Artículo
Javier Lorenzo Mesa · 11 mar, 2020 Lectura de 6 min

Node.js: Prueba de la instalación y conexión para usarlo con Caché

¡Hola a tod@s!

El desarrollo completo en JavaScript (Full-Stack) permite crear aplicaciones de última generación con Caché. En cualquiera de las aplicaciones (web) que se desarrollan hoy en día, hay que tomar muchas decisiones estructurales y debemos saber cuales son las decisiones correctas. Con el conector Node.js disponible para Caché, se puede crear un potente servidor de aplicaciones, que permitirá utilizar la última tecnología de JavaScript y marcos de aplicaciones (frameworks) tanto del lado del cliente como del servidor.

Con todas estas nuevas tecnologías, lo más importante es integrarlas de la manera más eficiente posible y que permitan generar una experiencia de desarrollo muy productiva. Este artículo pretende introducirte paso a paso en la tecnología Node.js.

Antes de empezar a desarrollar aplicaciones con Node.js para Caché, lo primero que hay que hacer es configurar un entorno.

Para utilizar Node.js con Caché, se necesita un módulo (conector) Node.js para Caché.

En cada kit de distribución de Caché, encontrarás diferentes versiones de este módulo en el directorio bin de Caché, los cuales se llaman, por ejemplo: cache0100.node, cache0120.node... Como las versiones de Node.js se actualizan con frecuencia, recomiendo que primero solicites al Centro de Soporte Internacional (WRC) que te envíen la última versión.

Una vez que recibas del WRC el paquete zip con la última versión, verás que hay diferentes números de versión dentro del directorio bin. No es difícil elegir la versión correcta, pero es muy importante saber que estos módulos son nativos y que **la plataforma de Caché, el procesador (x86 o x64) y la versión del módulo Node.js deben coincidir** con tu sistema. Este es un paso muy importante a tener en cuenta, pues de lo contrario el conector de Node.js no funcionará en tu sistema.

Las versiones de Node.js están disponibles en dos opciones: las "LTS" y las "actuales". Para nuestras aplicaciones, siempre recomiendo utilizar las LTS, ya que son las más adecuadas para usarse con Caché (proporcionan soporte a largo plazo y estabilidad).

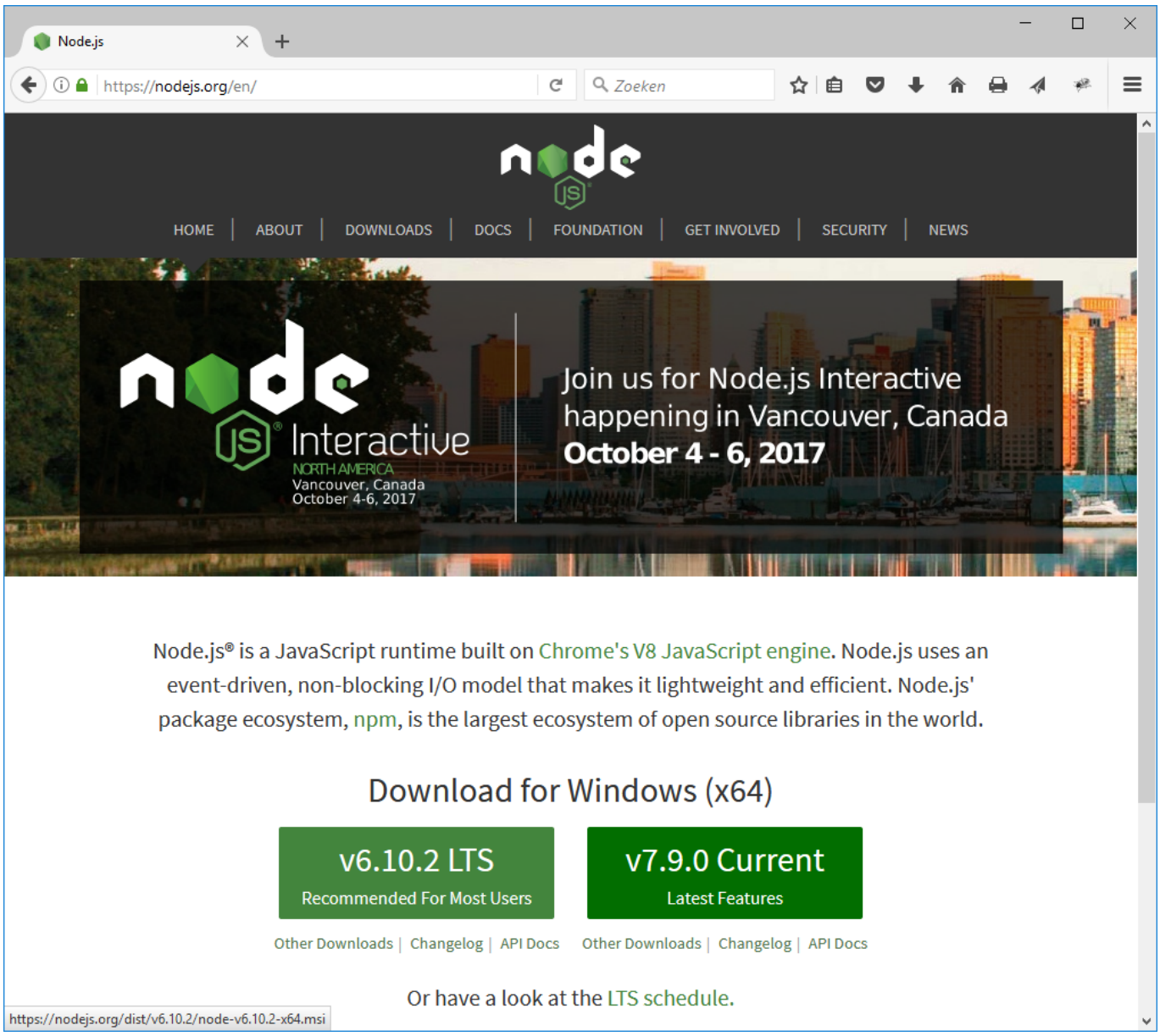
* Si deseas conocer todos los detalles sobre las versiones, puedes encontrar un resumen de todas las versiones en la página web de Node.js. Verás que es una lista larga, pero solo necesitas la última versión de LTS (incluso con los principales números de la versión, que actualmente son v6.x.x y v4.x.x). No utilices las versiones principales que tengan números impares, son versiones que no están diseñadas para usarse en programación y se utilizan para introducir y probar las funciones más recientes de JavaScript la versión 8.

Dos ejemplos:

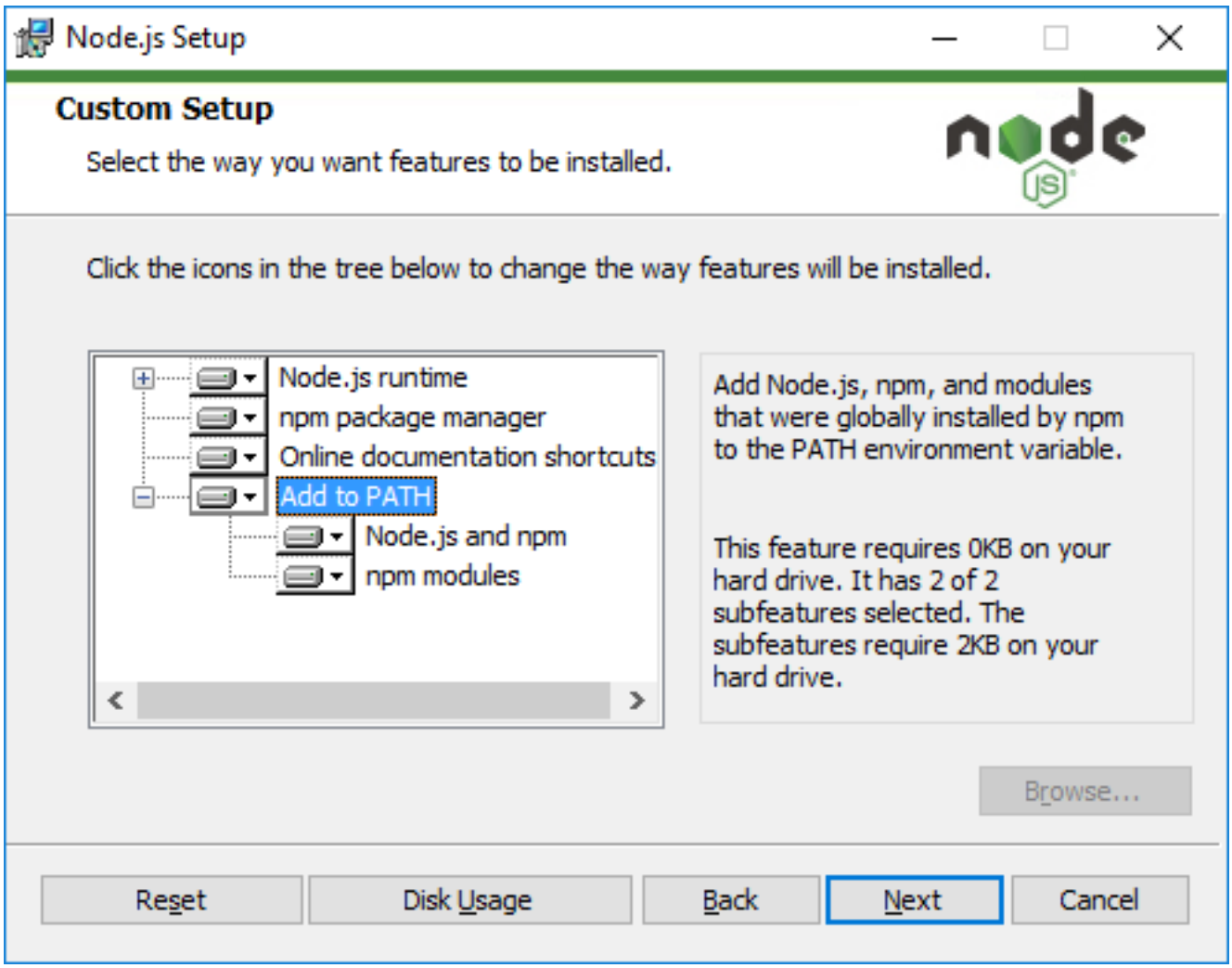
- si utilizas Caché 2016.2 ejecutándose en un sistema **Windows x64** y quieres utilizar **Node.js v6.9.4**, necesitarás que el archivo cache610.node esté en el directorio **bin\winx64**.
- si utilizas Caché 2008.2 ejecutándose en un sistema **Windows x86** y quieres utilizar **Node.js v4.8.2**, necesitarás que el archivo cache421.node esté dentro del directorio **bin\winx86**.

Ahora describiré paso a paso cómo descargar e instalar Node.js en un sistema Windows x64 y conectarlo con Caché.

Primero, se comienza con la descarga del último Node.js LTS release:



La versión actual es la v6.10.2, solo hay que hacer clic en el botón verde para descargarla. Instala esta versión en tu sistema con la configuración predeterminada, asegurándote de que la opción "add to PATH" esté instalada/verificada:



Ahora que has instalado el entorno de ejecución (runtime) en Node.js, se puede verificar si se instaló correctamente al abrir una línea de comandos y verificar la versión del nodo:

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>node -v
v6.9.1

C:\WINDOWS\system32>_
```

Como se puede ver, actualmente tengo instalada la versión Node.js v6.9.1 en mi sistema.

Por cierto, Node.js ahora está instalado dentro de tu sistema de Windows (en la carpeta "Archivos de programa").

En el portal de gestión de Caché, también deben asegurarse de que el %ServiceCallIn se encuentra entre tus servicios de seguridad.

Cuando se inicia una nueva aplicación con Node.js, es una práctica habitual crear primero un nuevo directorio para la aplicación. Lo primero que necesitamos crear es una pequeña aplicación/script en Node.js para probar la conexión entre Node.js y Caché. Este es el paso más importante para comprobar, en primer lugar, si nuestra configuración de Node.js funciona bien antes de comenzar con el desarrollo de otros módulos en Node.js.

Crea un directorio vacío para la prueba, por ejemplo C:\Temp\nodetest

Crea un subdirectorio nodemodules (C:\Temp\nodetest\nodemodules)

Inserta el cache610.node correcto (que se encuentra dentro del directorio bin\winx64 en el archivo del conector Node.js para Caché) dentro del directorio nodemodules y **cambia su nombre a cache.node**. En todas sus aplicaciones Node.js del lado del servidor, siempre es necesario que el conector para Caché cambie de nombre a cache.node, independientemente de la versión de Node.js, el procesador y sistema operativo en el que se esté ejecutando. ¡Esto hará que tus aplicaciones multiplataforma sean independientes de la versión y sistema operativos instalados!

* Cada vez que desarrolles una nueva aplicación del lado del servidor en Node.js, tendrás que añadir este módulo cache.node al directorio nodemodules. Simplemente copia la instancia desde tu aplicación de prueba al directorio nodemodules, que se encuentra en la nueva aplicación en sus sistemas. Recuerda que si instalas/implementas tus aplicaciones del lado del servidor en/hacia otro sistema, necesitará revisar cuál de las versiones debes instalar en dicho sistema, ya que la versión de Node.js, el procesador del sistema y el sistema operativo pueden ser diferentes!

Ahora, solo tenemos que crear nuestro script de prueba nodeTest.js dentro de C:\Temp\nodetest mediante este

código:

```
// first, load the Cache connector module inside node_modules
let cacheModule = require('cache');
// instantiate a Cache connector object in JavaScript
let db = new cacheModule.Cache();
console.log('Caché database object instance: ', db);

// Open the connection to the Caché database (adjust parameters for your Cache system
):
let ok = db.open({
  path: 'C:\\InterSystems\\Cache\\Mgr',
  username: '_SYSTEM',
  password: 'SYS',
  namespace: 'USER'
});

console.log('Open result: ', ok);
console.log('Version: ', db.version());

let d = new Date();
// construct a JavaScript global node object to set a test global in the USER namespa
ce
let node = {
  global: 'nodeTest',
  subscripts: [1],
  data: 'At ' + d.toUTCString() + ': global set from Node.js'
};
// set the global in the database
db.set(node);
// retrieve the global contents back from Cache
let result = db.get(node);
// show it on the console
console.log('Set global ^nodeTest(1) result: ', result);
// close the database connection
db.close();
```

Si guardas este script e instalas todo correctamente, este código debería funcionar inmediatamente cuando lo llames dentro de una línea de comandos:

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd \Temp\nodetest

C:\Temp\nodetest>node nodeTest.js
Cache database object instance:  Cache {}
Open result:  { ok: 1, result: 1, cache_pid: 9352 }
Version:  Node.js Adaptor for Cache: Version: 1.1.113 (CM); Cache Version: 2015.
2 build 705
Set global ^nodeTest(1) result:  { global: 'nodeTest',
  subscripts: [ 1 ],
  data: 'At Wed, 19 Apr 2017 15:42:17 GMT: global set from Node.js',
  ok: 1,
  defined: 1 }

C:\Temp\nodetest>
```

¡Enhorabuena! Ya has probado tu servidor para aplicaciones Node.js y lo has conectado con éxito a tu base de datos en Caché.

Con el script de prueba te darás cuenta de que el módulo cache.node contiene toda la funcionalidad que necesitas para acceder a tus datos en Caché (¡no solo los globales, también las funciones, las clases y el SQL!). Encontrarás toda la documentación sobre el módulo Node.js para Caché en los documentos disponibles online en Using Node.js with Caché. Dentro del archivo zip que te proporcione el WRC, también encontrarás un archivo PDF en el directorio de los documentos con todos los detalles sobre las funciones.

También te darás cuenta de que cuando comiences a programar con el módulo Node.js para Caché , este te proporcionará funciones de nivel inferior para acceder a Caché.

En el siguiente artículo, mostraré cómo se pueden utilizar algunos módulos de Node.js para escribir el código fuente de JavaScript, para acceder a sus globales en Caché, las funciones, las clases y las preguntas sobre SQL que utilizan una abstracción funcional de alto nivel.

#JavaScript #JSON #Lanzamiento #Node.js #Caché

URL de fuente:https://es.community.intersystems.com/post/nodejs-prueba-de-la-instalaci%C3%B3n-y-conexi%C3%B3n-para-usarlo-con-cach%C3%A9