
# Desenvolvimento de Software para WEB

## Aula 23 - MongoDB

Professor: Anderson Almada

# Introdução

- MongoDB é um software de banco de dados orientado a documentos livre, de código aberto e multiplataforma, escrito na linguagem C++.

- Classificado como um programa de banco de dados NoSQL, o MongoDB usa documentos semelhantes a JSON com esquemas.

- Suas características permitem com que as aplicações modelem informações de modo muito mais natural
    - Os dados podem ser aninhados em hierarquias complexas e continuar a ser indexáveis e fáceis de buscar.

# Instalação

- O primeiro passo é instalar o MongoDB (da mesma forma que foi realizado com o PostgreSQL).
  - Windows/Linux
    - https://www.mongodb.com/download-center/community
  - Ubuntu (Alternativa)
    - https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/

# Instalação

- Depois de instalado, crie a seguinte pasta na raiz do sistema
  - **/data**
  - Em seguida crie a seguinte pasta, agora dentro de /data
    - **/db**
  - No final, você terá:
    - **/data/db**

# Execute

- Para executar o banco de dados, você pode tanto deixar ele como um serviço em execução ou executar no terminal
  - **sudo mongod**

# Projeto WEB

- Depois de instalado o banco de dados, você poderá criar suas aplicações web. Como é um projeto maven, adicione a seguinte dependência no pom.xml

    - <dependency>
        <groupId>org.mongodb</groupId>
        <artifactId>mongo-java-driver</artifactId>
      </dependency>

# Crie o model

- Nesse exemplo, foi criado um model Curso que possui todos os campos como string até mesmo o id.

- O mongodb gera uma identificação de objeto que é uma string e será usada no exemplo para setar a informação de id do próprio model

- https://pastebin.com/njDMHmXs

# Crie a conexão

- Como será conectado localhost, não precisa passar os parâmetros

  **MongoClient mongo = new MongoClient();**

- Indica o nome do banco de dados que quer criar

  **MongoDatabase database = mongo.getDatabase("projeto");**

- Indica o nome da coleção (“tabela”) que você quer criar no banco de dados

  **MongoCollection<Document> collection = database.getCollection("cursos");**

# Inserir um objeto

- O objeto será transformado em um json e será inserido um elemento que é o documento que guarda informações do objeto

```
Curso2 curso = new Curso2(null, "anderson", "123");
ObjectMapper mapper = new ObjectMapper();
Document document;
try {
        document = Document.parse(mapper.writeValueAsString(curso));
        collection.insertOne(document);
} catch (JsonProcessingException e) {
        e.printStackTrace();
}
```

# Retornar todos os objetos

- Método para retornar todos os objetos inseridos

```
MongoCursor<Document> find = collection.find().iterator();
List<Curso2> list = new ArrayList<>();
while (find.hasNext()) {
        Document obj = find.next();
        Curso2 aux = new Curso2(
                                ""+obj.get("_id"),
                                ""+ obj.get("nome"),
                                ""+ obj.get("duracao"));
        System.out.println("GET ALL: " + aux);
}
```

# Retornar um objeto específico

- Retorna um objeto específico de acordo com filtros

- **De acordo com o id**

```
Bson filter2 = Filters.eq("_id", new ObjectId("5cffdfab1eabc52ba8c7cb89"));
```

- **De acordo com o nome**

```
Bson filter1 = Filters.eq("nome", "alex");
```

- **De acordo com o id e nome**

```
Bson filterResult = Filters.and(filter1, filter2);
```

# Retornar um objeto específico

- **De acordo com o id e nome e duração**

```
Bson filter3 = Filters.eq("duracao", "1234");
Bson filterResult2 = Filters.and(filterResult, filter3);
```

# Retornar um objeto específico

- **Depois de criado os filtros, executa da mesma forma que o de retornar todos**

```
find = collection.find(filterResult2).iterator();
list = new ArrayList<>();
while (find.hasNext()) {
	Document obj = find.next();
	Curso2 aux = new Curso2(
			""+obj.get("_id"),
			""+ obj.get("nome"),
			""+ obj.get("duracao"));
	System.out.println("GET ID: " + aux);
}
```

# Atualizar um objeto específico

- **Baseado nos filtros criados, é possível atualizar um objeto específico**

```
BasicDBObject update = new BasicDBObject("$set",
                    new BasicDBObject("nome", "alex").append("duracao", "1234"));
collection.updateOne(filterResult2, update);
```

# Deletar um objeto específico

- **Baseado nos filtros criados, é possível deletar um objeto específico**

  collection.deleteOne(filterResult2);

# Código-fonte completo

- https://pastebin.com/7ZAYfcdw

# Dúvidas??

E-mail: almada@crateus.ufc.br