Artículo
Eduardo Anglada · 10 mayo, 2021 Lectura de 4 min

Open Exchange

Uso de ZPM para Node.js

Vamos a ver cómo usar el gestor de paquetes ZPM para instalar módulos de Node.js.. Para poder usar Node.js primero tenemos que cargar la API nativa de Node.js

El principio aplicado con ZPM es similar al de los módulos de Python y funciona perfectamente. Tal y como veremos en el ejemplo de github ZPM se encarga de todo!

En el ejemplo vamos a usar InterSystems IRIS con WebSockets de forma nativa como un cliente eco (devuelve el texto introducido). Existen varios ejemplos como este, o este otro en los que se puede ver a IRIS actuando de cliente.

La lógica del ejemplo siempre es la misma, después de la inicialización automática del servicio de eco, se escribe un comando y se espera su respuesta:

- seleccionar el servidor eco (en nuestro caso usamos la opción 1)
- escribes el texto que se va a transmitir y añades una línea con el carácter * para indicar que ya no hay más texto
- las respuestas se ven con el comando **S**
- para detener el servicio y salir se emplea **X**

Aclaración: como el servicio se ejecuta en segundo plano su salida estándar no es visible, pero se escribe un archivo que puede ser volcado mediante el comando **L**

Requisitos previos

Asegúrate de tener instalado git y el escritorio Docker.

Instalación

Clona el repositorio en un directorio local:

```
$ git clone https://github.com/rcemper/Using-ZPM-for-Node.js
```

Abre el terminal en este directorio y ejecuta:

```
$ docker-compose build
```

(esto puede llevar algún tiempo hasta que se complete)

Ejecuta el contenedor IRIS con este proyecto:

```
$ docker-compose up -d
```

Cómo probarlo

Utilizando el Terminal de IRIS:

```
$ docker-compose exec iris iris session iris "##class(rccjs.WSockNodeJs).Run()"

*** Welcome to WebSocket by Node.js Native API Demo ***

********* Node.js process id = 1650  *********

Known Hosts (*=Exit) [1]:
1  ws://echo.websocket.org/
2  --- server 2 ----
3  --- server 3 ----
select (1): 1 ==> ws://echo.websocket.org/
#
Enter text to get echoed from WebSocketClient Service
Terminate with * at first position
or get generated text by %
or append new text with @

1    hello this is connected over
2    IRIS Native API for Node.js
3    -----------------
4    *

Select action for WebClient Service
New EchoServer (E), New Text(N), Send+Listen(S)
Show Log (L), Exit+Stop WsClient(X) [S] :s
%%%%%%%%%%%%%%%%%%%%%%%%%%

******* 3 Replies *******
1    hello this is connecte over
2    IRIS Native API for Node.js
3    -----------------

Select action for WebClient Service
New EchoServer (E), New Text(N), Send+Listen(S)
Show Log (L), Exit+Stop WsClient(X) [S]: L
%%%%%%%%%%%%%%%%%%%%%%%%%%

platform = linux: ubuntu

        ****************************
        *** no IRIS host defined ****
        Connect to IRIS on: localhost
Successfully connected to InterSystems IRIS.
        *** wait 3sec for request ***
        ******* Startup done ********

        *** wait 3sec for request ***
        *** wait 3sec for request ***
        *** wait 3sec for request ***
        *** wait 3sec for request ***
        echoserver:  ws://echo.websocket.org/
```

```
        ** Lines to process: 3 **
        ********* next turn *********

        * WebSocket Client connected *
        ****** Client is ready ******
Line: 1 text> 'hello this is connecte over '
Received: 1 > 'hello this is connecte over '
Line: 2 text> 'IRIS Native API for Node.js '
Received: 2 > 'IRIS Native API for Node.js '
Line: 3 text> '----------------- '
Received: 3 > '----------------- '

        ******* lines sent: 3 ******
        *** replies received: 3 ****

        *** wait 3sec for request ***
        *** wait 3sec for request ***

Select action for WebClient Service
New EchoServer (E), New Text(N), Send+Listen(S)
Show Log (L), Exit+Stop WsClient(X) [S] :x
%%%%%%%%%%%%%%%%%%%%%%%%%%
```

La carga del API nativa para Node.js se realiza en el iris.script:

```
do $System.OBJ.LoadDir("/opt/irisapp/src","ck")
zn "USER"
zpm "load /opt/irisapp/ -v":1:1
```

Que a su vez es ejecutado en el Dockerfile.

Este artículo está inspirado en otro de @Evgeny Shvarov, pero en este caso se aplica a los módulos de **Node.js** y usa este ejemplo de la API nativa de IRIS para Node.js.

#API #Globals #Node.js #InterSystems IRIS
Ir a la aplicación en InterSystems Open Exchange