Introduzione

Benvenuti in. Node.js una piattaforma nuova ed eccitante per scrivere applicazioni web e di rete che negli ultimi anni ha fatto parlare molto di sé e ha raccolto rapidamente un numero importante di seguaci nella comunità degli sviluppatori. In questo libro approfondirò l'argomento, il perché è così speciale e vi permetterò in breve tempo di scriverci dei programmi. Scoprirete presto che le persone sono molto flessibili con il nome di Node.js e spesso vi si riferiscono con *Node* o anche semplicemente con *node*. In questo libro farò lo stesso anch'io.

Perché Node.js?

Node.js ha avuto successo per un paio di ragioni principali, che vado a esporvi.

Il Web

In passato, scrivere applicazioni web era un processo piuttosto standard. Avete uno o più server sulla vostra macchina che restano in ascolto su una certa porta (per esempio 80 per HTTP) e quando si riceve una richiesta, si attiva un nuovo processo o un thread per partire con l'elaborazione e rispondere alla query. Questo lavoro spesso implica la comunicazione con servizi esterni, come database, cache di memoria, server di elaborazione esterna o anche solo il file system. Quando il lavoro finisce, il thread o il processo viene segnalato come "disponibile" e si possono gestire altre richieste.
È un processo lineare, semplice da capire e diretto da codificare. Esistono tuttavia due svantaggi che minano il modello.

1. Ognuno di questi processi porta con sé un sovraccarico di memoria. In alcune macchine, l'accoppiata PHP + Apache può arrivare fino a 10-15 MB per thread. Anche in ambienti dove un server di grandi dimensioni è in esecuzione continua e chiama thread per elaborare le richieste, ognuno di questi genera un sovraccarico per creare un nuovo stack e un ambiente di esecuzione, portandovi spesso al limite di memoria disponibile nel server.
2. Negli scenari di utilizzo più comuni, in cui un server web comunica con un database, un server di cache, un servizio esterno, oppure con il file system, la maggior parte

del tempo è inutilizzata, in attesa che i servizi terminino l'elaborazione e restituiscano i risultati. In questi momenti, il thread è effettivamente bloccato e non può fare nient'altro. Le risorse che consuma e i processi che esegue sono completamente fermi in attesa delle risposte. Soltanto dopo che il componente esterno ha restituito i risultati, il processo sarà libero di concludere l'elaborazione, inviare i dati al client e, infine, prepararsi per la prossima richiesta in arrivo.

Perciò, nonostante sia abbastanza semplice da capire e utilizzare, il modello è inefficiente se i vostri script passano la maggior parte del tempo in attesa che un database termini l'esecuzione di una query, uno scenario piuttosto comune in molte delle applicazioni web moderne.

Sono state sviluppate e messe in pratica molte soluzioni per affrontare questo problema. Potete acquistare server web sempre più grandi e performanti, dotati di maggiore memoria. Potete sostituire i server HTTP più potenti e ricchi di funzionalità, per esempio Apache, con versioni più piccole e leggere, come lighttpd o nginx. Potete creare versioni ridotte ed essenziali dei vostri linguaggi di programmazione preferiti, come PHP o Python (Facebook è andato oltre questa soluzione, creando un sistema che converte PHP in codice C++ nativo per ottimizzare dimensioni e velocità). Infine potete affiancare più server per aumentare il numero di connessioni simultanee che potete soddisfare.

Nuove tecnologie

Mentre gli sviluppatori web di tutto il mondo erano impegnati nell'eterna lotta contro le risorse dei server e i limiti al numero di richieste che possono elaborare, sono successe altre cose interessanti.

JavaScript, il vecchio linguaggio (risale al 1995, più o meno) ben conosciuto (e spesso criticato) per realizzare script lato client è tornato di nuovo alla ribalta. Le versioni moderne dei browser hanno ripulito le implementazioni e aggiunto nuove funzioni per renderlo più potente e meno capriccioso. Con l'avvento di librerie client, come jQuery, script.aculo.us o Prototype, la programmazione in JavaScript è diventata divertente e produttiva. Le API voluminose sono state affinate, e sono stati aggiunti effetti dinamici di grande impatto.

Allo stesso tempo, è scoppiata una nuova competizione tra i browser, con Google Chrome, Mozilla Firefox, Safari di Apple e Microsoft Internet Explorer a combattere tutti per la corona di re dei browser. Tutte queste compagnie stanno investendo molto nella porzione JavaScript dei loro prodotti, poiché le moderne applicazioni web continuano a crescere in modo sempre più dinamico e basate sugli script. In particolare, JavaScript V8 di Google Chrome è molto veloce, oltre a essere open source per essere utilizzato da chiunque.

Detto questo, si è presentata l'occasione per qualcuno di procedere con un nuovo approccio allo sviluppo di applicazioni di rete. Da qui, la nascita di Node.js.

Cos'è esattamente Node.js?

Nel 2009, un tale di nome Ryan Dahl stava lavorando per un'azienda chiamata Joyent, che si occupava di cloud e servizi di virtualizzazione in California. Stava cercando di sviluppare tecnologie push per applicazioni web, simili a quelle che utilizza Gmail, ma quello che trovava non lo soddisfaceva. Alla fine optò per JavaScript, a cui mancava un

modello di input/output (I/O) robusto (così che Dahl avrebbe potuto scrivere la sua versione), ma aveva il vantaggio di mettere subito a disposizione l'interprete V8.
Ispirato da alcuni progetti simili nelle comunità di Ruby e Python, prese il V8 di Chrome e una libreria di elaborazione degli eventi chiamata *libev*, e realizzò la prima versione di un nuovo sistema chiamato Node.js. La metodologia primaria e innovativa in Node.js è che è realizzato intorno a un modello non bloccante di programmazione degli eventi. In breve, non scriverete mai (o quasi) codice bloccante.
Se la vostra applicazione web – per elaborare una richiesta e generare una risposta – deve lanciare una query su un database, avvia l'esecuzione e informa Node.js su cosa fare quando sarà restituito il risultato. Nel frattempo, il vostro codice è libero di iniziare l'elaborazione di altre richieste in ingresso, oppure ogni altra operazione necessaria, come pulire i dati o eseguire analisi.
Grazie a questo semplice cambiamento nel modo in cui l'applicazione gestisce le richieste e lavora, potrete realizzare server web che arrivano a gestire centinaia, se non migliaia di richieste simultanee su macchine che non hanno a disposizione molte risorse di memoria. Node gira in un processo singolo e il vostro codice perlopiù in un thread singolo, perciò le risorse richieste sono molte meno di quelle di altre piattaforme.
Questa velocità e queste capacità hanno bisogno di qualche precisazione, di cui dovete essere ben consci, in modo da poter iniziare a lavorare con Node con gli occhi ben aperti. Prima di tutto, il nuovo modello è differente da qualsiasi altra cosa abbiate mai visto prima e a volte può confondere. Fino a che non avrete fatto vostri alcuni dei concetti principali, parte del codice scritto in Node.js potrebbe sembrarvi un po' strano. La gran parte di questo libro riguarda i metodi principali che molti programmatori utilizzano per gestire la modalità asincrona e non bloccante di Node e spiega come sviluppare il vostro. Questo metodo di programmazione è centrato su applicazioni che svolgono molti compiti con molti processi, server o servizi. Node.js dà il meglio di sé quando la vostra applicazione web si districa tra connessioni a database, server di cache, file system, server di applicazioni e molto altro. L'altra faccia della medaglia è che non si tratta per forza di un ambiente ottimale per creare servizi che svolgono elaborazioni lunghe e complesse. Per questo motivo, il modello di Node composto di un singolo thread in un singolo processo può creare problemi se una certa richiesta occupa molto tempo per generare un dizionario di password complesse o elaborare un'immagine. In queste situazioni dovete essere molto cauti nell'utilizzo delle risorse, oppure considerare la possibilità di svolgere queste attività su piattaforme esterne, da richiamare poi come servizi del vostro programma Node.js.
Infine, Node.js è una piattaforma piuttosto nuova e ancora in fase di sviluppo attivo. Non ha ancora raggiunto (ad agosto del 2013) la versione 1.0 e sono rilasciate con costanza nuove versioni, talvolta con un ritmo vertiginoso.
Per ridurre i problemi causati dagli aggiornamenti frequenti, gli sviluppatori hanno iniziato a etichettare le porzioni del sistema con differenti gradi di stabilità, passando da *Unstable* a *Stable* e infine a *Locked*. I cambiamenti alle porzioni *Stable* o *Locked* sono rari e implicano ampie discussioni nella comunità per capire quanti problemi possano portare. Procedendo nella lettura del libro, faremo il punto su quali aree sono meno stabili di altre e vi suggerirò come poter minimizzare i pericoli di modifica delle API.
La buona notizia è che Node.js ha attirato una comunità ampia e vivace, oltre a mailing list, forum e gruppi di utenti che si dedicano a promuovere la piattaforma e fornire

aiuto se necessario. Una ricerca veloce su Google risponderà al 99 percento delle vostre domande in un paio di secondi, perciò non abbiate timore di cercare!

A chi si rivolge questo libro?

Ho scritto questo libro dando per scontato che abbiate familiarità con la programmazione, le funzionalità e la sintassi di almeno uno dei linguaggi maggiori, come Java, C/C++, PHP o C#. Sebbene non sia necessario che siate degli esperti, è probabile che siate andati oltre il concetto di "imparare X cose in Y giorni".
Se siete come me, avete creato alcuni progetti HTML/CSS/JavaScript e quindi avete "lavorato con" JavaScript, ma potreste non avere grande familiarità con esso e magari avete utilizzato soltanto il codice trovato in qualche blog o mailing list. In realtà, con il problema delle differenze dei browser e una UI piuttosto inelegante, potreste aver alzato un sopracciglio al solo sentir parlare di JavaScript. Non abbiate paura: entro la fine del primo capitolo, i ricordi spiacevoli di questo linguaggio apparterranno al passato, e spero che vi starete preparando a scrivere il vostro primo programma in Node.js con semplicità e un bel sorriso!
Ho dato per scontata anche una conoscenza di base delle applicazioni web: i browser inviano una richiesta HTTP a un server remoto, che la elabora e invia una risposta tramite un codice che indica il successo o il fallimento, oltre a dati opzionali come il codice HTML di una pagina da visualizzare o un oggetto JavaScript Object Notation, (JSON), contenente i dati richiesti. È probabile che vi siate connessi a server di database in passato, lanciato query, atteso le righe risultanti e così via. Quando inizierò ad affrontare i programmi di esempio, riprenderò il discorso e richiamerò la vostra attenzione su tutto ciò che può essere nuovo o particolare.

Come usare questo libro

Questo libro va trattato come un tutorial. Ho provato a bilanciare parole e codice, per dimostrare il più possibile ed evitare lunghe e noiose spiegazioni. In quelle situazioni in cui penso che sia interessante un approfondimento, vi segnalerò risorse o altre documentazioni per imparare di più, se lo desiderate (ma non è mai una necessità).
Il libro è diviso in quattro parti principali.

- "Parte I – Primi passi". Inizierete a installare ed eseguire Node, darete un'occhiata a JavaScript e alle estensioni utilizzate nel V8 di Google e in Node.js, per arrivare a scrivere la vostra prima applicazione.
- "Parte II – Imparare a correre". Inizierete a sviluppare applicazioni server più potenti e interessanti, e comincerò a insegnarvi alcuni dei concetti principali e delle pratiche utilizzate nei programmi Node.js.
- "Parte III – Scrivere applicazioni web". Vedrete alcuni strumenti molto potenti e i moduli disponibili per scrivere le vostre applicazioni, per aiutarvi con i server web e connettervi ai database.
- "Parte IV – Ottenere il massimo da Node.js". Chiuderò il libro dando uno sguardo ad altri argomenti avanzati, come i sistemi con cui eseguire le applicazioni in server

di produzione, come testare il codice e perfino come utilizzare Node.js per realizzare utility da riga di comando!

Mentre procedete con la lettura del libro, lanciate il vostro editor e inserite il codice, controllate come funziona con la vostra versione di Node.js e iniziate a scrivere e sviluppare il vostro software. Arriverete a creare una piccola applicazione di condivisione di fotografie, che spero vi fornisca idee e ispirazione per realizzare molto altro.

Scaricate il codice sorgente

Il codice sorgente per la maggior parte degli esempi e dei progetti dimostrativi inclusi in questo libro può essere scaricato all'indirizzo `https://github.com/marcwan/LearningNodeJS`. Vi consiglio di prelevarlo e giocarci ma non perdete l'opportunità di scrivere voi stessi del codice e provarlo.

Il codice disponibile su GitHub include alcuni esempi del tutto funzionanti ed è stato testato con successo su Mac, Linux e Windows, con le ultime versioni di Node.js. Se con il rilascio di nuovi aggiornamenti di Node si dovessero rendere necessarie modifiche al codice, troverete qui i cambiamenti e le note, perciò scaricate una nuova versione di tanto in tanto.

Se avete domande o problemi con il codice presente nel libro, sentitevi liberi di visitare il link `https://github.com/marcwan/LearningNodeJS` e segnalarli; il sito sarà monitorato e le risposte saranno fornite in modo piuttosto veloce.

Ringraziamenti

Desidero ringraziare tutti i Mark di PHPTR (sembra sia un nome molto comune) che mi hanno aiutato nella realizzazione di questo libro e di altri progetti. I copy editor sono stati un aiuto eccezionale.

Ho un grande debito di gratitudine verso Bill Glover e Idriss Juhoor per le eccellenti revisioni tecniche e di stile.

Infine, un grazie particolare a Tina, semplicemente per esserci.

L'autore

Marc Wandschneider è il co-fondatore di Adylitica, leader nel design e nello sviluppo di applicazioni web e mobili altamente scalabili. Gira il mondo fornendo consulenze ai team che lavorano su progetti software. Dopo la laurea conseguita presso la McGill University's School of Computer Science, ha passato cinque anni in Microsoft come sviluppatore e coordinatore dei reparti di Visual Basic, Visual J++ e .NET Windows Forms. Durante la permanenza presso SourceLabs, in qualità di Contract Software Developer/Architect, ha realizzato la piattaforma wiki open source SWiK. È autore di *PHP and MySQL LiveLessons* e *Sviluppare applicazioni web con PHP e MySQL*.