OBCOM MetaServer
Instalación y Configuración

Versión 1.2

OBCOM INGENIERÍA S.A.
Av. Holanda 1555, Of. 1204
Providencia, Santiago, Chile
Teléfono: +562 2761-7459
http://www.obcom.cl

Contenido

1 Introducción

El programa **EcuLink** permite que programas **Clientes** obtengan servicios de programas **Servidores** que ejecutan en el backend de una organización. Lo típico es que los programas **Clientes** ejecuten en estaciones de trabajo distintas, mientras que en el backend, el **EcuLink** y los **Servidores** ejecutan todos en un mismo computador, cada uno en un proceso independiente del sistema operativo.

Los procesos de un sistema operativo son estructuras relativamente pesadas, que requieren de recursos (CPU, Memoria) para administrar su seguridad, paginación, comunicaciones, y otros. Cuando el número programas **Servidores** es pequeño, no hay problema en tener cada uno en un proceso independiente. Por otra parte, si el número de **Servidores** requerido es grande (muchas decenas, o centenares de programas), utilizar procesos independientes obliga al sistema operativo a consumir una cantidad de recursos muy grande. Esto genera degradación en la calidad de los servicios, debido a que el sistema entra en un estado que se denomina Trashing: el trabajo necesario para administrar los procesos es mucho mayor que el trabajo que realizan dichos procesos.

Cuando el número de **Servidores** es grande, lo recomendable es utilizar hebras para ejecutar la lógica de estos servidores. Una hebra es una estructura liviana, que ejecuta dentro de un proceso del sistema operativo, y que comparte recursos con las otras hebras del mismo proceso. Esta característica hace que un conjunto de hebras consuma mucho menos recursos que un número igual de procesos independientes. Por esta razón, todos los Application Servers del mercado utilizan hebras para poder alcanzar los niveles de rendimiento necesarios en organizaciones con muchos clientes y/o servicios.

El programa **MetaServer** es un Application Server que permite ejecutar programas **Servidores** del **EcuLink** en hebras dentro de un proceso independiente. Esto hace un mejor uso de los recursos de un computador, lo cual permite tener muchos más programas **Servidores** ejecutando que si se utilizaran procesos independientes.

Como todo Application Server, el programa **MetaServer** permite descargar una versión de un programa **Servidor**, y reemplazarla por otra distinta. Todo esto, sin necesidad de detener el proceso principal, ni detener ninguna de las otras hebras en ejecución. Adicionalmente, el programa **MetaServer** implementa las siguientes funcionalidades:

- Configuración integrada y común del logging estándar de Java.
- Integración con sistema JMX de administración estándar de Java.
- Cargar bibliotecas Java (jars) en base a patrones de nombres.

El programa **MetaServer** está escrito 100% en Java 1.6 de Oracle, y puede utilizarse en cualquier sistema operativo donde esté disponible esta versión de Java. El programa **MetaServer** sólo puede ejecutar programas Servidores escritos en Java.

El programa **MetaServer** tiene una fuerte integración con el servidor **EcuLink**, y requiere la versión 4.1.1.130 o superior de este producto. Además, se necesita la biblioteca de comunicaciones **EcuLink.jar** versión 1.54 o superior.

2 Instalación

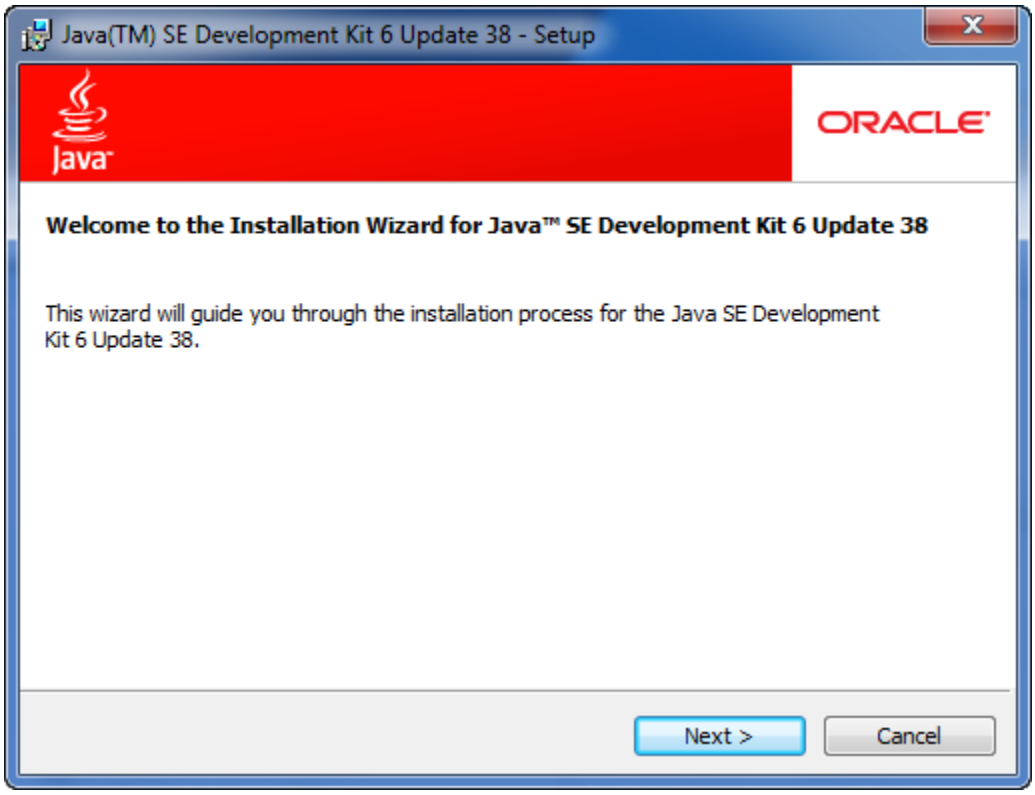
Como se mencionó anteriormente, el programa **MetaServer** está escrito en Java 1.6. Por consiguiente, el primer paso del proceso de instalación consiste en verificar que el sistema tenga disponible una versión adecuada de **Java**. Una vez hecho esto, se procede a instalar y configurar el programa **MetaServer**, propiamente tal.

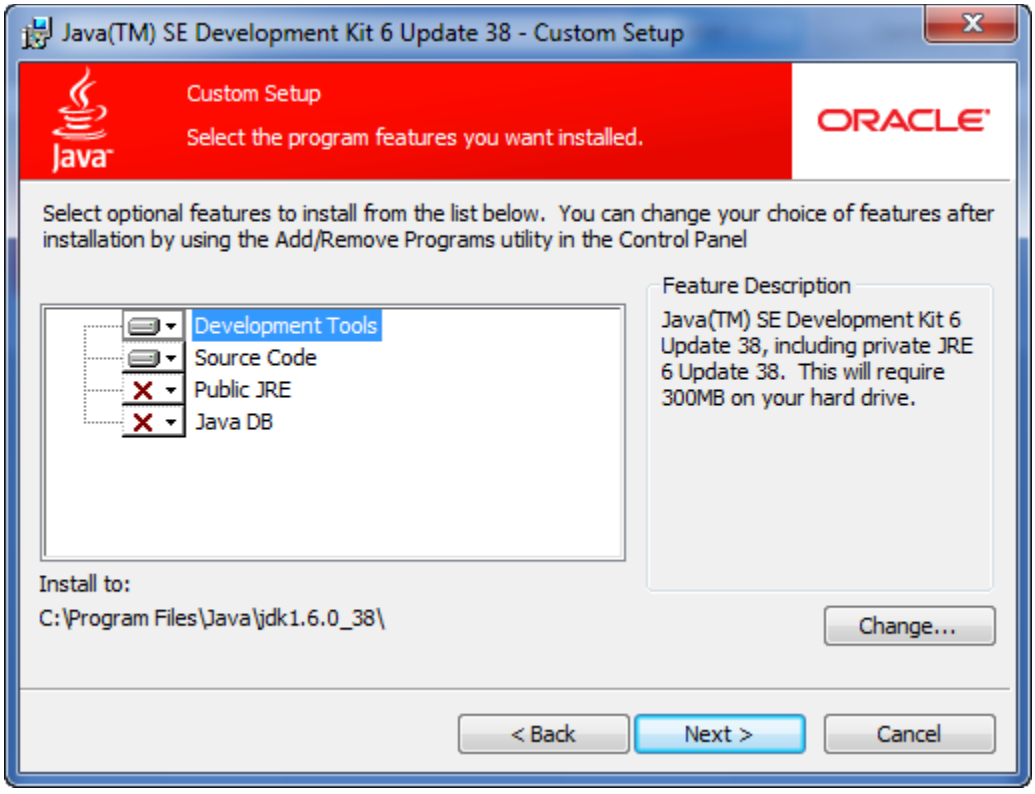
2.1 Instalación de Java

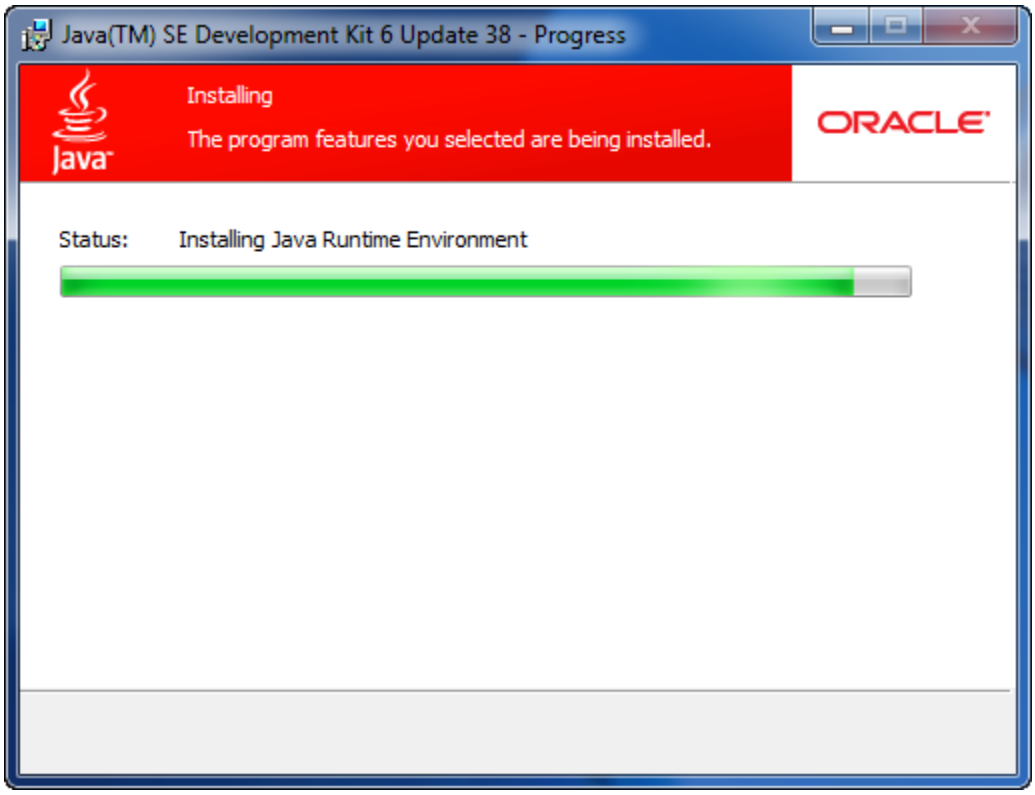
Existen dos versiones disponibles de Java de Oracle: la versión liviana denominada "**Java Runtime Environment**" (JRE), y la versión completa denominada "**Java SE Development Kit**" (JDK). El programa **MetaServer** puede ser utilizada con cualquiera de estas dos versiones, pero se recomienda utilizar la **JDK**, debido a que incluye un intérprete de Java optimizado para programas servidores. El instalador del **JDK** se puede descargar de la siguiente URL:

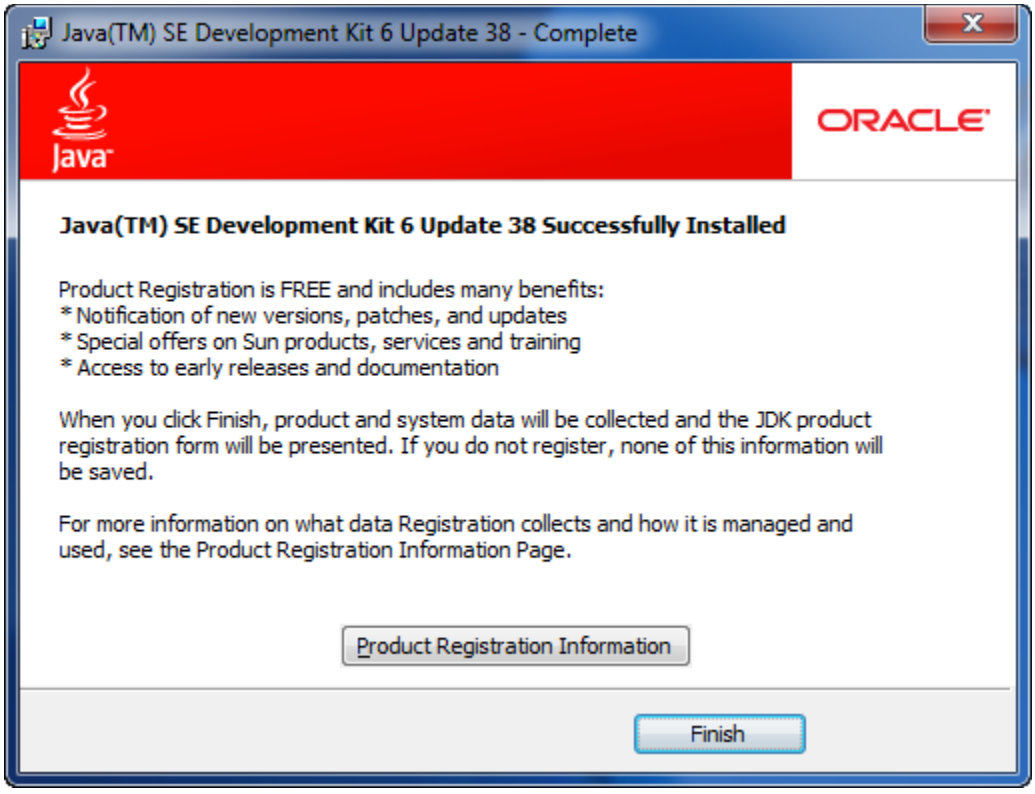
http://www.oracle.com/technetwork/java/javase/downloads/index.html

La última versión disponible del **JDK** es la **1.6.0_38** (Java 6 Update 38), y el programa instalador para la plataforma Windows de 32 bits se llama: **jdk-6u38-windows-i586.exe**. Al ejecutar este programa, se utiliza la siguiente secuencia de ventanas para instalar el **JDK**:


Java(TM) SE Development Kit 6 Update 38 - Setup
Java
ORACLE
Welcome to the Installation Wizard for Java™ SE Development Kit 6 Update 38
This wizard will guide you through the installation process for the Java SE Development Kit 6 Update 38.
Next >
Cancel


Java(TM) SE Development Kit 6 Update 38 - Custom Setup
Java
Custom Setup
Select the program features you want installed.
ORACLE
Select optional features to install from the list below. You can change your choice of features after installation by using the Add/Remove Programs utility in the Control Panel
Development Tools
Source Code
Public JRE
Java DB
Feature Description
Java(TM) SE Development Kit 6 Update 38, including private JRE 6 Update 38. This will require 300MB on your hard drive.
Install to:
C:\Program Files\Java\jdk1.6.0_38\
Change...
< Back
Next >
Cancel

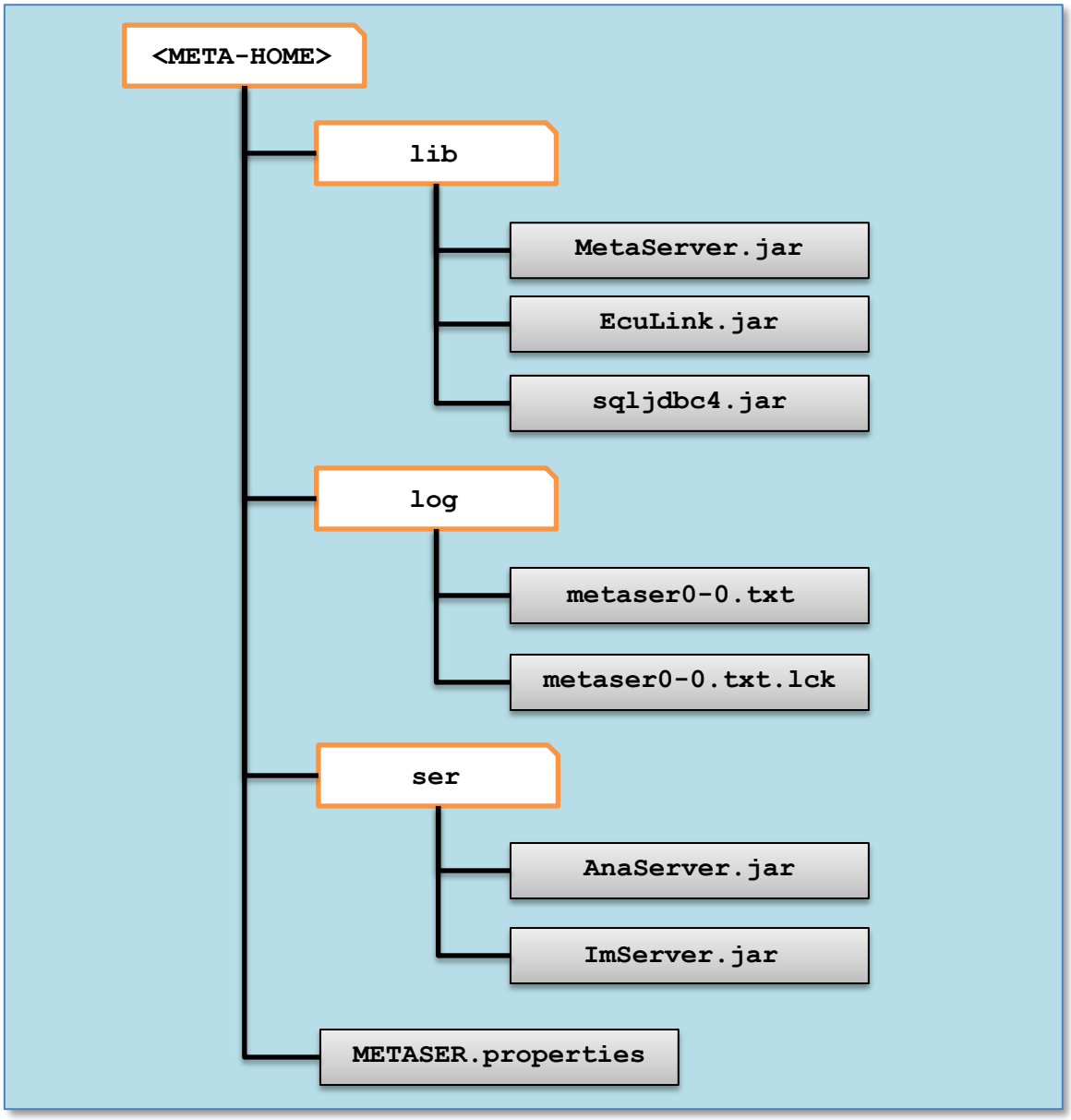

Java(TM) SE Development Kit 6 Update 38 - Progress
Installing
The program features you selected are being installed.
ORACLE
Status: Installing Java Runtime Environment


Java(TM) SE Development Kit 6 Update 38 - Complete
ORACLE
Java(TM) SE Development Kit 6 Update 38 Successfully Installed
Product Registration is FREE and includes many benefits:
* Notification of new versions, patches, and updates
* Special offers on Sun products, services and training
* Access to early releases and documentation
When you click Finish, product and system data will be collected and the JDK product registration form will be presented. If you do not register, none of this information will be saved.
For more information on what data Registration collects and how it is managed and used, see the Product Registration Information Page.
Product Registration Information
Finish

En este proceso, el **JDK** se instaló en el directorio "**C:\Program Files\Java\jdk1.6.0_38**", el cual se especificó en la segunda ventana de instalación. Este directorio se denomina **<JDK-HOME>**, y se usará más adelante para configurar el servidor **EcuLink** y el programa **MetaServer**.

2.2 Instalación del MetaServer

Una vez instalado el **JDK**, el próximo paso consiste en instalar el **MetaServer**. La instalación es muy simple: se crea una estructura de directorio para el **MetaServer**, y luego se copia a esta estructura las componentes y archivos de configuración del programa. La estructura de directorio recomendada para el MetaServer se muestra en la siguiente figura:



Como se observa, el directorio del programa **MetaServer** es denominado **<META-HOME>**. Este directorio posee tres subdirectorios, llamados **lib**, **log** y **ser**. También contiene un archivo llamado **METASER.properties**, el cual contiene la configuración del programa **MetaServer**. Este archivo de configuración se explica en la sección 3.3.

El subdirectorio **lib** contiene las bibliotecas Java utilizadas por todos los **Servidores**. Se puede ver el archivo **MetaServer.jar**, el cual que corresponde al programa **MetaServer**. También se ve el archivo **EcuLink.jar**, el cual que corresponden a la biblioteca de comunicaciones con el servidor **EcuLink**. Por último, se observa el archivo **sqljdbc4.jar**, el cual que corresponde a la biblioteca de comunicaciones con el servidor de bases de datos **Microsoft SQL Server**.

En el subdirectorio **log** se mantienen los archivos de registro (logging) generados por el programa **MetaServer** y los programas **Servidores**. En la figura se muestran los archivos de ejemplo llamados **metaser0-0.txt** y **metaser0-0.txt.lck**. El primero corresponde al archivo de registro, propiamente tal. El segundo archivo con extensión "**.lck**" es el archivo de bloqueo (locking) que utiliza el sistema de registro (logging) de Java.

Finalmente, el subdirectorio **ser** contiene las bibliotecas Java de los programas **Servidores**. En la figura se observan los archivos de ejemplo **AnaServer.jar** y **ImServer.jar**. **Importante**: las bibliotecas de los **Servidores** se deben colocar en el subdirectorio **ser**, y no en el subdirectorio **lib**. Como se verá más adelante, los **Servidores** se pueden actualizar a versiones más nuevas sin necesidad de detener el programa **MetaServer**, siempre y cuando ellos se encuentren instalados en el subdirectorio **ser**.

3 Configuración

El programa **MetaServer** es, en cierta forma, una extensión a la funcionalidad del servidor **EcuLink**, y mantiene una estrecha relación con este último. Por este motivo, la configuración del programa **MetaServer** involucra definir parámetros del servidor **EcuLink**, y definir parámetros específicos al **MetaServer**.

3.1 Configurar un MetaServer en EcuLink

El programa servidor **EcuLink** se configura utilizando un archivo llamado **Start.EcuLink** (también llamado **EcuLink.ini**). Para que **EcuLink** utilice un programa **MetaServer** llamado **METASER**, es necesario agregar las siguientes 5 líneas al archivo de configuración:

```
1: server     METASER, new
2:     server METASER, ismetaserver 1
3:     server METASER, file "<JDK-HOME>\bin\javaw.exe"
4:     server METASER, directory "<META-HOME>"
5:     server METASER, parameters "<JAVA-PARAMS>"
```

El significado de estas líneas es el siguiente:

1. Define un nuevo servidor llamado **METASER**.
2. Indica que el servidor **METASER** es un **MetaServer**.
3. Define el intérprete de Java que ejecutará al **MetaServer** (leer sección 2.1).
4. Define el directorio raíz (HOME) del **MetaServer** (leer sección 2.2).
5. Define parámetros de inicio del **MetaServer** (leer a continuación).

La línea 5 define un conjunto de parámetros denominados **<JAVA-PARAMS>**, los cuales son utilizados por Java para ejecutar el programa **MetaServer**. Algunos de estos parámetros son opcionales y otros son obligatorios. Los parámetros obligatorios son los siguientes:

```
-cp lib\*;lib cl.obcom.metaserver.MetaServer
```

a. **`-cp`**: define el **class path** de Java, que corresponde a una lista de archivos o directorios separados por "**;**" (punto y coma). Java busca en esta lista las clases que necesita ejecutar. En este caso, el valor "**`lib\*;lib`**" indica a Java que busque en todos los archivos con extensión "**.jar**" del subdirectorio "**lib**" del directorio **<META-HOME>**.
b. **`cl.obcom.metaserver.MetaServer`**: indica el nombre de la clase que Java debe comenzar a ejecutar. Esta es la clase principal del programa **MetaServer**.

Los parámetros opcionales que se pueden agregar a la línea 5 son los siguientes:

a. **`-server`**: indica que se utilice un intérprete optimizado para ejecutar programas de tipo servidor; es decir, programas que se ejecutan durante tiempos prolongados, y que hacen uso importante de memoria, hebras o CPU.
b. **`-Xmx256m`**: indica la cantidad máxima de memoria que puede consumir Java. En este caso, se indica un máximo de 256 MB (megabytes). Se recomienda usar valores de 512 megabytes como mínimo, y no especificar valores de más de 1200 megabytes cuando se usan intérpretes de Java de 32 bits.
c. **`-Dcom.sun.management.config.file=METASER.properties`**: define el archivo de configuración del sistema de monitoreo JMX de Java. En este caso, se indica que el archivo "**`METASER.properties`**" contiene los parámetros JMX. Una vez configurado JMX, la herramienta "**<JDK-HOME>\bin\jvisualvm.exe**" puede conectarse a un programa **MetaServer**, y mostrar gráficos con los valores de distintos parámetros, tales como uso de memoria, consumo de CPU, cantidad de hebras, etc.

Importante: el nombre de la clase principal (**cl.obcom.metaserver.MetaServer**) debe siempre estar escrita al final de la lista de parámetros de Java. Si se agregan parámetros opcionales, ellos deben escribirse antes de este nombre.

3.2 Configurar un Servidor Java en EcuLink

Una vez que se ha definido un **MetaServer** en **EcuLink**, es posible especificar qué programas **Servidores** Java serán ejecutados en hebras de este **MetaServer**. Para que **EcuLink** asocie un programa **Servidor** Java llamado **IMSER** con un **MetaServer** llamado **METASER**, es necesario agregar (como mínimo) las siguientes líneas al archivo de configuración de **EcuLink**:

```
1: server      IMSER, new
2:      server IMSER, metaserver "METASER"
3:      server IMSER, file "cl.obcom.ImServer@ser\ImServer.jar"
```

```
4:      server IMSER, parameters "<IMSER-PARAMS>"
```

El significado de estas líneas es el siguiente:

1. Define un nuevo servidor llamado **IMSER**.
2. Asocia el servidor con el **MetaServer** llamado **METASER** (leer sección 3.1).
3. Define la clase principal del programa **IMSER** y su **class path** (leer a continuación).
4. Define parámetros de partida específicos de este servidor.

La línea 3 específica el nombre de la clase principal del programa servidor, y este nombre es obligatorio. Opcionalmente, esta línea puede incluir un **class path** para este servidor, el cual debe escribirse después del nombre de la clase separador por un "@". El **class path** especifica una lista de archivos y/o directorios separados por ";" (punto y coma), la cual usa el programa **MetaServer** para encontrar la clase principal del programa servidor indicado.

En el ejemplo anterior, la línea 3 indica un **class path** de "**`ser\ImServer.jar`**". Con esto, el programa **MetaServer** buscará la clase principal del servidor **IMSER** en el archivo "**ImServer.jar**" en el subdirectorio "**ser**" del directorio "**`<META-HOME>`**". Si no encuentra la clase en este archivo, entonces buscará en el **class path** que se definió con el parámetro "**`-cp`**" (ver sección 3.1).

3.3 Configuración del MetaServer

Los parámetros específicos del programa **MetaServer** se configuran en un archivo separado del archivo de configuración del servidor **EcuLink**. Este archivo se encuentra normalmente en el directorio **`<META-HOME>`**, y tiene por nombre **`<NAME>.properties`**, donde **`<NAME>`** es el nombre definido para el **MetaServer** dentro del archivo de configuración del **EcuLink** (por ejemplo, METASER).

3.3.1 Parámetros relacionados con EcuLink

Los parámetros relacionados con **EcuLink** son los siguientes:

```
1: ecu.server = ECUSERVER
2: layouts.url = file:///C:/EcuLink/Layouts/*.asp
```

El significado de estas líneas es el siguiente:

1. **`ecu.server`**: define la dirección IP y puerto del servidor **EcuLink**. Alternativamente, define el nombre de una variable de ambiente que entrega esta información. En este ejemplo, se utiliza la variable de ambiente "**`ECUSERVER`**", la cual está definida dentro del archivo de configuración del **EcuLink**.
2. **`layouts.url`**: define la URL para obtener las plantillas (también llamadas Records o Layouts) de los mensajes que se envían y reciben del servidor **EcuLink**. En este ejemplo, se indica que las plantillas están en el directorio local "**`C:/EcuLink/Layouts`**".

3.3.2 Parámetros relacionados con JMX

Los parámetros relacionados con **JMX** (Java Management Extensions) son los siguientes:

```
1: com.sun.management.jmxremote = true
2: com.sun.management.jmxremote.port = 10210
3: com.sun.management.jmxremote.ssl = false
4: com.sun.management.jmxremote.authenticate = false
```

El significado de estas líneas es el siguiente:

1. **`com.sun.management.jmxremote`**: activa (true) o desactiva (false) el mecanismo de administración JMX (Java Management Extensions).
2. **`com.sun.management.jmxremote.port`**: define el número del puerto que se debe utilizar para que la herramienta "**<JDK-HOME>\bin\jvisualvm.exe**" se puede conectar al programa **MetaServer**. **Importante**: cada instancia de **MetaServer** debe tener asignado un puerto único, de lo contrario Java generará un error y no partirá ejecutando.
3. **`com.sun.management.jmxremote.ssl`**: indica que se fuerce comunicación JMX encriptada con protocolo SSL. Para utilizar esta funcionalidad, es necesario definir otros parámetros relacionados con certificados digitales.
4. **`com.sun.management.jmxremote.authenticate`**: indica que se autentique con usuario y contraseña al quien intente conectarse vía JMX. Para utilizar esta funcionalidad, es necesario definir otros parámetros relacionados con repositorios de usuarios.

Para más información de estos parámetros referirse a la documentación oficial Java en:

http://download.oracle.com/javase/6/docs/technotes/guides/management/agent.html

3.3.3 Parámetros relacionados con Logging

Los parámetros relacionados con registro (logging) son los siguientes:

```
1: .level = INFO
2: handlers = java.util.logging.FileHandler
3: java.util.logging.FileHandler.level = INFO
4: java.util.logging.FileHandler.pattern = log/metaser%u-%g.txt
5: java.util.logging.FileHandler.limit = 200000
6: java.util.logging.FileHandler.count = 5
```

El significado de estas líneas es el siguiente:

1. **`.level`**: indica el nivel mínimo de severidad que será registrado. Lo niveles de severidad definidos por Java son SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST. En este ejemplo, por defecto sólo los niveles SEVERE, WARNING e INFO serán registrados. Mensajes con los otros niveles de severidad serán ignorados.
2. **`handlers`**: define una lista separada por "," (coma) de administradores de registros que se deben utilizar. En este ejemplo sólo se utiliza el administrador de registro asociado a archivos. Los parámetros de configuración de este administrador están en las líneas 3-6.

3. **`java.util.logging.FileHandler.level`**: indica el nivel mínimo de severidad que será registrado en los archivos. Normalmente este nivel y el nivel especificado en la línea 1 son el mismo.
4. **`java.util.logging.FileHandler.pattern`**: indica el directorio y nombre que tendrán los archivos de registro. En este ejemplo, los archivos de registro quedarán en el subdirectorio "**log**" del directorio "**`<META-HOME>`**". Los archivos de registro tendrán una extensión "**.txt**", y el nombre comenzará con el prefijo "**`metaser`**", el cual estará seguido de un número único "**`%u`**" y un número de secuencia "**`%g`**".
5. **`java.util.logging.FileHandler.limit`**: indica el tamaño máximo que puede alcanzar un archivo de registro antes de que se cree el próximo. En este ejemplo, se especifica un máximo de 200000 bytes.
6. **`java.util.logging.FileHandler.count`**: indica el número máximo de archivos de registro que puede crear Java. Una vez que se alcanza el máximo, se borra el archivo de registro más viejo.

Para más información de estos parámetros referirse a la documentación oficial Java en:

http://download.oracle.com/javase/6/docs/api/java/util/logging/FileHandler.html