FIAP GRADUAÇÃO

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Arquiteturas Disruptivas e Big Data

PROF. ANTONIO SELVATICI

Versão 2 – Agosto/2016

SHORT BIO

É engenheiro eletrônico formado pelo Instituto Tecnológico de Aeronáutica (ITA), com mestrado e doutorado pela Escola Politécnica (USP), e passagem pela Georgia Institute of Technology em Atlanta (EUA). Desde 2002, atua na indústria em projetos nas áreas de robótica, visão computacional e internet das coisas, aliando teoria e prática no desenvolvimento de soluções baseadas em Machine Learning, processamento paralelo e modelos probabilísticos. Desenvolveu projetos para Avibrás, Rede Globo, IPT e Systax.

PROF. ANTONIO SELVATICI

profantonio.selvatici@fiap.com.br

MongoDB

MongoDB em Aplicações

- O modo ordinário de acessar o MongoDB em aplicações é através do uso de drivers que permitem executar consultas e modificar os dados sem a necessidade de uma linguagem intermediária como SQL
- Vamos ver dois exemplos de drivers: para Node.js e para Java
  - A página oficial contendo a documentação da API para MongoDB encontra-se em http://mongodb.github.io/node-mongodb-native/
  - A página oficial do driver do MongoDB para Java encontra-se em http://mongodb.github.io/mongo-java-driver/

NodeJS e MongoDB

- A alta disponibilidade do MongoDB torna seu uso com o Node.JS bastante apropriado
  - Lembrando que o NodeJS é um mecanismo para rodar servidores de rede de baixa latência e baixo consumo de memória, preferencialmente com uma ou poucas threads
- Tipicamente, NodeJS é usado para criar um serviço que responde a requisições na web de acordo com uma api (webservice)
- Os serviços que empregam NodeJS em geral não devem demorar para fornecer a resposta (baixa latência), sob o risco de sobrecarregar o servidor
- Por isso, MongoDB é muitas vezes a opção de banco de dados preferida para aplicações web rodando em NodeJS
- Há muitos pacotes para rodar MongDB com NodeJS
  - O pacote mais popular é o Mongoose, que permite usar o MongoDB como repositório para ODM (Object Data Mapping)
  - Há um driver da própria MongoDB para NodeJS, que torna sua utilização "parecida" com a do cliente JS oficial
  - Uma lista de projetos para acesso do MongoDB pelo NodeJS encontr-se em:
    - http://docs.mongodb.org/ecosystem/drivers/node-js/

Instalando o driver do MongoDB para NodeJS

- A forma oficialmente suportada é:
  - `npm install mongodb --msvs_version=2013`
  - `A instalação deve informar a versão do Visual Studio`
  - `Deve estar instalado e configurado o Python 2.7`
    - `set PYTHON=C:\Python27\python.exe`
- A página oficial contendo a documentação da API encontra-se em
  - http://mongodb.github.io/node-mongodb-native/
- Para instanciar um objeto mongodb, fazemos:
  - `var MongoClient = require('mongodb').MongoClient;`
- Para conectar a um database, passamos seu endereço e nome:

```
var url = 'mongodb://localhost:27017/test';
var colecao = null;
MongoClient.connect(url, function(err, db) {
    if(!err) {
        console.log("Conectado ao servidor");
        // A partir daí, podemos inquirir ao MongoClient a
        //coleção que nos interessa
        colecao = db.collection('colecao');
    } else console.log(''+err);
});
```

Operações CRUD com o MongoClient

- Os métodos **insert**, **update** e **remove** possuem duas versões:
  - `insertOne, updateOne e removeOne` operam sobre um documento apenas
  - `insertMany, updateMany e removeMany` operam sobre vários documentos simultaneamente
- Além dos argumentos padrão do cliente mongo, esses métodos requerem uma função de callback: `function(err, result)`
  - Se a chamada funcionou, `err é null`, e `result` possui o objeto de retorno da chamada
  - O objeto de retorno tipicamente possui estatísticas sobre a quantidade de documentos afetados pela chamada

Exemplo de criação de dados

```
db.collection('sensors').insertOne( {
      tipo : 'temperatura',
      valor: 23.5,
      loc: [ 2.5,3.4]
}, function(err, result) {
    if(!err) console.log('Novo dado de
sensor');
    console.log('' + result);
  });
```

Recuperando dados

- O método **find** também aceita os mesmos argumentos do cliente mongo, e retorna um objeto cursor, que pode ser inquirido de três formas:
  - O método `toArray()`, que fornece para o callback uma lista de documentos: `.find(...).toArray(function(err, docs){ ... });`
  - O método `next()`, invocado para cada documento de saída, capturando um por vez:
    ```
    .find(...).next(function(err, doc){ ... });
    ```
  - O método `each()`, invocado para cada documento de saída após recuperar todos os documentos:
    ```
    .find(...).each(function(err, doc){ ... });
    ```
- Também aceita os métodos `count()`,`sort()`,`skip()` e `limit()`

Exemplo de busca de documentos

```
db.collection('sensors').find({
    tipo:'temperatura'
}).each(function(err, doc) {
    if(err) {
        console.log('Erro:' + err);
    } else if (doc != null) {
         console.dir(doc);
    }
});
```

Driver do MongoDB para Java

- Para usar o driver para o MongoDB, é necessário ter o seguinte JAR no caminho de classes do Java
  - mongo-java-driver.jar
- Para conectar a uma instância, passamos seu endereço e porta:

```
MongoClient mongoClient = new MongoClient(
"localhost" , 27017 );
```

- Para selecionar um banco:

```
MongoDatabase db =
mongoClient.getDatabase("fiap");
```

Operações CRUD com o MongoClient

- Os métodos originais **insert**, **update** e **remove** possuem duas versões:
  - `insertOne, updateOne e deleteOne` operam sobre um documento apenas
  - `insertMany, updateMany e deleteMany` operam sobre vários documentos simultaneamente
- Esses métodos entendem como documento o tipo `org.bson.Document`, que pode ser construído de formas diferentes:
  - Através do construtor vazio e usando o método `append(String chave, Object valor)`, onde valor é o objeto desejado
  - Através do método `Document.parse(String json)`

Exemplo de criação e busca de dados

```
MongoDatabase db = mongoClient.getDatabase("fiap");
//Inserção
db.getCollection("sensors").insertOne(
    new Document()
        .append("tipo","temperatura")
        .append("valor",23.5)
        .append("loc", Arrays.asList(2.5,3.4))
);
//Consulta
Document selector = new Document("tipo", "temperatura");
for(Documento doc : db.getCollection("sensors").find(selector)) {
    System.out.println(doc.toJson());
}
```

- A consulta também suporta os métodos `sort()`, `count()`, `limit()`, `skip()`, `project()`

Exercício

- Criar um programa usando Node.JS ou Java que faça buscas na coleção unicorns
  - Busca por nome, ordenando a saída em ordem alfabética
  - Busca por unicórnios que gostam de maçã (apple) ou cenoura (carrot)
  - Busca por unicórnios pesando entre 450 e 700 libras

REFERÊNCIAS

- MongoDB. https://www.mongodb.com
- Documentação do Node.js MongoDB connector. https://mongodb.github.io/node-mongodb-native/index.html
- Documentação do Java Driver for MongoDB. https://mongodb.github.io/mongo-java-driver/

FIAP