

NodeJS - chat, persistência em banco relacional e envio de mensagens por e-mail

Bruno da Silva, Joseane M. Faria

Pós Graduação em Engenharia de Software – Turma 27 – Pontifícia Universidade Católica de Minas Gerais (PUCMinas)

Belo Horizonte – MG – Brazil

bdsilvas@gmail.com, jmfaria1@hotmail.com

***Abstract.** This work has the intention of do an introduction on development using Nodejs, where among the explored points are included, the development ease to server-side applications, asynchronous execution, development using functional paradigm. Here are included also, facilities and difficulties the crossed way.*

***Resumo.** Este trabalho tem o intuito de fazer uma introdução no desenvolvimento utilizando NodeJS, onde dentre os pontos explorados incluirão, facilidade de desenvolvimento de aplicações Server-Side, execução assíncrona, utilização do paradigma funcional para desenvolvimento. Serão incluídos também, facilidades e dificuldades do caminho percorrido.*

1. Informações Gerais

O Node.js é uma plataforma de software, atualmente na versão 0.10.21, instalável, usada para criar programas de rede escalável do lado servidor, usando JavaScript como linguagem de script. Diferentemente da maioria dos programas em JavaScript, não é executada em um navegador web, mas sim como uma aplicação de JavaScript do lado do servidor Para o Node.js não é necessário um servidor web, como Apache, pois uma biblioteca de servidor built-in HTTP, torna possível executar um servidor web sem o uso de software externo.

Ele veio para solucionar o problema de escalabilidade, onde diferentes de linguagens como PHP e JAVA, o Node.js proporciona milhares de conexões simultâneas.

O Node.js é uma compilação pacote de motor de JavaScript do Google V8. O Node executa V8 JavaScript do Google um interpretador ultrarrápido escrito em C++ que tem um aspecto exclusivo: é possível fazer o download do mecanismo e integrá-lo em *qualquer* aplicativo que você desejar.

2. Principais características

2.1. Programação direcionada a eventos

O Node.js utiliza o que é chamado de modelo de programação direcionado a eventos, como por exemplo, quando uma conexão é aberta ou fechada, e dados recebidos pela conexão. O JavaScript, em si, é uma excelente linguagem para programação direcionada a eventos, pois permite funções e fechamentos anônimos e, mais importante, a sintaxe é familiar para quase todos que alguma vez já programaram. As funções de callback que são chamadas quando um evento ocorre podem ser escritas

no mesmo local onde você captura o evento. Basta aguardar um evento, escrever uma função de callback e a programação direcionada a eventos toma conta de tudo!

O Node.js é bom para situações em que um grande volume de tráfego é esperado e a lógica e o processamento necessários do lado do servidor não são necessariamente volumosos antes de responder ao cliente. Alguns exemplos são: uma API RESTful, Fila do Twitter e Servidor de arquivos de imagem. Para criar páginas criadas dinamicamente e aplicativos pesados em bancos de dados relacionais, o Node.js não é uma boa solução.

2.2. Paradigma Funcional

A linguagem funcional identifica blocos de códigos, geralmente repetidos, e encapsulam em funções, tornando em funcionalidades simples. Trazem consigo a ideia de programação modular. Como não existe um ponto inicial para o código executar, funciona bem com o NodeJS, que é assíncrono.

Traz como vantagens a manutenção e menor ocorrência de erros, alto nível de abstração.












A desvantagem é a menor eficiência quando um problema envolve muitas variáveis ou muitas atividades sequenciais.

2.3. Arquitetura Modular

O NodeJS divide sua estrutura em módulos quando instalado.

Esta arquitetura modular ainda contempla componentes internos como, para escutar portas TCP e HTTP, e serviços de I/O.

Ele também disponibiliza que criemos módulos para ele, ou utilizemos outros módulos de terceiros, da comunidade aberta, como Amazon S3 e Twitter API, bastando utilizar o utilitário Node Package Manager. Este é um módulo instalado para gerenciar a instalação de outros módulos.

	.bin	01/10/2013 21:42	Pasta de arquivos
	async	11/10/2013 22:34	Pasta de arquivos
	carrier	01/10/2013 21:42	Pasta de arquivos
	emailjs	05/10/2013 23:32	Pasta de arquivos
	express	01/10/2013 21:42	Pasta de arquivos
	mailer	05/10/2013 23:19	Pasta de arquivos
	net	01/10/2013 21:43	Pasta de arquivos
	nodemailer	13/10/2013 10:57	Pasta de arquivos
	socket.io	01/10/2013 21:43	Pasta de arquivos
	sqlite3	01/10/2013 21:45	Pasta de arquivos
	sys	01/10/2013 21:41	Pasta de arquivos

1- Diretório "node_modules", listando módulos utilizados nesse trabalho

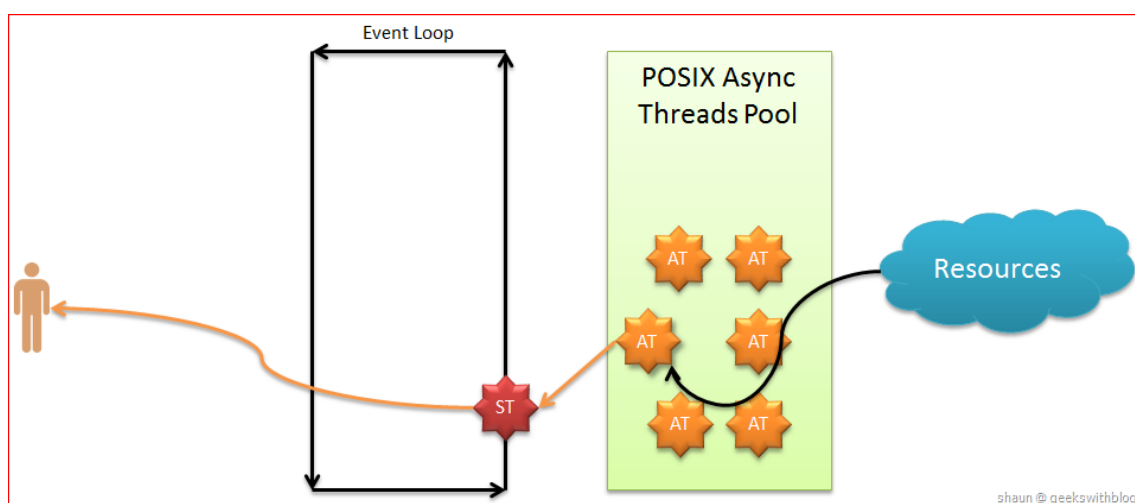
2.4. Execução Assíncrona

O NodeJS foi criado para utilizar ao máximo a programação assíncrona.

O NodeJS diz que o desenvolvimento fique disponível até que as ações se completem, sem bloquear a execução do código.

Enquanto na maioria das linguagens de programação, devemos esperar que um trecho do código execute, independente de quanto tempo ele demore, na programação assíncrona isso não é necessário. Podemos ter várias funções executando em paralelo, e definir através das *callbacks* qual deve executar ao final de um processo.

O desenvolvimento assíncrono em NodeJS é possível devido as call-backs, pois permite realizar processamentos paralelos, dando uma grande vantagem de desempenho.



2- Execução assíncrona

3. Sobre o Trabalho

3.1. Facilidades

Como o Nodejs é projetado para permitir o desenvolvimento de serviços de forma rápida e fácil, é visível, pela quantidade de código utilizado, qual é o retorno que essa plataforma nos dá, ainda mais quando comparado a outras soluções, como é o caso do Java.

A sintaxe é simples, já conhecida como “like C”, e que foi utilizada por várias outras linguagens, como C++, PHP, Java, C# e Javascript, onde essa última é a linguagem utilizada para desenvolvimento com Nodejs, sendo então interpretada pelo motor V8.

A possibilidade de extensão da plataforma de desenvolvimento se dá por meio de instalação de módulos, onde com o ambiente do Nodejs instalado, alguns já são instalados, como é o caso do Express. Para desenvolvimento que venha utilizar outros módulos, como envio de e-mails ou acesso a Banco de Dados, basta utilizar o gerenciador de pacotes do Nodejs, chamado de NPM. Utilizá-lo é muito simples, bastando ir para linha de comando e executar “npm install nome_de_pacote”, ele se encarrega de baixar descompactar e compilar algumas libs, análogo ao que é feito em

ambientes like unix, com o uso de apt-get.

3.2. Dificuldades

Utilizar uma ferramenta para executar um trabalho o qual ela não foi projetada para fazer, não apenas para o Nodejs, mas para qualquer outra solução, é bem mais trabalhoso do que a solução se apresenta. Essa condição foi uma das etapas encontradas nesse trabalho, onde a utilização de Nodejs com Bancos de Dados relacionais são suportadas, mas não são situações ideais, o que seriam Bancos NoSQL. Isso ocorre pelo principal motivo de sua criação, que é o assincronismo na execução de scripts, justamente o que permite sua agilidade ao trabalhar com envios e recebimentos de mensagens.

Outra condição que foi um obstáculo, é o uso de programação funcional, o qual se torna obrigatório quando há trechos no código que devem ser síncronos. Para isso é utilizado a call-back, que numa tradução livre é a re-chamada, ou seja, uma função A que chama uma função B, ao executar, dentro de B, A será chamada. Essa condição parece simples, porém quando há mais de um nível de call-backs, o desenvolvimento se torna complexo e por mais que se pense em uma solução, parece que o problema nunca é extinto, apenas é transferido para outro local no código. Isso se deve ao fato de não apenas de que Nodejs não é apenas uma nova ferramenta que foi estudada, e sim um novo paradigma de desenvolvimento, o qual exige uma nova postura lógica para construção de soluções.

Vale lembrar que há extensões do Nodejs que oferecem chamadas síncronas, porém essas não foram utilizadas.

4. Ferramentas

4.1. Utilizadas

Para execução do trabalho foi utilizado apenas o Bloco de Notas para criar e alterar os códigos.

Nodejs v0.10.21, com os pacotes net, carrier, “sqlite3”, “nodemailer”, “socket.io” e “express”.

Os testes foram realizados em uma máquina virtual contendo o Windows XP, o qual executou o papel do servidor Nodejs. Portanto, nessa máquina estava instalado o ambiente de execução e os pacotes necessários para executar o código em Nodejs, e a base de dados em SQLite.

O servidor foi executado via prompt do DOS.

Como cliente, qualquer dispositivo que contenha um cliente a suportar o protocolo Telnet, nesse caso uma máquina com Windows 7, utilizando o prompt do DOS.

Por fim, SQLite v3 como Banco de Dados.

4.2. Existentes

Como Nodejs foi, desde o início de sua criação, uma aposta de “big players” do mercado, como Google, Microsoft, Twitter, LinkedIn, é impossível acreditar que não

viriam IDEs para auxiliar no seu desenvolvimento. Segue então algumas encontradas, sendo que o conjunto delas é bem maior.

- Vim
- Aptana Studio
- Cloud9
- JetBrains
- Nodeclipse
- Sublime text
- Nide
- Komodo IDE

5. Código Fonte

Os códigos fontes utilizados nesse trabalho são originados em sua grande maioria de pesquisas feitas na internet, os quais, em nenhum momento têm os créditos de seus autores removidos. Suas fontes são citadas em “Referências”.

Os arquivos de código, e suas funcionalidades, foram.

telnet.js: Servidor de chat que aguarda conexões em uma determinada porta, ecoando as mensagens enviadas para todas as conexões existentes, como também controla o fluxo de execução, que é inserir Nickname e mensagem enviada na base de dados, verificar se todos os Clientes se desconectaram, disparando a execução do script que faz a consulta em banco do histórico do chat, e posteriormente o envio do mesmo por email, aos participantes.

- sqliteInsert.js: script chamado para fazer a inserção dos dados na base de dados.
- sqliteSelect.js: script chamado para retornar os dados existentes na base de dados, originado do chat.
- Mail3.js: script que processa o envio do histórico do chat a todos os participantes do mesmo.

Como não há nada de complexo no código executado, as descrições do que está sendo executado são encontradas no mesmo.

6. Referências

<<NodeJS>>

<http://nodejs.org>

<http://nodebr.com/como-funciona-a-funcao-require-do-node-js/>

<http://www.ibm.com/developerworks/br/library/os-nodejs/>

<http://nodebr.com/>

<http://msdn.microsoft.com/pt-br/magazine/jj991974.aspx>

<http://www.criarweb.com/artigos/caracteristicas-nodejs.html>

<http://udgwebdev.com/nodejs-para-leigos-introducao/>

<http://din.uem.br/~ia/ferramen/lisp/Paradigma.html>

<<Telnet Chat>>

<http://udgwebdev.com/nodejs-criando-um-mini-chat/>

<https://github.com/pgte/carrier>

<http://socket.io/>

<<Sqlite>>

<http://blog.modulus.io/nodejs-and-sqlite>

<http://github.grumdrig.com/node-sqlite/>

<<Email>>

<http://www.nodemailer.com/>

<http://blog.nodeknockout.com/post/34641712180/sending-email-from-node-js>