Cosmos WebServer

Cosmos WebServer (CWS) es una utilidad que permite utilizar Cosmos 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.

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

BASE100

BASE 100, S.A.
www.base100.com

Índice

1. Introducción

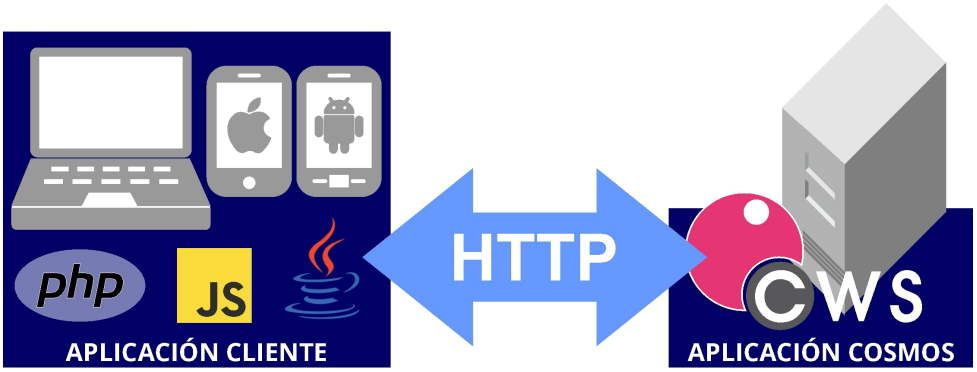
Cosmos WebServer se puede arrancar en línea de comando o bien como servicio Windows.

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.

La aplicación Cosmoswebserver.exe permite arrancar en línea de comando el servidor CWS. Esta aplicación recibe como parámetro un fichero de configuración.

Sintaxis:

```
Cosmoswebserver.exe –ini <ruta del fichero de configuración>
```

Parámetro:

-ini	Ruta del fichero de configuración.

3. Configuración

3.1 Fichero de configuración de la aplicación Cosmos WebServer

En el fichero de configuración que recibe como parámetro la aplicación Cosmoswebserver.exe 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 ejecución:

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

Ejemplo de fichero de configuración:

```
[Server]
CONFIG=c:\cosmosRestApp\etc\stockConfiguration.xml
PORT=8080
RESOURCEPATH=c:/cosmosRestApp/resources/webapp
[JAVA]
-Xms256m
-Xmx1024m
-Djava.class.path=c:\\cosmos\\bin\\cosmoswebserver.jar;
```

3.1.1 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.

3.2 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.

3.2.1 Ejemplo de fichero de configuración

A continuación se muestra un ejemplo de fichero de configuración de un servidor REST. En él se define 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>
```

3.2.2 Secciones del fichero de configuración

Sección <project>

Dentro se está sección se definen las siguientes propiedades:

Propiedad name	Identificador del proyecto en el servidor web.
Propiedad dirPath	Ruta del fichero de proyecto Cosmos dentro del sistema de archivos del servidor.
Propiedad 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:

Propiedad name	Nombre del módulo Cosmos.
Propiedad 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.

Propiedad name	Nombre del método Cosmos.
Propiedad path	Ruta de acceso en la URL del servidor.
Propiedad http	El verbo http que activa el método.

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.

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

3.2.3 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 propiedade "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.

3.3 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 processQuerySingleResult 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. Instalación como servicio Windows

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

Para iniciarla 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.

Para iniciar Cosmos WebServer como servicio Windows se debe utilizar la siguiente sintaxis:

```
Cosmoswebserver.exe [–install|start|stop|remove] [servicio] [–user <usuario>
–passwd <password>]
```

4.1 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.

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.

4.2 Inicio del servicio

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

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

Como hemos visto en la sección 3.1, CosmosWebServer necesita el nombre de un fichero de configuración. Este nombre lo obtendrá CosmosWebServer 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 CosmosWebServer 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.

4.3 Parada del servicio

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

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

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

4.4 Desinstalación del servicio

Para desinstalar el servicio CosmosWebServer instalado con "-install" se deberá ejecutar CosmosWebServer con el parámetro "-remove <nombre_de_servicio>":

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

5. Ficheros de LOG

5.1 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, CosmosWeb Server 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 3.1 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).

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

5.2 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
```

5.3 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.**

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.

5.3.1 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 la pantalla se mostrará la misma información:

En el fichero de log 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 clausula 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.