Expresso São Miguel - Setor de T.I

Integração com API do Mongo

05 de dezembro de 2019

VISÃO GERAL

Biblioteca Java e Delphi responsável pela integração de aplicações com a API do MongoDB.

OBJETIVOS

1. Prover os métodos de manipulação de dados do MongoDB.
2. Centralizar conexões com o Banco MongoDB em todas as aplicações Expresso São Miguel.
3. Facilitar a manipulação de dados no MongoDB.

ESPECIFICAÇÕES

Esta biblioteca de integração está desenvolvida em Java e Delphi. Utilizada através da importação do .jar ou pelo pom.xml em projetos java, e pela utilização do ExpComponents em Delphi.

MÉTODOS E UTILIZAÇÃO

Pré configuração necessária

A API do MongoDB utiliza a API de Configurações para buscar quais são as configurações de conexão com o Banco MongoDB. Sendo assim é obrigatório cadastrar uma configuração para a aplicação que você está utilizando.

Essa configuração é feita tanto no ambiente de homologação (172.16.0.174) quanto no ambiente de produção(172.16.0.161) nos bancos SistemaEXP.

1. Cadastre uma aplicação na tabela T_APLICACAO_CONFIGURACAO.
2. Cadastre as configurações de conexão com o MongoDB na tabela T_APLICACAO_CONFIGURACAO_VALOR.
    2.1. **Obrigatoriamente** as chaves das configurações devem ser padronizadas da seguinte forma:
        - MONGO_HOST: Endereço de conexão com o MongoDB Server.
        - MONGO_PORTA: Porta de conexão com o MongoDB Server Padrão 27017.
        - MONGO_DATABASE: Nome da base de dados do MongoDB.
        - MONGO_USUARIO: Usuário para autenticar. **Se existir.**
        - MONGO_SENHA: Senha para autenticar. **Se existir.**
3. **CADASTRE EM AMBOS OS AMBIENTES UTILIZANDO AS MESMAS CHAVES NA TABELA T_APLICACAO_CONFIGURACAO.**
4. Apenas mude as configurações em T_APLICACAO_CONFIGURACAO_VALOR para Homologação e Produção.

Instalação

Delphi

Está disponível dentro do pacote de componentes *ExpComponents*. Baixe a versão mais recente do Trunk e compile a biblioteca.

Após compilar, em seu projeto, adicione a pasta da biblioteca dentro do Search Path de nível de projeto.

Os fontes da biblioteca estão disponíveis em: “Delphi\Components\MongoDB”.

Java

Biblioteca está disponível juntamente com o pacote expressosaomiguel-core em duas versões similares, apenas com suporte a JDKs diferentes (Consulte a versão mais recente de cada versão com o responsável).

Há duas maneiras de se utilizar a biblioteca.

1. Em projetos baseados em Maven, adicione a dependência ao arquivo pom.xml.
    1.1. Adicione ao pom.xml o expressosaomiguel-web-parent como parent para ter acesso a esta biblioteca e as demais produzidas pela Expresso São Miguel.
    1.2. Adicione apenas a biblioteca do mongo caso não queria ter todas as outras disponíveis, para isso é obrigatório informar qual a versão da biblioteca está sendo utilizada.
2. Em projetos java que não são baseados em Maven é necessário importar o .jar para dentro das dependências.

Configuração de Ambiente

Delphi

A biblioteca se baseia no modo de execução do projeto para saber em qual ambiente deve executar as consultas.

Executando a aplicação em DEBUG, a biblioteca vai se conectar no ambiente de homologação que está localizado em https://rkehomologacao.expressosaomiguel.com.br.

Executando a aplicação em RELEASE, a biblioteca vai se conectar no ambiente de produção que está localizado em https://apimongo.expressosaomiguel.com.br.

Java

Como em aplicações Java não podemos dizer em qual modo estamos executando, precisamos colocar manualmente no arquivo application.properties.

Se não existir um arquivo crie-o dentro da pasta resources do teu projeto.

Coloque a tag esm.mongo.buildType no arquivo.

Defina o valor em HOM para a biblioteca consultar o servidor de homologação.

Defina o valor em PROD para a biblioteca consultar o servidor de produção.

Utilização

Delphi / Java

TExpMongoIntegrator (Delphi) / MongoIntegratorBuilder (Java)

Responsável por configurar nossa integração com o banco do MongoDB e disponibiliza os métodos para interação com o Mongo (Salvar, Buscar, Deletar).

Delphi:

```
TExpMongoIntegrator.Builder
  .WithConfigurationId('408E1B66-D5D3-43D2-987D-4F3A1314D109')
  .WithCollection('TESTE');
```

Java:

```
MongoIntegratorBuilder.builder()
        .withCollection("TESTE")
        .withConfigurationId("408E1B66-D5D3-43D2-987D-4F3A1314D109");
```

- Builder (Delphi) / builder() (Java)
    - Retorna uma instância do TExpMongoIntegrator (Delphi) / MongoIntegratorBuilder (Java).
- WithConfigurationId (Delphi) / withConfigurationId (Java)
    - Parâmetro *ConfigurationId(Dephi) / configurationId(Java)* refere-se a chave de configuração existente na T_APLICACAO_CONFIGURACAO.
    - Retorna a instância do TExpMongoIntegrator(Delphi) / MongoIntegratorBuilder(Java).
- WithCollection (Delphi) / withCollection (Java)
    - Parâmetro *Collection* refere-se a coleção mongo na qual será executado os métodos.
    - Retorna a instância do TExpMongoIntegrator (Delphi) / MongoIntegratorBuilder (Java).

TExpMongoIntegratorSearchFile (Delphi) / MongoIntegratorSearch (Java)

Responsável por prover a função de download de arquivos do MongoDB.

Para acessar, deve-se através do TExpMongoIntegrator.Builder (Delphi) / MongoIntegratorBuilder.builder() (Java), após configura-lo, chamar o método Search(Delphi) / search() (Java).

Delphi:

```
TExpMongoIntegrator.Builder
    .WithConfigurationId('408E1B66-D5D3-43D2-987D-4F3A1314D109')
    .WithCollection('TESTE')
    .Search
    .Query(TMongoQuery.Query
      .Where('metadata.key')
      .InValue('38901c2b-9f9b-429a-9b76-aa89acf9006a')
      .AndValue('c6751f4f-9e7c-4341-a083-7304225fac17'))
    .Execute;
```

Java:

```
MongoIntegratorBuilder.builder()
        .withCollection("TESTE")
        .withConfigurationId("4F3A1314D109")
        .search()
        .query(query().where("metadata.key").in("key1").and("key2"))
        .execute();
```

- Query
    - Parâmetro ***ConditionCriteria (Delphi) / conditionCriteria (Java)*** recebe um objeto que **implementa** a interface IConditionCriteria (Delphi) / ConditionCriteria (Java). Neste caso já como padrão temos a classe TMongoQuery.Query (Delphi) / MongoQuery.query() (Java) que retorna um objeto desta interface.
        - Where (Delphi) / where() (Java)
            - Parâmetro **Attribute (Delphi) / attribute (Java)** refere-se a qual o campo do documento disponível no mongo a consulta será feita.
        - InValue (Delphi) / in (Java)
            - Parâmetro **Value (Delphi) / value (Java)** refere-se ao valor que será buscado.
        - AndValue (Delphi) / and (Java)
            - Parâmetro **Value (Delphi) / value (Java)** refere-se a outro valor que será buscado, como uma espécie de WHERE **Attribute** IN **(Value1, Value2)**.

- Execute (Delphi) / execute() (Java)
    - Executa a requisição na api do MongoDB.
    - Retorna um objeto do tipo TSearchFilesResponse(Delphi) / SearchFilesResponse(Java)

TSearchFilesResponse (Delphi) / SearchFilesResponse (Java)

Objeto de retorno da execução do Search. Temos nele uma lista de TSearchFilesResponseObject (Delphi) / SearchFilesResponseObject (Java) que nada mais é que os arquivos do MongoDB.

Contém o Base64/base64, Filename/filename, ContentType/contentType e os atributos do Metadata/metadata em um TDictionary/HashMap.

Status/status refere-se ao status da integração True para sucesso, False para erro.

Message/message refere-se a mensagem de retorno da integração.

TExpMongoIntegratorDeleteFile (Delphi) / MongoIntegratorDeleteFile (Java)

Responsável por prover a função de remover arquivos do MongoDB.

Para acessar, deve-se através do TExpMongoIntegrator.Builder (Delphi) / MongoIntegratorBuilder.builder() (Java), após configurá-lo, chamar o método DeleteFiles (Delphi) / deleteFiles() (Java).

Delphi:

```
TExpMongoIntegrator.Builder
    .WithConfigurationId('408E1B66-D5D3-43D2-987D-4F3A1314D109')
    .WithCollection('TESTE')
    .DeleteFiles
    .WithObjectId(True)
    .AddFileId('5de1112d8345441eecb86483')
    .Execute;

TExpMongoIntegrator.Builder
  .WithConfigurationId('408E1B66-D5D3-43D2-987D-4F3A1314D109')
  .WithCollection('TESTE')
  .DeleteFiles
  .WithFieldForSearch('metadata.key')
  .AddFileId('c6751f4f-9e7c-4341-a083-7304225fac17')
  .Execute;
```

Java:

```
MongoIntegratorBuilder.builder()
        .withCollection("TESTE")
        .withConfigurationId("4F3A1314D109")
        .deleteFiles()
        .withFieldForSearch("metadata.key")
        .addFileId("key")
        .execute();

MongoIntegratorBuilder.builder()
        .withCollection("TESTE")
        .withConfigurationId("4F3A1314D109")
        .deleteFiles()
        .withObjectId(Boolean.TRUE)
        .addFileId("id")
        .execute();
```

Há duas maneiras de executar o delete. Passando por qual field da collection a pesquisa deve ocorrer, ou fazendo diretamente pelo atributo _id da collection (Padrão de toda a collection MongoDB). Isso dependerá de qual chave você estará usando para deletar.

- WithObjectId (Delphi) / withObjectId (Java)
    - Parâmetro **DeleteByObjectId (Delphi) / deleteByObjectId (Java)** passado como True faz com que o método *WithFieldForSearch (Delphi) / withFieldForSearch (java)* não seja obrigatório e no método *AddFileId (Delphi) / addFieldId (Java)* deve ser o Id do arquivo no MongoDB (Atributo _id) sem o "ObjectId()", apenas a chave.

- WithFieldForSearch (Delphi) / withFieldForSearch (Java)
    - Parâmetro **FieldForSearch (Delphi) / fieldForSearch (Java)** é apenas necessário caso o WithObjectId seja False. Refere-se a qual field o mongo executará a consulta e posteriormente o delete.
- AddFileId (Delphi) / addFileId (Java)
    - Parâmetro **FileId (Delphi) / fileId (Java)** refere-se ao valor de busca no MongoDB. Se o WithObjectId for True é obrigatório passar o _id aqui, somente chave sem o "ObjectId()".
- Execute
    - Executa a requisição na api do MongoDB.
    - Retorna um objeto do tipo TDeleteFileResponse (Delphi) / DeleteFileResponse (Java)..

TDeleteFileResponse(Delphi) / DeleteFileResponse (Java)

Objeto de retorno da execução do DeleteFiles. Temos duas propriedades.

Status refere-se ao status da integração True para sucesso, False para erro.

Message refere-se a mensagem de retorno da integração.

TExpMongoIntegratorSaveFile (Delphi) / MongoIntegratorSaveFile (Java)

Responsável por prover a função de fazer Upload de arquivos do MongoDB.

Para acessar, deve-se através do TExpMongoIntegrator.Builder (Delphi) / MongoIntegratorBuilder.builder() (Java), após configurá-lo, chamar o método SaveFiles (Delphi) / saveFiles() (Java).

Delphi:

```
TExpMongoIntegrator.Builder
  .WithConfigurationId('408E1B66-D5D3-43D2-987D-4F3A1314D109')
  .WithCollection('TESTE')
  .SaveFiles
  .AddFile(TMongoFile.New
      .WithKey('Key')
      .WithFilename('TesteKelvin.png')
      .WithContentType('.png')
      .WithBase64(B64)
      .AddAttributeMetadata('usuario', 'kelvin')
      .AddAttributeMetadata('data', '04/12/2019 11:13'))
  .Execute;
```

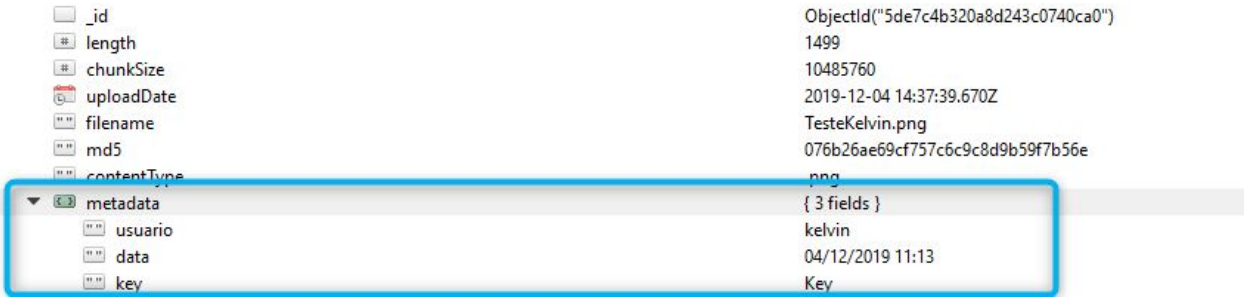
Java:

```
MongoIntegratorBuilder.builder()
        .withCollection("TESTE")
        .withConfigurationId("408E1B66-D5D3-43D2-987D-4F3A1314D109")
        .saveFiles()
        .addFile(
                file().withName("ArquivoEnviadoTeste.jpg")
                        .withContentType(".jpg")
                        .withKey("key")
                        .withBase64("base64")
                        .addAttributeMetadata("chave", "valor")
        )
        .execute();
```

- AddFile (Delphi) / addFile (Java)
    - Parâmetro **MongoFile(Delphi) / mongoFile(Java)** refere-se a objeto do tipo TMongoFile(Delphi) / MongoFile(Java) que nada mais é que os dados do arquivo a ser enviado ao MongoDB.
        - TMongoFile(Delphi) / MongoFile(Java)
            - New(Delphi) / file(Java)
                - Retorna uma instância de um TMongoFile(Delphi) / MongoFile(Java).

- WithKey
    - Parâmetro **Key** refere-se a chave que será salva em metdata.key dentro do objeto no mongo. Padronizado para todas as coleções terem esse campo. Passando ele, o mongo salvará com a key informada. Se não passar nada o Mongo irá criar um UUID ramdom e devolverá.
- WithFilename
    - Parâmetro **Filename** refere-se ao nome do arquivo que está sendo enviado ao Mongo.
- WithContentType
    - Parâmetro **ContentType** refere-se a extensão do arquivo. Exemplo: “.jpg” “.png” “.pdf”.
- WithBase64
    - Parâmetro **Base64** refere-se a string base64 do arquivo a ser enviado.

- AddAttributeMetadata
    - Refere-se ao objeto Metadata que será salvo no documento no MongoDB.
    - Parâmetro **Key** refere-se a chave.
    - Parâmetro **Value** refere-se ao valor da chave.
    - Lembrando que juntamente com os dados que por aqui, vai ser salvo junto o atributo **Key** na Metadata.



- Execute
    - Executa a requisição na api do MongoDB.
    - Retorna uma lista de objetos do tipo TSaveFileResponse(Delphi) / SaveFileResponse(Java).

TSaveFileResponse(Delphi) / SaveFileResponse(Java)

Objeto de retorno da execução do SaveFiles.

Status refere-se ao status da integração True para sucesso, False para erro.

Message refere-se a mensagem de retorno da integração.

Data é um objeto do tipo TSaveFileResponseObject(Delphi) / SaveFileResponseObject(Java).

TSaveFileResponseObject(Delphi) / SaveFileResponseObject(Java)

Id refere-se ao ObjectId gerado pelo MongoDB.

Filename é o nome do arquivo envaido ao Mongo.

Key é a chave salva em metadata.key.