

記事

[Toshihiko Minamoto](#) · 2021年12月14日 5m read

[Open Exchange](#)

Node.jsでのZPMの使用

着想: [@Evgeny Shvarov](#)とその記事より

[Deploying InterSystems IRIS Embedded Python Solutions with ZPM Package Manager](#)

このアイデアを発展させ、同じことをNode.jsのモジュールで行ってみました。

このケースは、私の「[IRIS Native API for Node.js](#)」の例に基づいています。

InterSystems IRIS

は[クライアント](#)としてネイティブでWebSocketsをサポートしているというわかりきった返答を期待して:
その通りです。そして、[私](#)がその昔書いた[関連記事](#)と[OEXのサンプル](#)へのリンクはこちらです。

ZPMで適用される原理は、Pythonの場合と似ており、完全に機能します。

変更内容

- その他すべての**必須**Node.jsコンポーネントは、ランタイム時にインストールできるようになりました。
- Dockerビルド中には、intersystems-iris-nativeモジュールのみがプリインストールされます。
- Node.jsが起動すると、デバッグ目的でそのプロセスIDが表示されます。
- 一部の視覚的な外観とランタイムが改善されました。

Node.jsでは、~~追加の**必須**コンポーネントをインストールする際に、npmを使用し、十分な権限でインストールしなければならないという課題に直面します。~~
このステップとアクセス権の調整は、Dockerfileでカバーされています。
しかし、すべての.jsモジュールを処理するのは、ZPMです。

この例のロジックは変更されていません。

- Nodes.jsサービスが開始される
- 選択したエコーサーバーのアドレスがそれに渡される > E
- 送信されるテキストを作成する > N
- それを送信して、返信がどのようにドロップされるかを確認する > S
- サービスを停止して終了する > X

元の例とは異なり、サービスはバックグラウンドで実行しています。

従って、その出力は見えませんが、アクションによって表示できるログファイルに書き込まれます > L

前提条件

[git](#)と[Docker desktop](#)がインストールされていることを確認してください。

インストール

このリポジトリを任意のローカルディレクトリにClone/git pullします。

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

このディレクトリでターミナルを開き、以下を実行します。

```
$ docker-compose build
```

これが完了するまでしばらく時間が掛かることがあります。

このプロジェクトで IRIS コンテナを実行します。

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

テスト方法

IRIS ターミナルを使用します。

```
$ 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] <strong>:</strong><span style="">L</span></strong></span></strong>
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
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 ***
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
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

[#API #Node.js #グローバル #InterSystems IRIS](#)
[InterSystems Open Exchangeで関連アプリケーションを確認してください](#)

ソースURL:

<https://jp.community.intersystems.com/post/nodejs%E3%81%A7%E3%81%AEzpm%E3%81%AE%E4%BD%BF%E7%94%A8>