Cosmos WebServer

Cosmos WebServer (CWS) es una utilidad para que Cosmos trabaje como proveedor de servicios web, permitiendo crear servicios REST.

Esta utilidad está disponible a partir de la versión 6.0 de Cosmos, excepto el fichero de Log: Cwslog.log, que se ha incorporado a partir de la versión 7.2.

A partir de la versión 7.4 de Cosmos, Cosmos WebServer admite también conexiones HTTPS.

A partir de la versión 7.4.2 de Cosmos, se puede indicar la versión de la máquina virtual de Java con variables de entorno, sin necesidad de modificar la variable PATH del sistema.

A partir de la versión 7.8.3 de Cosmos, se puede definir la codificación de caracteres en la que trabajan los servicios definidos.

A partir de la versión 8.0 de Cosmos, se puede modificar el número de «acceptors» y «selectors» de Jetty con una variable de entorno, instalar/iniciar/parar/desinstalar el servicio de manera silenciosa y en el fichero log4j, con la traza en modo DEBUG, se podrán consultar los parámetros que se envían en la petición.

A partir de la versión 8.4 de Cosmos, se puede definir el directorio de ubicación y el nombre del fichero de log «cwslog.log». Además, puede definirse un fichero por cada servicio.

Cosmos WebServer no funciona con licencias de tipo Cosmos SQL Desktop ni Cosmos SQL Workgroup.

Índice

1. Introducción

Cosmos WebServer (CWS) es una utilidad para que Cosmos trabaje como proveedor de servicios web, permitiendo crear servicios REST.

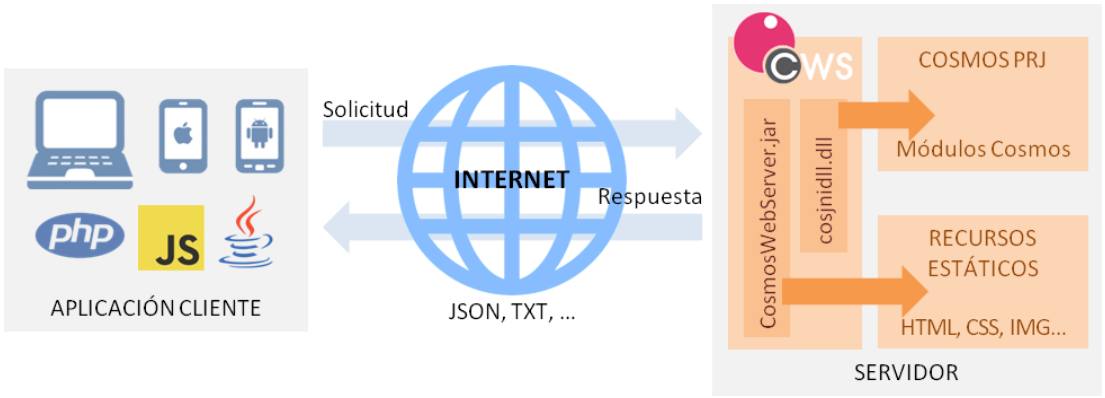
Cosmos WebServer se puede arrancar en línea de comando o bien como servicio Windows.

Una vez arrancado, el servidor atiende las peticiones HTTP en el puerto definido (por defecto 8081). Una petición puede convertirse en invocación a un método de una aplicación Cosmos configurada.

A continuación se detalla este funcionamiento.

2. Arquitectura

El esquema básico de Cosmos WebServer (CWS) es el que se muestra en la siguiente figura:



Una instalación Cosmos WebServer estará compuesta por:

- El servidor web (coswebserver.exe + cosmoswebserver.jar).
- El runtime de Cosmos embebido en una DLL (cosjnidll.dll).
- La aplicación Cosmos donde se definirán los módulos que implementan los servicios REST.

Cosmos WebServer utiliza la dll "JVM.dll" de la máquina virtual de Java. Por lo tanto, para ejecutar CWS es necesario disponer de una máquina virtual Java de 32 bits (con versión 1.8 o superior) instalada en la misma máquina que Cosmos WebServer.

A partir de la versión 7.4 de Cosmos, Cosmos WebServer admite también conexiones HTTPS.

La aplicación Cosmoswebserver.exe permite arrancar el servidor CWS desde la línea de comando o como servicio Windows.

3. Configuración

3.1 Fichero de configuración cosmoswebserver.ini

Este fichero incluye las siguientes funcionalidades:

a) Una entrada para cada servicio Windows que se vaya a crear.
b) Indica al servicio Cosmos WebServer la máquina virtual Java que debe utilizar (disponible a partir de la versión 7.4.2 de Cosmos).

Estas funcionalidades no son excluyentes entre sí.

Para instalar Cosmos WebServer como un servicio Windows es necesario crear un fichero con el nombre «cosmoswebserver.ini» que debe estar ubicado en "COSMOSDIR/etc".

Por cada servicio que se cree, se deberá definir una sección con el nombre del servicio. Cada sección contendrá una variable, llamada INIFILE, con el path absoluto del fichero de configuración para el servidor REST.

Ejemplo:

```
[ServicioWeb]
INIFILE=c:\cosmos\etc\stockserver.ini
[ServicioWebCompras]
INIFILE=c:\cosmos\etc\comprasserver.ini
```

Sección [Enviroment]

A partir de la versión 7.4.2 de Cosmos, es posible indicar a Cosmos WebServer la versión de la máquina virtual de Java que deberá utilizar. Para ello, será necesario definir en la sección [Environment] las variables de entorno CWSUSEJAVAVERSION o CWSUSELASTJAVAVERSION:

- **CWSUSEJAVAVERSION.** Indica a Cosmos WebServer la versión de Java que utilizará en su ejecución y en la de métodos Java desde el propio Cosmos WebServer.
- **CWSUSELASTJAVAVERSION.** Indica a Cosmos WebServer que utilice la versión más reciente de Java instalada en el PC.

Si Cosmos WebServer no se inicia como servicio, pero se desea indicar la máquina virtual de Java que se va a utilizar, también será necesario definir estas variables en *cosmoswebserver.ini*.

3.2 Fichero de configuración de la aplicación Cosmos WebServer

En el fichero de configuración de Cosmos WebServer se indicará, entre otras cosas, el puerto donde escuchará el servidor, la ruta del fichero de configuración de los servicios REST Cosmos, los parámetros propios de la máquina virtual Java (ruta del fichero cosmosrestserver.jar, parámetros de memoria) y, opcionalmente, la ruta de recursos HTML del servidor.

Ejemplo de fichero de configuración:

```
[Server]
CONFIG=c:\cosmosRestApp\etc\stockConfiguration.xml
PORT=8080
RESOURCEPATH=c:/cosmosRestApp/resources/webapp
CWSLOGFILE=c:\tmp\logs\CWSVentas.log
[JAVA]
-Xms256m
-Xmx1024m
-Djava.class.path=c:\\cosmos\\bin\\cosmoswebserver.jar;
-Dlog4j.configurationFile=c:/cosmos/etc/log4j2.xml
-Dcom.base100.cosmos.server.threadacceptorselector=10,100
```

3.2.1 Variables de entorno

CONFIG

Esta variable indica la ruta del fichero de configuración de los servicios REST del servidor (la configuración de este fichero se explica en el apartado “Fichero de configuración del servidor REST”).

PORT

Indica el puerto del servidor donde CWS escuchará peticiones HTTP.

RESOURCEPATH

Esta variable de entorno indica la ruta del directorio donde CWS almacenará recursos que serán accesibles mediante la URL del servidor. Estos recursos podrán ser páginas HTML, ficheros de imágenes, etc.

NUMBEROFTHREADS

Esta variable será necesario definirla para que la ejecución de Cosmos WebServer sea multihilo, es decir, que permita procesar más de una petición REST a la vez sin tener que esperar a que finalice una para procesar la siguiente.

Se define en la sección [Server] del fichero INI del servicio.

El valor de esta variable será el número de hilos que deberá ser capaz de manejar el servidor.

Para que Cosmos WebServer sea multihilo:

- Es obligatoria la definición de la variable.
- Su valor deber ser mayor de 1.
- Cosmos WebServer debe estar instalado como servicio y no como aplicación de consola.

Cuando Cosmos WebServer esté definido como multihilo, la conexión a la base de datos solo se podrá hacer en arquitectura cliente-servidor.

> *No se podrán establecer conexiones locales a la base de datos. En conexiones locales retornará el mensaje:*
> *“Imposible conectar con el servidor de SQL. (Unable to connect to local database from Cosmos WebServer in MULTI-THREAD MODE. The connection must be client-server)”.*

CWSLOGFILE

Permite definir la ruta u el nombre del fichero «cwslog.log». Se define en el fichero de configuración de Cosmos WebServer sección «[Server]».

Dcom.base100.cosmos.server.threadacceptorselector

Esta variable modifica el número de «acceptors» y «selectors» del Jetty. El número máximo de la suma de los 2 elementos no puede pasar de 180. Esta variable debe definirse en la sección [JAVA].

Parámetros:

Acceptors.	Hilos del proceso del Jetty que van a ser los encargados de aceptar peticiones HTTP en el puerto especificado en la configuración.
Selectors.	Hilos del proceso del Jetty que van a ser los encargados de gestionar la comunicación en las conexiones HTTP con el cliente.

Por cada conexión HTTP, habrá un hilo correspondiente a un selector.

Esta variable se ha implementado en la versión 8.0 de Cosmos. En versiones anteriores a la 8.0, si se ejecutaban muchos hilos simultáneamente, Cosmos WebServer llegaba a bloquearse.

Ejemplo:

```
-Dcom.base100.cosmos.server.threadacceptorselector=10,100
acceptors = 10
selectors = 100
```

Para configurar una conexión HTTPS será necesario definir las siguientes variables:

SSL_PORT

- Indica el puerto del servidor donde CWS escuchará peticiones HTTPS.
- Esta variable es obligatoria si se desea configurar una conexión HTTPS.
- Si no se definen las variables PORT y SSL_PORT, la conexión será HTTP (no segura) por el puerto 8081.
- Si se indica PORT y no se indica SSL_PORT, la conexión será HTTP (no segura) por el puerto indicado.
- Si no se indica PORT y sí SSL_PORT, la conexión utilizará HTTPS (conexión segura), pero no se permitirá hacer conexión HTTP (no segura).
- Si se indican PORT y SSL_PORT, las conexiones utilizarán HTTPS (conexión segura) y HTTP (conexión no segura) a los puertos indicados. Los números de los puertos indicados deberán ser diferentes.

SSL_KEYSTORE

- Indica la ruta del fichero de almacén de certificados.
- Se deberá utilizar el carácter “/” para separar nombres de directorios.
- Esta variable es obligatoria si se desea configurar una conexión HTTPS.

SSL_KEYSTORE_PWD

- Indica la contraseña del almacén de certificados.
- Esta variable es obligatoria si se desea configurar una conexión HTTPS.

SSL_KEYSTORE_ALIAS

- Indica el alias del certificado que se desea utilizar. Si no se indica, utilizará el primer certificado del almacén.
- Esta variable es opcional.
- Aunque el valor indicado en esta variable no sea válido, Cosmos WebServer se iniciará correctamente.

SSL_KEYSTORE_ALIAS_PWD

- Indica la contraseña del alias indicado en SSL_KEYSTORE_ALIAS. Si no se especifica, utilizará la contraseña indicada en SSL_KEYSTORE_PWD.
- Esta variable es opcional.

> *Si se produce un error en el proceso de arranque de Cosmos WebServer, el servicio no se iniciará y no se podrán realizar conexiones de ningún tipo. Por ejemplo, si la contraseña del almacén de certificados es incorrecta, o si el puerto de escucha HTTP o HTTPS indicado ya está en uso o si se indica el mismo puerto para las conexiones HTTP y HTTPS, se producirá un error en el proceso de arranque y no se podrán realizar conexiones con el servidor.*

3.3 Fichero de configuración del servidor REST

En este fichero de configuración se especifica, en formato XML, el nombre de la aplicación Cosmos, el nombre de los módulos y de los métodos que serán los encargados de implementar los servicios REST, así como la ruta (URL) y verbo HTTP (GET, POST, PUT, DELETE, etc.) que se deberá emplear para acceder a cada uno de estos métodos. Cada servicio REST en Cosmos se corresponderá con un método definido en las secciones <method>, a las que se accederá mediante su ruta y la del módulo <module> y proyecto <Project> al que pertenecen (propiedades path de <Project>, <module> y <method>). Cada método Cosmos correspondiente con un servicio REST deberá retornar un string con el resultado de la ejecución del servicio REST.

3.3.1 Ejemplo de fichero de configuración

A continuación se muestra un ejemplo de fichero de configuración de un servidor REST. En él se definen el proyecto, los módulos y los métodos Cosmos que implementarán los servicios REST, así como la manera de acceder a ellos:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
<project name="stock" path="stock/rest" dirPath="c:\cosmosRestApp\cosmosRestApp.PRJ">
        <module name="security" path="security">
            <method name="authenticate" path="authenticate" http="GET">
                <queryParameter name="username" defaultValue = ""/>
                <queryParameter name="password" defaultValue=""/>
                <queryParameter name="domain"/>
            </method>
        </module>
        <module name="states" path="states">
            <method name="resultList" path="resultList" http="POST">
                <queryParameter name="firstResult" defaultValue="0"/>
                <queryParameter name="maxResults" defaultValue="10"/>
                <queryParameter name="orderBy"/>
                <bodyParameter name="filter"/>
            </method>
            <method name="resultCount" path="resultCount" http="POST">
                <bodyParameter name="filter"/>
            </method>
            <method name="findByKey" path="{key}" http="GET">
                <urlParameter name="key"/>
            </method>
            <method name="deleteByKey" path="{key}" http="DELETE">
                <urlParameter name="key"/>
```

```
            </method>
            <method name="update" path="{key}" http="PUT">
                <urlParameter name="key"/>
                <bodyParameter name="entity"/>
            </method>
            <method name="create" http="POST">
                <bodyParameter name="entity"/>
            </method>
        </module>
        <module name="items" path="items">
            <method name="resultList" path="resultList" http="POST">
                <queryParameter name="firstResult" defaultValue="0"/>
                <queryParameter name="maxResults" defaultValue="10"/>
                <queryParameter name="orderBy"/>
                <bodyParameter name="filter"/>
            </method>
            <method name="resultCount" path="resultCount" http="POST">
                <bodyParameter name="filter"/>
            </method>
            <method name="findByKey" path="{item}/{supplier}" http="GET">
                <urlParameter name="item"/>
                <urlParameter name="supplier"/>
            </method>
            <method name="deleteByKey" path="{item}/{supplier}" http="DELETE">
                <urlParameter name="item"/>
                <urlParameter name="supplier"/>
            </method>
            <method name="update" path="{item}/{supplier}" http="PUT">
                <urlParameter name="item"/>
                <urlParameter name="supplier"/>
                <bodyParameter name="entity"/>
            </method>
            <method name="create" http="POST">
                <bodyParameter name="entity"/>
            </method>
        </module>
</project>

</configuration>
```

3.3.2 Secciones del fichero de configuración

Sección <project>

Dentro se está sección se definen las siguientes propiedades:

name	Identificador del proyecto en el servidor web.
dirPath	Ruta del fichero de proyecto Cosmos dentro del sistema de archivos del servidor.
path	Ruta desde la que se accede a los métodos del proyecto en la URL del servidor.

Sección <module>

Por cada módulo del proyecto Cosmos que defina un servicio REST se deberá definir una sección <module>. Las secciones <module> se deberán definir como secciones hijas de la sección <project>, una por cada módulo. En esta sección se definen las siguientes propiedades:

name	Nombre del módulo Cosmos.
path	Ruta de acceso en la URL del servidor.

Sección <method>

Por cada uno de los métodos del servidor REST definidos en un módulo se definirá una sección llamada <method> como sección hija de la sección <module> correspondiente. Se deberá definir una sección <method> por cada método del módulo que implemente un servicio web. En esta sección se definen las siguientes propiedades:

name	Nombre del método Cosmos.
path	Ruta de acceso en la URL del servidor.
http	El verbo http que activa el método.
parametersCharset	Codificación de caracteres del parámetro recibido por la función Cosmos que implementa la petición. Si no se define, el valor por defecto es ANSI con el código de página local. Valores posibles son: UTF-8 o un número que indicará el código de página ANSI.
returnCharset	Codificación de caracteres del valor de retorno de la función Cosmos que implementa la petición. Si no se define, el valor por defecto es ANSI con código de página en local. Valores posibles son: UTF-8 o un número que indicará el código de página ANSI.

Si el método recibe parámetros, se indicará el nombre y, opcionalmente, el valor por defecto de cada parámetro, así como la manera en que el parámetro llega al método. Un parámetro puede llegar mediante la URL o mediante la sección BODY de la petición HTTP.

Dependiendo del tipo de parámetro que reciba el método se deberán definir las siguientes secciones dentro de la sección <method>:

Sección <queryParameter>

Si un parámetro llega por la URL con el formato:

http:\\<servidor>:<puerto>\prjpath\modulepath\methodname?param1=value1

Se dirá que es un parámetro de query, y se definirá dentro de esta sección.

Dentro de esta sección se definen las siguientes propiedades:

Propiedad name	Nombre del parámetro.
Propiedad defaultValue	Valor por defecto. Esta propiedad es opcional.

Sección <urlParameter>

Si un parámetro llega por la URL, sin nombre de método, con el formato:

“http:\\<servidor>:<puerto>\prjpath\modulepath\value1\value2”

Se dice que es un parámetro de url, y se definirá dentro de esta sección.

Dentro de esta sección se definen las siguientes propiedades:

Propiedad name	Nombre del parámetro.

Propiedad defaultValue	Valor por defecto. Esta propiedad es opcional.

Sección <bodyParameter>

Si un parámetro llega desde la sección body de la petición HTTP, se dice que es un parámetro de cuerpo de petición, y se debe definir en esta sección.

Dentro de esta sección se definen las siguientes propiedades:

Propiedad name	Nombre del parámetro.
Propiedad defaultValue	Valor por defecto. Esta propiedad es opcional.

Sección <headerParameter>

Este parámetro permite consultar toda la información de la cabecera HTTP.

Dentro de esta sección se definen las siguientes propiedades:

Propiedad name	Nombre del parámetro.

El nombre del parámetro, ya sea de tipo <queryParameter>, <urlParameter>, <bodyParameter> o <headerParameter>, deberá coincidir con el nombre del parámetro indicado en el fuente Cosmos del método.

3.3.3 Ejemplo de llamada a servicio REST Cosmos

Dado el fichero XML de configuración definido anteriormente, para ejecutar el servicio resultCount del módulo states.smd, se deberá ejecutar desde la parte cliente una operación de tipo POST en la URL: http://<host>:<puerto>/stock/rest/states/resultCount

host	Indica el nombre o la dirección IP del servidor donde se está ejecutando Cosmoswebserver.exe.
puerto	Indica el puerto especificado en la variable de entorno PORT del fichero de configuración pasado como parámetro en la línea de comando de Cosmoswebserver.exe
stock/rest	Indica la ruta de acceso al proyecto de Cosmos definida en la propiedad “path” de la sección <project> del fichero XML de configuración.
states	Indica la ruta de acceso al módulo “states”, definida en la propiedad “path” de la sección <module>.
resultCount	Indica la ruta de acceso al método “resultCount”, definida en la propiedad “path” de la sección <method>.

Es decir, en la propiedad “name” de <project> se guarda el identificador del proyecto, y en la propiedad “name” de <module> y <method> se guarda el nombre del módulo Cosmos y el del método Cosmos respectivamente. En la propiedad “path” se almacena el término que se debe utilizar en la URL para acceder al proyecto, módulo y método.

3.4 Creación de un servicio REST en Cosmos

Un servicio REST en Cosmos se traduce como una función o método definido en un módulo de un proyecto en Cosmos, que recibe unos parámetros desde la URL o desde el cuerpo de una petición HTTP (request), y retorna un String con el resultado de la ejecución del mismo, así como de un código de estado donde se indica si la ejecución ha sido exitosa, fallida, etc.

Este código deberá ser un código de estado estándar HTTP (200 – ok, 201 – created, 405 – Method not allowed, etc.).

Este código de estado lo establecerá el método o función Cosmos correspondiente al servicio REST mediante la ejecución del método SetExecStatus de la clase Module al finalizar el cuerpo de la función.

Ejemplo del servicio REST findByKey implementado en la función findByKey del módulo States.smd. En ella se realiza la conexión a la base de datos y se indica que no muestre errores por pantalla. La función process-QuerySingleResult realiza la búsqueda, recopila los registros en un objeto JSON, que retorna como cadena de caracteres, indica el estado de la ejecución con el método SetExecStatus, y se desconecta de la base de datos.

```
public function findByKey(key as char) return char
objects begin
    retStr stmtStr as char default NULL
end
begin
    Conecta();
    ErrorLevel(0);
    stmtStr = "select * from states ";
    if key is not null then begin
        stmtStr = stmtStr + " where state = '" + key + "'";
        retStr = processQuerySingleResult(stmtStr);
    end else begin
            Module.SetExecStatus(BAD_REQUEST);
    end
    Desconecta();
    return retStr;
end

private function processQuerySingleResult(stmtStr as char) return char
objects begin
    retStr as char default ""
    miCursor as SqlCursor
    jsonArray as json
    jsonRecord as json
    stateStr nameStateStr as char
    retStatus as integer
    found as boolean default FALSE
end
begin

    miCursor.Prepare(stmtStr);
    if Sql.Error() == 0 then begin
        miCursor.Open();
        if miCursor.Fetch(stateStr, nameStateStr).Found() then begin
            jsonRecord.Clear();
            jsonRecord.Set("state", stateStr);
            jsonRecord.Set("sname", nameStateStr);
            found = TRUE;
        end
        miCursor.Close();
        if not found then retStatus = getNoRecorsStatusCode();
```

```
    end else begin
        retStatus = BAD_REQUEST;
    end
    miCursor.Free();

    Module.SetExecStatus(retStatus);
    retStr = jsonRecord;

    return retStr;
end
```

4. Codificación de caracteres

Codificación de caracteres en versiones de Cosmos inferiores a la 7.8

En versiones de Cosmos anteriores a la 7.8, Cosmos convertía la cadena de caracteres enviada en el parámetro de la petición a ANSI con el código de página local. Esto provocaba que, si no existía correspondencia entre los caracteres de la petición y los de la codificación ANSI local del equipo donde se ejecuta Cosmos WebServer, la conversión y la representación gráfica de estos no fuera la correcta.

Cómo funcionará a partir de la versión 7.8

Por defecto funcionará como en versiones anteriores.

Si es necesario trabajar con un set de caracteres distinto al ANSI de la máquina local, será necesario definir las propiedades *parametersCharset* y *returnCharset* en el fichero xml de configuración del servicio Web de Cosmos. Al definir esta propiedad, Cosmos realizará la conversión al charset indicado en *parametersCharset*, y las cadenas de caracteres enviadas en la repuesta utilizarán el set de caracteres indicado en la propiedad *returnCharset*, siendo por defecto ANSI con el código de página local.

Propiedades parametersCharset y returnCharset.

Valores posibles son: UTF-8 o un número que indicará el código de página ANSI.

Se definen en el fichero xml de configuración del servicio a nivel de método.

parametersCharset. Codificación de caracteres del parámetro recibido por la función Cosmos que implementa la petición. Si no se define, el valor por defecto es ANSI con el código de página ANSI.

returnCharset. Codificación de caracteres del valor de retorno de la función Cosmos que implementa la petición. Si no se define, el valor por defecto es ANSI con código de página ANSI.

Ejemplo.

Si en el cuerpo de la petición se envía un JSON con codificación UTF-8 con texto en cirílico y desde el servicio Cosmos se quiere trabajar en ANSI, el valor que debe indicarse en la propiedad parametersCharset debe ser «1251», que es el correspondiente a la codificación ANSI en cirílico. Si se desea trabajar en UTF-8, el valor que debe indicarse en la propiedad parametersCharset debe ser «UTF-8».

Si la respuesta se quiere enviar en UTF-8, el valor de la propiedad returnCharset deber ser «UTF-8». La función Cosmos que implementa la petición debe ser la responsable de realizar la conversión al set de caracteres indicado en returnCharset.

5. Instalación y ejecución de Cosmos WebServer

La utilidad Cosmos WebServer puede ser iniciada desde línea de comando o como servicio Windows.

Antes de iniciar la instalación del aplicativo deberá estar instalada la máquina virtual de Java y asegurarse que incluye la DLL JVM.dll.

En versiones de Cosmos anteriores a la 7.4.2, para que Cosmos WebServer localice la DLL deberá añadirse a la variable PATH del sistema la ruta donde se encuentre dicha DLL.

A partir de la versión 7.4.2 de Cosmos no es necesario indicar la ruta de la JVM.dll en la variable de entorno PATH del sistema operativo. Para ello se han implementado las variables de entorno CWSUSEJAVAVERSION y CWSUSELASTJAVAVERSION. Estas dos variables de entorno deberán definirse en la sección **[Environment]** del fichero de configuración *cosmoswebserver.ini.*

Esto implica que, a partir de la versión 7.4.2 de Cosmos, el fichero *cosmoswebserver.ini* debe existir necesariamente en caso de utilizar las variables de entorno mencionadas, tanto si el aplicativo se arranca desde la línea de comando como desde un servicio de Windows.

El fichero *cosmoswebserver.ini* deberá estar situado en "COSMOSDIR\etc".

5.1 Ejecución desde línea de comando

Para iniciar Cosmos WebServer desde la línea de comando tendremos que utilizar la siguiente sintaxis:

```
Cosmoswebserver.exe –ini c:\cosmos\etc\stockserver.ini
```

El problema de iniciar Cosmos WebServer desde la línea de comando reside en que, al reiniciarse el servidor donde está instalado, es necesario reiniciar manualmente Cosmos WebServer.

Este problema se soluciona instalando Cosmos WebServer como servicio Windows.

5.2 Ejecución como servicio Windows

5.2.1 Instalación del servicio

Para instalar Cosmos WebServer como servicio será necesario ejecutar Cosmos WebServer con el parámetro "–install <nombre_de_servicio>":

```
Cosmoswebserver.exe –install ServicioWeb –user .\usuario –passwd passwd
```

El parámetro "nombre_de_servicio" indicará un nombre de servicio Windows no existente, y que será el que identificará al servicio de Cosmos WebServer. Este nombre de servicio deberá estar incluido en el fichero de configuración *cosmoswebserver.ini*.

Los parámetros "–user" y "–passwd" indican respectivamente el usuario que arranca el servicio y su contraseña. Estos parámetros son opcionales, si no se indican, el servicio arrancará con una cuenta de sistema local.

El servicio se instalará por defecto como "AUTOMÁTICO", es decir, cuando se reinicie el servidor el servicio se iniciará automáticamente.

El servicio se podrá instalar de manera silenciosa utilizando el parámetro «silent».

5.2.2 Inicio del servicio

Para iniciar el servicio Cosmos WebServer instalado con "-install" se deberá ejecutar Cosmos WebServer con el parámetro "–start <nombre_de_servicio>":

```
Cosmoswebserver.exe –start ServicioWeb
```

Cosmos WebServer necesita el nombre de un fichero de configuración. Este nombre lo obtendrá del fichero *cosmoswebserver.ini*, que deberá estar situado en "COSMOSDIR\etc". Por cada servicio que queramos instalar se deberá definir una sección con el nombre del servicio y una variable, llamada INIFILE, con el path absoluto del fichero de configuración para ese servicio.

Ejemplo de *cosmoswebserver.ini* con dos servicios Cosmos WebServer configurados:

```
[ServicioWeb]
INIFILE=c:\cosmos\etc\stockserver.ini
[ServicioWebCompras]
INIFILE=c:\cosmos\etc\comprasserver.ini
```

A partir de ese momento, el servicio estará disponible para aceptar nuevas conexiones.

El servicio se podrá iniciar de manera silenciosa utilizando el parámetro «silent».

5.2.3 Parada del servicio

Para detener el servicio Cosmos WebServer instalado con "–install" se deberá ejecutar Cosmos WebServer con el parámetro "–stop <nombre_de_servicio>":

```
Cosmoswebserver.exe –stop ServicioWeb
```

A partir de ese momento el servicio estará instalado, pero no disponible para aceptar nuevas conexiones, hasta que no se rearranque con "-start".

El servicio se podrá parar de manera silenciosa utilizando el parámetro «silent».

5.2.4 Desinstalación del servicio

Para desinstalar el servicio Cosmos WebServer instalado con "–install" se deberá ejecutar Cosmos WebServer con el parámetro "–remove <nombre_de_servicio>":

```
Cosmoswebserver.exe –remove ServicioWeb
```

El servicio se podrá desinstalar de manera silenciosa utilizando el parámetro «silent».

6. Ficheros de LOG

6.1 Fichero log4j

El servidor Cosmos WebServer permite la generación de un fichero de log en el que se almacenará información del funcionamiento de Cosmos WebServer.

Cosmos WebServer utiliza la herramienta Log4j2 para la generación del fichero de log.

Por defecto, Cosmos WebServer no genera fichero de log. Si se desea que la aplicación almacene información de log se deberá indicar en el fichero de configuración de Cosmos WebServer (ver el apartado 3.2 anterior).

El nombre del fichero de configuración de log se indicará en la sección [JAVA] mediante la definición de la siguiente variable:

```
-Dlog4j.configurationFile=c:/cosmos/cosmoswebserver/log4j2.xml
```

En este caso, el fichero de configuración de log se llama log4j2.xml. Este es un fichero de configuración en formato XML donde se podrá indicar la ruta y el nombre del fichero de log, el formato de salida y el nivel de «log» que se desea (ALL, DEBUG, ERROR, FATAL, INFO, OFF, TRACE y WARN).

A partir de la versión 8.0 de Cosmos, si se indica como nivel de «DEBUG» en el fichero log4j se podrán consultar los parámetros que se envían en la petición.

Ejemplo de fichero de configuración log4j2.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
LogLevel=ALL, TRACE, DEBUG, INFO, WARN, ERROR, FATAL, OFF
See: http://logging.apache.org/log4j/2.x/manual/index.html
 -->
<Configuration status="INFO">
  <Appenders>
    <Console name="Console" target="SYSTEM_OUT">
      <PatternLayout pattern="%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n"/>
    </Console>
    <RollingFile name="RollingFile" fileName="c:/cosmos/cosmosdata/logs/app.log"
                 filePattern="logs/$${date:yyyy-MM}/app-%d{MM-dd-yyyy}-%i.log.gz" ap-
pend="true">
      <PatternLayout>
        <Pattern>%d %p %c{1.} [%t] %m%n</Pattern>
      </PatternLayout>
      <Policies>
        <TimeBasedTriggeringPolicy />
        <SizeBasedTriggeringPolicy size="250 MB"/>
      </Policies>
    </RollingFile>
  </Appenders>
  <Loggers>
    <Root level="All">
      <AppenderRef ref="Console"/>
      <AppenderRef ref="RollingFile" level="INFO"/>
    </Root>
  </Loggers>
</Configuration>    retStr stmtStr as char default NULL
```

En este fichero de configuración se indica la ruta completa del fichero de log (propiedad «fileName» de la sección RollingFile), y el nivel de log (propiedad «level» de sección «AppenderRef»).

Para más información, ver: http://logging.apache.org/log4j/2.x/manual/index.html

6.2 Fichero CosmosWebServer.log

Este fichero se crea en el directorio c:\tmp, y en él Cosmos WebServer escribirá la información generada en la instalación, puesta en marcha, parada y eliminación del servicio Windows o en el proceso de arranque de Cosmos WebServer como aplicación.

Cosmos WebServer como servicio

```
2018-07-19 16:20:49  Checking JVM:JVM found
2018-07-19 16:20:49  Installing the service:cosmoswebserver
2018-07-19 16:20:49  cosmoswebserver instalado
2018-07-19 16:20:50  La configuración del servicio cosmoswebserver ha sido cambiada.
2018-07-19 16:21:07  Checking JVM:JVM found
2018-07-19 16:21:07  Starting the service:cosmoswebserver
2018-07-19 16:21:07  La configuración del servicio cosmoswebserver ha sido cambiada.
2018-07-19 16:21:08  Checking JVM:JVM found
2018-07-19 16:21:08  Running as service
2018-07-19 16:21:08  cosmoswebserver puesto en marcha.
2018-07-19 16:21:08  Starting Cosmos Web Server
2018-07-19 16:21:08  Ini file loaded: c:\cosmos\etc\cwsdemo.ini
2018-07-19 16:21:48  Checking JVM:JVM found
2018-07-19 16:21:48  Stopping the service:cosmoswebserver
2018-07-19 16:21:48  cosmoswebserver parado.
2018-07-19 16:21:52  Checking JVM:JVM found
2018-07-19 16:21:52  Removing the service:cosmoswebserver
2018-07-19 16:21:52  cosmoswebserver eliminado.
```

Cosmos WebServer como aplicación

```
2018-07-19 16:22:16  Checking JVM:JVM found
2018-07-19 16:22:16  Starting Cosmos Web Server
2018-07-19 16:22:16  Ini file loaded: c:\cosmos\etc\cwsdemo.ini
```

6.3 Fichero Cwslog.log

La salida de los errores generados por el runtime de Cosmos WebServer durante los procesos podrá ser a fichero si en el directorio temporal se crea un fichero con el nombre **CWSLog.log.**

A partir de la versión 8.4 de Cosmos, el nombre y la ubicación del fichero son configurables. Además, se podrá definir un fichero de log por cada servicio.

Cosmos WebServer como servicio

Si se produce un error durante la ejecución de un servicio Cosmos y el ErrorLevel es mayor de 0, en el fichero de log se volcará información referente al error, así como el método y el módulo donde se ha producido el error, y la pila de llamadas de la aplicación.

Si se produce un error durante la ejecución y el ErrorLevel es igual a 0, no se escribirá la información del error en el fichero de log.

Si en el código fuente del servicio Cosmos se ejecuta el método Trace, en el fichero de log se escribirá el texto asociado al método Trace.

Cosmos WebServer como aplicación

Si se produce un error durante la ejecución de un servicio Cosmos y el ErrorLevel es mayor de 0, en el fichero de log se volcará información referente al error, así como el método y el módulo donde se ha producido el error, y la pila de llamadas de la aplicación. Esta misma información se mostrará en pantalla mediante un MessageBox.

6.3.1 Ejemplo

En el proyecto de ejemplo se han realizado las siguientes modificaciones:

- En la función findByKey del módulo states se cambia la instrucción select para provocar un error.
- Para comprobar el valor de algunas variables se llama dos veces al método Trace.

Función findByKey:

```
public function findByKey(key as char) return char
objects begin
    retStr stmtStr as char default NULL
end
begin
    Conecta();
    key.Trace();
 //   ErrorLevel(0);
    stmtStr = "select * from1 states";
    stmtStr.Trace();
    if key is not null then begin
        stmtStr = stmtStr + " where state = '" + key + "'";
        retStr = processQuerySingleResult(stmtStr);
    end
    Desconecta();
    return retStr;
end
```

Cosmos WebServer como servicio

En el fichero de log se mostrará la siguiente información:

```
[2018-07-19 10:35:40] CWS Service Mode
[2018-07-19 10:35:40] Char::Trace
MA
[2018-07-19 10:35:40] CWS Service Mode
[2018-07-19 10:35:40] Char::Trace
select * from1 states
[2018-07-19 10:35:40] CWS Service Mode
[2018-07-19 10:35:40] Warning
Warning (M50700000)
File STATES
Method STATES::processQuerySingleResult
Falta la clausula FROM en la sentencia.
```

```
==========
CALL STACK
==========
-->SqlCursor::Prepare
STATES::processQuerySingleResult
STATES::findByKey
```

Cosmos WebServer como aplicación

En el fichero de log y en la pantalla se mostrará la siguiente información:

```
[2018-07-19 10:44:29] CWS Application Mode
[2018-07-19 10:44:29] Warning
Warning (M50700000)
File STATES
Method STATES::processQuerySingleResult
Falta la cláusula FROM en la sentencia.
==========
CALL STACK
==========
-->SqlCursor::Prepare
STATES::processQuerySingleResult
STATES::findByKey
```

NOTA: En este caso, al no haber utilizado al método SetExecStatus en la cabecera de la respuesta, el servidor Web retornará: 500 Server Error.