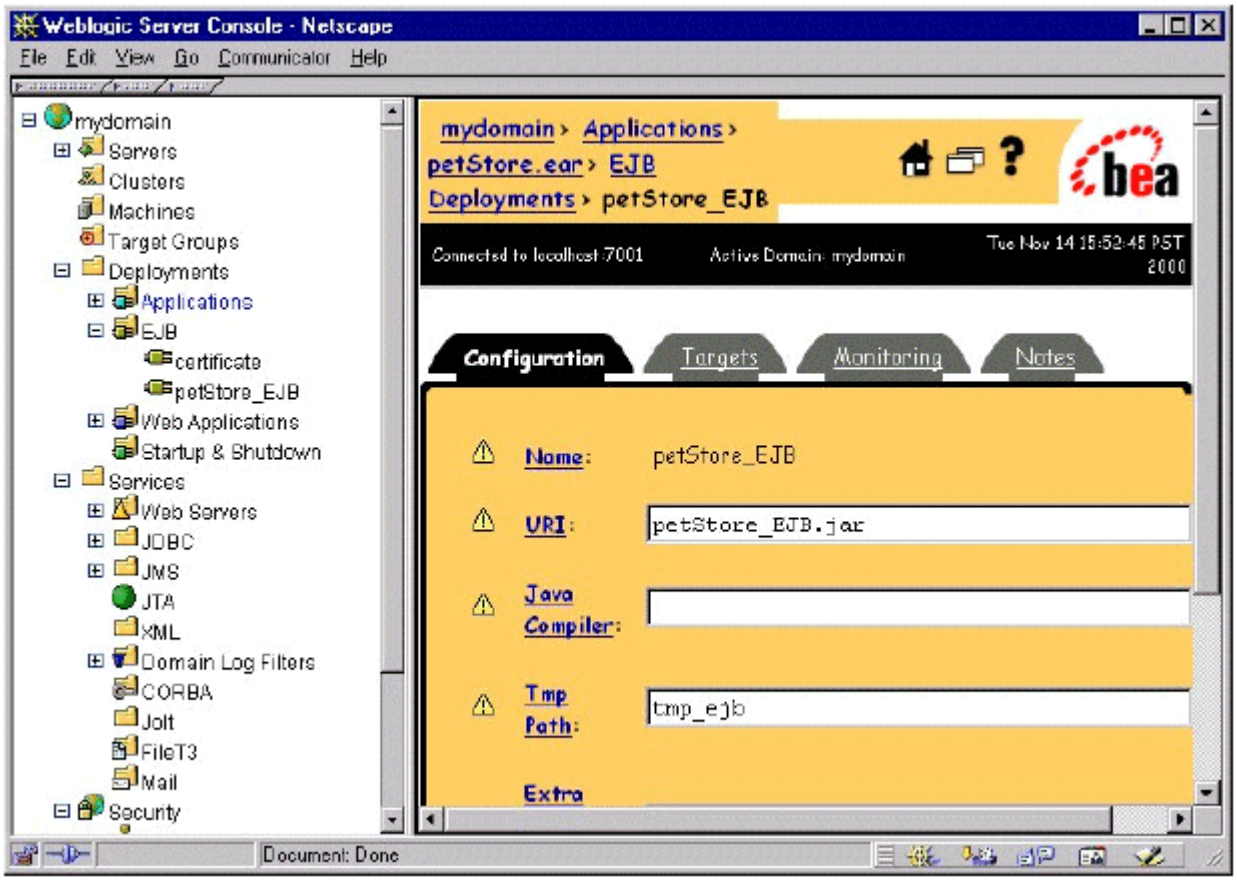


Figura 2-1 La Consola de administración

Para utilizar la consola de administración para configurar un servicio, seleccionamos un item en el panel izquierdo, y después elegimos la pestaña de configuración en el panel derecho. Las consola de administración muestra los atributos configurables en el panel derecho. Podemos utilizar la ayuda en línea para encontrar información detallada sobre los atributos visualizados.

El proceso usual para configurar un servicio en la consola de administración es configurar el servicio y después seleccionar los objetivos (los servidores WebLogic) en los cuales deseamos desplegar el servicio.

Cada servicio desplegado guarda estadísticas de ejecución, que podemos visualizar en la pestaña "Monitorig" del panel derecho de la consola de administración.