Artigo

Introdução ao MongoDB

Veja neste artigo uma introdução ao framework que está sendo cada vez mais utilizados pelos desenvolvedores Java em seus recentes projetos. Também faremos um exemplo práticos, num comparativo com o SQL.

MongoDB é uma nova ideia de **banco de dados** trazendo conceitos de Banco de Dados Orientado a Documentos. No restante do artigo veremos mais sobre o MongoDB, explorando seus conceitos, vantagens, desvantagens, instalação, manipulação e realizaremos um exemplo prático de como manuseá-lo.

Banco de Dados Orientado a Documentos

A definição geral apresentada é que os Bancos de Dados orientados a Documentos utilizam o conceito de dados e documentos autocontidos e auto descritivos, e isso implica que o documento em si já define como ele deve ser apresentado e qual é o significado dos dados armazenados na sua estrutura.

Banco de Dados Orientados a Documentos tem como característica conter todas as informações importantes em um único documento, sendo que os

possuir identificadores únicos universais (UUID), possibilitar a consulta de documentos através de métodos avançados de agrupamento e filtragem (**MapReduce**) e também permitir redundância e inconsistência.

Esses bancos de dados também são chamados de Bancos **NoSQL** (Not Only SQL). Esse termo NoSQL é devido à ausência do SQL, mas esse tipo de Banco de Dados não se resume apenas a isso, por isso o termo não é o mais correto para esse novo tipo de Banco de Dados. No entanto, ele é aceito porque o termo já se popularizou na comunidade. Alguns chegaram a defender o termo NoREL (Not Relational), mas diferente do anterior este não foi muito aceito. De forma resumida esse tipo de Banco de Dados não traz consigo as ideias do modelo relacional e nem a linguagem SQL. Uma diferença fundamental entre os dois modelos surge quando precisamos criar relacionamentos nos bancos de dados relacionais, diferente dos bancos orientados a documentos que não fornecem relacionamentos entre documentos, o que mantém seu design sem esquemas. Dessa forma, ao invés de armazenar dados relacionados em uma área de armazenamento separado, os bancos de dados de documentos integram esses dados ao próprio documento. Contudo, ainda assim é possível armazenar dados relacionados de forma separada, isso pode ser feito usando uma coleção separada, porém, é preciso tomar cuidado com isso visto que os ganhos de desempenhos podem ser perdidos.

Existem diversos Banco de Dados NoSQL hoje, os mais conhecidos são: MongoDB, DynamoDB, Azure Table Storage, Berkeley DB, **Hadoop**, **Cassandra**, Hypertable, Amazon SimpleDB, CouchDB, RavenDB, **Neo4J**, Infinite Graph, InforGrid, etc.

Características do MongoDB

O MongoDB é um banco de dados orientado a documentos, diferente dos Bancos

Dessa forma, já temos uma primeira diferença entre os dois modelos, onde o Banco orientado a documentos lida com documentos e não com registros como no modelo relacional onde tudo é representado usando uma abordagem bidimensional (tabelas representadas através de duas dimensões: linhas e colunas).

Este Banco de Dados tem como característica ser código-fonte aberto licenciado pela GNU AGPL (Affero General Public License) versão 3.0, possuir alta performance, não possuir esquemas, ser escrito em C++, multiplataforma e ser formado por um conjunto de aplicativos JSON. Apesar do projeto MongoDB ter iniciado em 2007 o Banco de Dados somente foi concluído em 2009 lançando assim a primeira versão do MongoDB. Diversas linguagens e plataforma já possuem drivers para o MongoDB, entre elas destacam-se: C, C#, C++, Haskell, Java, JavaScript, Perl, PHP, Python, Ruby e Scala. Além disso, o MongoDB possui binários para diversas plataformas como Windows, Mac OS X, Linux e Solaris.

Entre as empresas que já utilizam o MongoDB destacam-se: Globo.com, SourceForge, FourSquare, MailBox (serviço de e-mail do Dropbox), LinkedIn, SAP, MTV, Pearson Education, e muitos outros. Uma lista com todos os serviços em ambiente de produção que estão utilizando o MongoDB pode ser encontrado em http://www.mongodb.org/about/production-deployments/.

Instalando o MongoDB

Para instalar o MongoDB devemos primeiramente baixa-lo em http://www.mongodb.org/downloads escolhendo uma versão de sistema operacional.

Agora já podemos descompactar o zip para alguma pasta.

Após isso devemos ir até a pasta onde descompactamos o MongoDB e ir na pasta

Agora já podemos executar o cliente de Shell do MongoDB. O aplicativo de Shell do MongoDB está incluído junto com a distribuição sendo localizado na pasta “bin”. No Windows, está na forma do aplicativo mongo.exe.

Através deste Shell podemos criar bancos de dados, documentos e coleções. Se em qualquer momento for preciso obter ajuda, basta dar o comando "help" na linha de comando do Shell do Mongo.

Por padrão, o Shell do Mongo se conecta ao banco de dados "test". Para mudar para outro banco de dados, usamos o comando "use nomeDoBanco". Se o banco de dados não existir o MongoDB o criará assim que forem incluídos dados nele. O Shell deve apresentar a mensagem: switched to db meuMongoDB.

Para criar um documento e armazená-lo em uma nova coleção chamada "colors" podemos usar o comando abaixo no Shell:

```
> db.colors.save({name:"red",value:"FF0000"});
```

Para verificar se o documento foi armazenado no banco de dados executamos o comando abaixo no Shell:

```
> db.colors.find();
```

Visto que o Shell do MongoDB usa **JavaScript**, é possível escrever construções regulares em JavaScript ao interagir com os bancos de dados.

A **Listagem 1** cria uma coleção de documentos de caracteres, cada um contendo a representação da cadeia de caractere e seu código ASCII associado.

Listagem 1. Coleção de documentos

```
... var doc = {char:char, code: char.charCodeAt(0)};
... db.alfabeto.save(doc);
... }
```

Este loop criará 26 documentos, um para cada letra minúscula do alfabeto, cada documento contendo o caractere em si e seu código de caractere ASCII.

Utilizando MongoDB

O MongoDB utiliza a sintaxe JSON que retém os dados usando pares de chave/valor. Segue na **Listagem 2** um exemplo criação de um esquema:

Listagem 2. Retendo dados

```
db.usuarios.insert( {
                    nome: "Higor Medeiros",
                    cidade: "Porto Alegre",
                    estado: "Rio Grande do Sul"
            }
  )
```

O equivalente em SQL seria como o código da **Listagem 3.**

Listagem 3. Retendo dados no SQL

```
CREATE TABLE USUARIOS (

id
MEDIUMINT NOT NULL AUTOINCREMENTS,
nome
Varchar(30),
cidade
Varchar(60),
estado
Varchar(60),
PRIMARY
KEY (id))
```

Depois ainda teríamos que criar um INSERT para adicionar as informações referentes ao usuário.

No primeiro exemplo acima temos um documento criado no MongoDB que possui três atributos: nome, cidade e estado. Como podemos observar não temos nenhuma definição de tipos dos dados, tamanho máximo para cada atributo, regras de validação ou restrições.

Para armazenar outros dados no banco de dados executamos o código abaixo:

```
db.meudb.save(MeusDados)
```

Sendo que o documento MeusDados poderia ter o conteúdo da **Listagem 4**:

Listagem 4. Conteúdo de MeusDados

```
MeusDados = {
          nome: "Higor Medeiros",
          cidade: "Porto Alegre",
          estado: "Rio Grande do Sul"
}
```

No exemplo acima estamos armazenando o documento "MeusDados" no banco "meudb".

Também podemos criar estruturas diferentes. Segue na **Listagem 5** outra forma de armazenar os dados acima:

Listagem 5. Conteúdo de MeusDados em outra armazenagem.

```
MeusDados = {
            nome: "Higor Medeiros",
            endereco: {cidade: "Porto Alegre", estado: "Rio Grande do Sul"}
  }
```

Também podemos fazer uma estrutura ainda mais complexa, como a da **Listagem 6**:

Listagem 6. MeusDados sob outra armazenagem

```
MeusDados = {
          nome: "Higor Medeiros",
          endereco: {cidade: "Porto Alegre", estado: {nome:"Rio Grande do Sul", sig
}
```

Podemos notar que não há regras de validação muito rígidas, assim podemos armazenar qualquer tipo de documento no banco de dados. Quando precisarmos adicionar um novo atributo no banco de dados podemos incluir este atributo apenas no documento onde ele é necessário, diferente do modelo relacional em que uma coluna aplica-se a todos os registros. Obviamente que isso deve ser realizado com cuidado para que a base de dados não fique totalmente desorganizada e com diversos atributos perdidos.

O interessante é que por meio de um update o MongoDB permite que possamos acrescentar dados, isso acontece porque não existe um esquema dentro das coleções, ou seja, as collections podem ser dinâmicas.

O MongoDB também permite a exclusão dos dados que é feito através do comando abaixo:

```
db.usuarios.remove( { estado: "Rio Grande do Sul" } )
```

Utilizando um **Banco de Dados Relacional** seria o equivale ao código abaixo:

```
DELETE FROM usuarios WHERE estado = "Rio Grande do Sul"
```

Outro comando interessante para deleção muito utilizado é simplesmente colocarmos o código da **Listagem 7:**

Listagem 7. Comando Delete

```
"DELETE FROM usuarios"
Utilizando o MongoDB faríamos semelhante conforme abaixo:
db.usuarios.remove()
```

Para atualização de dados com SQL temos o código da **Listagem 8:**

Listagem 8. Comando Update

```
UPDATE usuarios SET estado = "Rio de Janeiro" WHERE cidade = "Rio de Janeiro"
```

No MongoDB o equivalente seria conforme o código da **Listagem 9.**

Listagem 9. Comando Update no Mongo

```
db.usuarios.update( { cidade: { $eq:"Rio de Janeiro" } },
                          $set:   { estado: "Rio de Janeiro"} },
                          { multi: true }
  )
```

Outro comando muito utilizado é para fazer consultas. O MongoDB possui uma poderosa linguagem de consulta baseada em documentos, que torna a transição de um banco de dados relacional para o MongoDB bastante fácil, pois as consultas são convertidas com bastante facilidade.

Segue na **Listagem 10** uma consulta realizada com SQL:

Listagem 10. Consulta no SQL

```
SELECT * FROM usuarios WHERE estado = "Rio de Janeiro"
```

Utilizando o MongoDB teríamos o equivalente a **Listagem 11**:

Listagem 11. Consulta no Mongo

```
db.usuarios.find(
                  { estado: { $eq: "Rio de Janeiro" } }
)
```

Também poderíamos ter comparações numéricas, conforme exemplo abaixo:

```
SELECT * FROM usuarios WHERE idade > 25 AND idade <= 50
```

Utilizando o MongoDB teríamos o equivalente conforme abaixo:

```
db.usuarios.find(
                    { idade: { $gt: 25, $lte: 50 } }
  )
```

Também poderíamos ter o operador "like", que é muito utilizado como mostra a consulta a seguir:

```
SELECT * FROM usuarios WHERE nome like "Higor%"
```

Utilizando o MongoDB teríamos o equivalente conforme demonstrado abaixo:

```
db.usuarios.find(
            { nome: /^Higor/ }
  )
```

Outra operação muito comum é a ordenação ascendente e descendente em SQL conforme exemplificado abaixo:

```
SELECT * FROM usuarios WHERE cidade = "Porto Alegre" ORDER BY nome ASC
```

Utilizando o MongoDB teríamos o equivalente conforme abaixo:

```
db.usuarios.find({cidade : "Porto Alegre"}).sort({nome:1})
```

Para ordenar de forma descendente teríamos o SQL:

```
SELECT * FROM usuarios WHERE cidade = "Porto Alegre" ORDER BY nome desc
```

Utilizando o MongoDB teríamos o equivalente conforme abaixo:

```
db.usuarios.find({cidade : "Porto Alegre"}).sort({nome:-1})
```

Outro comando muito utilizado nas consultas SQL é o join. Abaixo temos uma consulta SQL usando um join e procurando dentro desta tabela, segue abaixo o código:

```
SELECT * FROM usuarios u INNER JOIN eventos e ON e.id = u.id
            WHERE e.publicado is not null
            GROUP BY u.email
```

Utilizando o MongoDB teríamos o equivalente conforme abaixo:

```
db.usuarios.find(('eventos.publicado': ($ne: null)))
```

Fazendo uma analogia com o Banco de Dados Relacional tem-se que o documento JSON seria o registro, a collection seria a tabela, os índices são os mesmos e o embedding e o linking seria o join.

As vantagens em utilizar o MongoDB começam a aparecer quando podemos representar objetos do mundo real da forma que eles são, e caso novos atributos surjam podemos aplica-los somente onde é necessário e não em todos os casos

Enquanto no modelo relacional somos obrigados a pensar em evitar o tempo todo a redundância de dados, motivo pelo qual existem os relacionamentos, no MongoDB temos uma situação diferente em que não há relacionamentos e a duplicação por vezes é até mesmo incentivada como nos casos observados acima onde os dados são armazenados do jeito que quisermos.

A ideia do MongoDB é que tenhamos documentos autocontidos obtendo todas as informações que necessitamos sem que seja necessário realizarmos vários joins, dessa forma fazemos apenas uma consulta e o retorno será o documento inteiro com todas as informações resultando num ganho significativo de performance.

Vantagens

Utilizando MongoDB temos uma melhor performance, visto que uma única consulta retorna tudo o que precisamos saber sobre o documento.

Os banco de dados NoSQL apresentam vantagens sobre os outros quando precisamos de escalabilidade, flexibilidade, manipulação de quantidade massiva de dados, bom desempenho e facilidade para consultas.

O MongoDB possui consultas bastantes simples de serem realizadas, visto que não existem transações e joins. As consultas são mais fáceis de escrever e mais fáceis de ajustar.

O escalonamento horizontal com Sharding é muito bem implementado no MongoDB. Sharding é utilizado quando temos muitos dados e estamos no limite do disco, dessa forma dividimos esses dados entre várias máquinas, assim temos mais rendimento e maior capacidade de armazenamento em disco. Portanto, quanto mais Shards maior será o armazenamento e o desempenho. O MongoDB disponibiliza essa opção. Para mais detalhes sobre Sharding podemos consultar a documentação oficial na página oficial do MongoDB em

http://www.mongodb.org/display/DOCS/Configuring+Sharding. Banco de dados relacionais muito utilizados como o MySQL não suportam esse tipo de solução por padrão, para isso teríamos que manipular os dados em uma camada acima da base de dados, sendo muito mais trabalhoso.

Como a linguagem de consulta SQL é fácil de ser convertida para MongoDB temos uma maior facilidade para migração de organizações que atualmente usam bancos de dados relacionais.

Por fim, o MongoDB também possui a funcionalidade chamada GridFS que é responsável por armazenar arquivos de grandes dimensões. Isso significa que vídeos, arquivos, etc, podem ser armazenados diretamente no banco de dados, diferente do modelo relacional que tinhamos que armazenar tudo no Filesystem e disponibilizar apenas uma referência para esses arquivos no banco de dados. O tempo de resposta desses arquivos armazenados no MongoDB é comparado ao tempo de resposta dos arquivos em Filesystems.

Desvantagens

Uma desvantagem é quando queremos alterar todos os registros relacionados a uma unidade semântica, nesse caso é preciso tratar um a um.

Uma pergunta a ser respondida ainda é a dúvida que todos os usuários possuem em relação ao suporte do MongoDB que todas as grandes empresas e projetos de alto valor necessitam. Até que ponto o MongoDB oferece esse suporte necessário? Outro ponto é em relação a disponibilidade do serviço. Essas e outras perguntas serão respondidas ao longo do tempo, o fato é que muitas empresas de grande porte tem utilizado em projetos de complexidade média, uma escolha natural devido ao pequeno tempo que essa nova tecnologia está no mercado e os resultados têm sido muito satisfatórios.

Criando uma Aplicação com MongoDB

Para exercitar os conhecimentos adquiridos com MongoDB vamos criar uma aplicação de exemplo para manipular um banco de dados MongoDB.

Primeiramente devemos baixar o MongoDB em www.mongodb.org/downloads escolhendo a versão para o nosso sistema operacional.

Após isso, basta descompactar numa pasta ou executar o instalador (se for ".msi").

Feito isso devemos criar uma pasta "data" e dentro da pasta "data" uma pasta "db" no C:/ do Windows. Nessa pasta o MongoDB colocará os bancos de dados criados.

Agora já podemos executar o aplicativo "mongod.exe" que está dentro da pasta "bin" do MongoDB. Esse aplicativo executa os bancos de dados do MongoDB na porta 27017. Para gerenciar as tabelas podemos utilizar o Shell do MongoDB executando o aplicativo "mongo.exe".

Também devemos baixar o arquivo ".jar" para que o nosso projeto Java consiga se comunicar com o Banco de dados. Para isso baixe o arquivo em https://github.com/mongodb/mongo-java-driver/downloads e coloque na raiz do projeto que será criado abaixo.

Agora devemos ir até o Eclipse e criar um novo projeto Java, conforme mostram as **Figuras 1 a 3.**

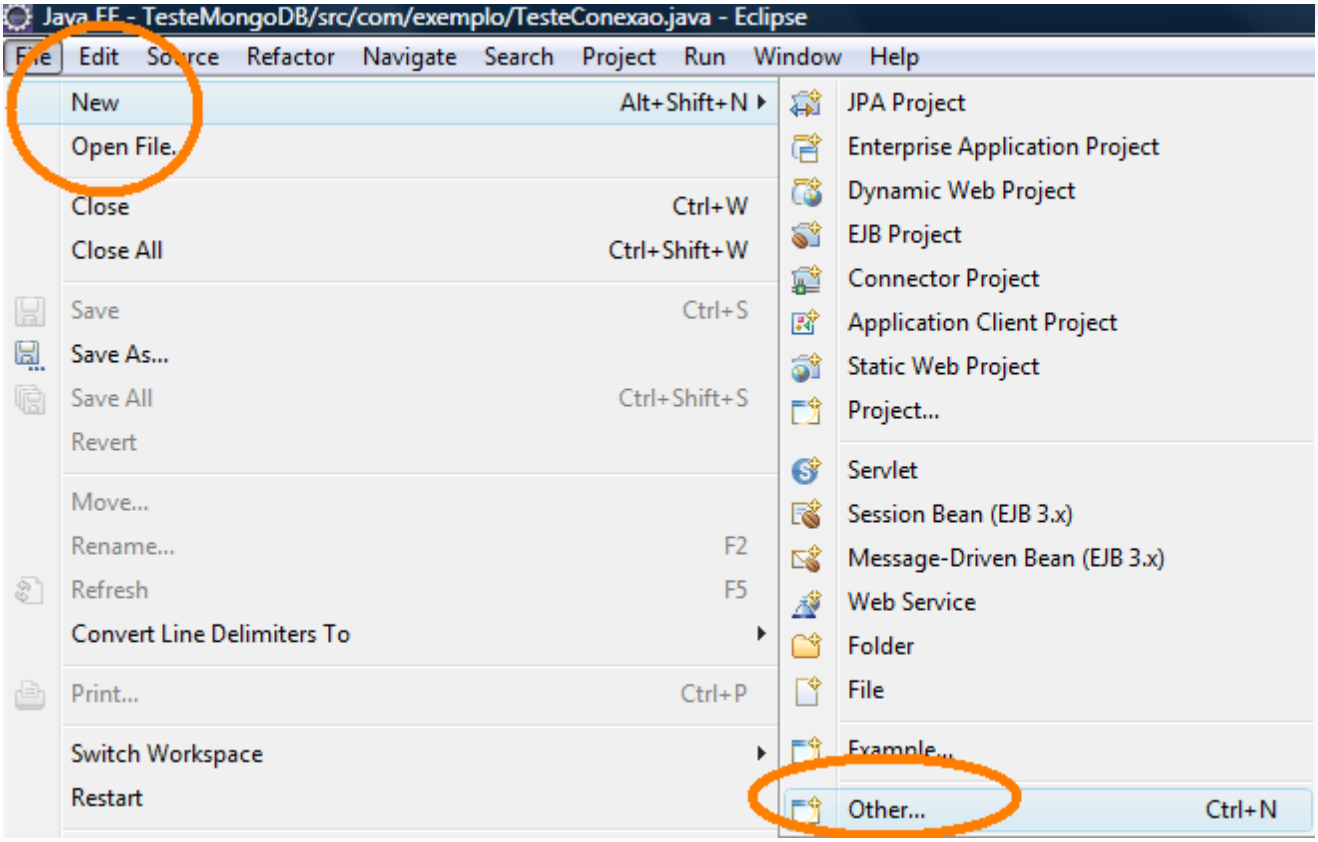

Figura 1. Selecionando Other para selecionar o tipo de projeto.

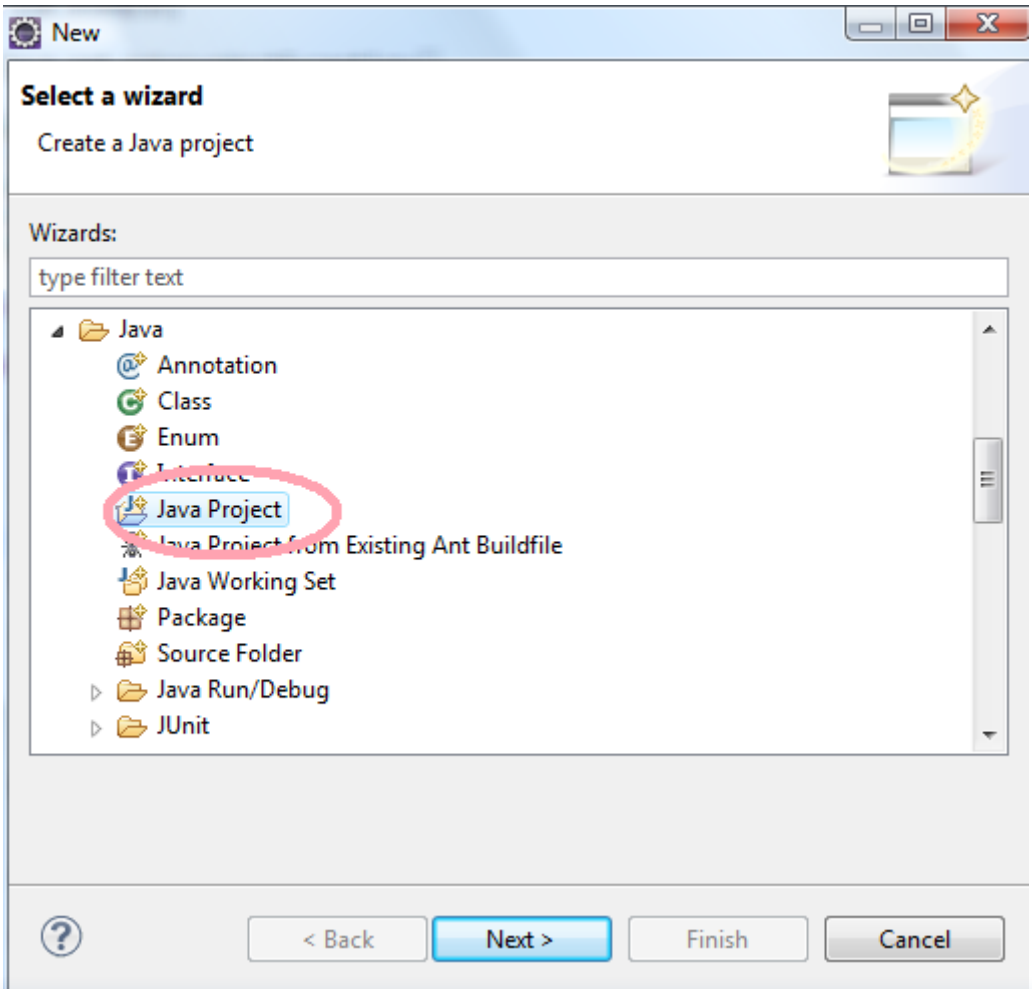

Figura 2. Criando um projeto Java.

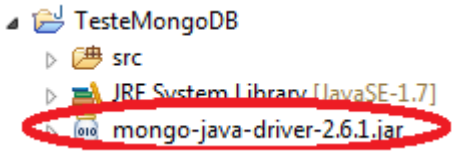

Figura 3. Colocando o jar na raiz do projeto.

Agora que já temos o nosso projeto Java criado vamos fazer um exemplo criando um pacote chamado “com.exemplo” e uma classe ExemploMongoDB.java dentro do pacote criado.

A nossa classe deverá ficar como a da **Listagem 12.**

Listagem 12. Exemplo de conexão e manipulação de dados com MongoDB.

```
package com.exemplo;

  import java.net.UnknownHostException;
  import java.util.List;


  import com.mongodb.BasicDBObject;
  import com.mongodb.DB;
  import com.mongodb.DBCollection;
  import com.mongodb.DBCursor;
  import com.mongodb.Mongo;
  import com.mongodb.MongoException;


  public class ExemploMongoDB {

        public static void main(String args[]) throws UnknownHostException, MongoE

              Mongo mongo = new Mongo("localhost", 27017);

              DB db = mongo.getDB("testemongodb");

              BasicDBObject doc = new BasicDBObject("username", "higormed").
                          append("nome",   "Higor Medeiros").
                          append("cidade", "Porto Alegre");
```

```
                collec.insert(doc);

            //lista as coleções
            DBCursor cursor = collec.find();
        int i=1;
        while (cursor.hasNext()) {
            System.out.println("Documento Inserido: "+i);
            System.out.println(cursor.next());
            i++;
        }

        System.out.println("Banco de dados armazenados:");
             List<String> dbs = mongo.getDatabaseNames();
             for(String dbStr : dbs){
                    System.out.println(dbStr);
             }

             System.out.println("fim execucao");

    }

}
```

O código acima cria um novo banco de dados caso ainda não exista um com o nome “testemongodb”, cria um novo documento com o nome “dados” e com os valores definidos, lista os documentos inseridos e por fim lista todos os bancos de dados criados até o momento.

Com isso já podemos criar um banco de dados, documentos, valores, listagens, e muitos mais com o MongoDB. Como exercício tente atualizar os dados, listá-los e por fim excluir os dados e listá-los novamente para verificar a remoção.

Referências

[1] B. Kyle. MongoDB in Action. Manning, 2011.

[2] Explore o MongoDB. Disponível em: http://www.ibm.com/developerworks/br/library/os-mongodb4/.

Por **Higor**
Em 2014

Suporte ao aluno - Deixe a sua dúvida.