Unisanta Science and Technology, **2016, 6, December**
Published Online 2016 Vol.5 N°2 http://periodicos.unisanta.br/index.php/sat)

UNISANTA
Science and Technology
ISSN 2317-1316

Generous: *Application Programming Interface* (API) para Mapeamento de Objeto no MongoDB

[1]Adriano Santos da Cruz & [2]Marcelo Pereira Bergamaschi

[1]**UNISANTA** – Universidade Santa Cecília
Rua Oswaldo Cruz, 266 – Boqueirão – Santos/SP

[2]**IFSP**- Instituto Federal de Educação, Ciência e Tecnologia de São Paulo
Rua Maria Cristina, 50 - Jardim Casqueiro – Cubatão/SP

berga@ifsp.edu.br

Received november 2015

Resumo: Neste trabalho descreve-se a procura por uma *Application Programming Interface* (API) em Java que contivesse as funcionalidades para persistir dados no banco de dados MongoDB, trabalhar com conversão de "objetos para *JSON*" ou de "*JSON* para objeto" e gerar *log* de erros da aplicação foi frequente. As API's pesquisadas atendem determinadas funcionalidades isoladamente e não como uma única interface, por este motivo houve a necessidade de criar uma API para persistir dados como se fossem objetos, o "Generous" será um projeto *open source* (programa de código aberto) onde nele é possível mapear estes objetos sem a necessidade de se utilizar *annotations* (anotações que dão informações sobre o código que está sendo escrito) em classes Java apenas para o MongoDB que é um banco de dados orientado a documentos que salva os registros em um formato *JSON*.

Palavras chave: Generous, API, open source, mapeamento de objeto, banco de dados MongoDB.

Generous: Application Programming Interface (API) for Object Mapping in MongoDB

Abstract: In this work we describe the search for an *Application Programming* Interface (API) in Java that contains the functionality to persist data in MongoDB database, working with conversion "objects to *JSON*" or "*JSON* to object" and generate application error *log* was common. The API's surveyed meet certain features in isolation and not as a single interface, therefore it was necessary to create an API to persist data as objects, the "Generous" is an *open source* project (open source program) where it is can map these objects without the need to use *annotations* (notes which give information about the code being written) in Java classes just for MongoDB is a database-driven documents that saves records in a *JSON* format.

Keywords: Generous, API, open source, object mapping, MongoDB database.

1. Introdução

As API's (*Application Programming Interface* ou em português *Interface* de Programação de Aplicação) são conjuntos de rotinas e padrões estabelecidos pelo desenvolvedor do *software* para utilização dos métodos que o compõe e que são somente acessados através de programação.

Atualmente vem se utilizando dessa tecnologia, pois agrega valor ao usuário final, ao programador e a própria aplicação que será desenvolvida, além disso, possibilita diversas formas de interação e torna mais fácil a utilização de funções rotineiras que acabam se tornando padrões.

Para se ter noção da importância da utilização delas até os Sistemas operacionais utilizam as funcionalidades de API's.

Sua utilização não se limita somente em versão *desktop*, a funcionalidade da API em *web* pode ser ainda maior. Existem diversas empresas que produzem esse tipo de aplicação e disponibilizam seus códigos para serem utilizados em outros *sites*. Um bom exemplo de API *web* é o *Google Maps*, pois diversos outros sites utilizam este serviço dentro de suas páginas, usando o código original e adaptando-o da maneira mais conveniente.

Enfim, API's diferentes estão presentes em navegadores, aplicativos de diversas linguagens e para variadas finalidades [CANALTECH, 2014].

A "API Generous" foi desenvolvida com ferramentas *open source* na linguagem Java para mapeamento de objeto orientado a documentos no MongoDB, que tem como objetivo tornar mais fácil a codificação de programas que utilizem esse banco de dados e que sejam de fácil manuseio pelo os seus usuários para utilizar as principais funções básicas.

2. Materiais e Métodos

A "API Generous" foi desenvolvida no Sistema Operacional Windows 7 Ultimate, mas é compatível em outros sistemas operacionais, foi utilizada a linguagem de programação Java com a *Integrated Development Environment* (IDE) "*Eclipse Luna*" versão de desenvolvedor Luna Service Release 1 (4.4.1) e versão do Java 1.8.0_40, o banco de dados escolhido foi o MongoDB versão mongodb-win32-x86_64-2008plus-2.6.6-signed IDE Robomongo 0.8.4 e para enriquecer o projeto foi agregada quatro API's já existentes que são: Gson-2.2.4, log4j-1.2.17, log4mongo-java-0.7.4 e mongo-java-driver-2.12.3.

O desenvolvimento deste projeto surgiu, primeiramente da necessidade de uma API deste gênero em projetos pessoais, e que foi utilizado como base outras API's e muitas pesquisas de trabalhos correlatos como o Hibernate OGM, para ter como apoio no processo de ideia e criação do "Generous" [BERNARD, 2015].

3. Discussão

O "Generous" basicamente é uma junção de quatro API's com algumas funções e/ou adaptações necessárias para futuros projetos e é somente uma versão experimental já que possui algumas funcionalidades a serem acrescentadas que não estão na versão atual 1.0.

Esta API desenvolvida tem a pretensão de se tornar um projeto *open source*, a princípio foi por que para desenvolvê-la foi utilizado ferramentas deste gênero e também para que a comunidade possa acrescentar e expor suas impressões a respeito desta aplicação.

Para entender melhor sua funcionalidade serão expostas as funcionalidades das API's que foram acopladas ao projeto [UNIVERSIA BRASIL, 2014].

O **Gson** é uma biblioteca Java do Google onde seus principais objetivos é fornecer funções simples para converter objetos Java para *JSON* e vice-versa, amplo suporte de Java Generics, permitir representações personalizadas para objetos e suporte para objetos arbitrariamente complexas com hierarquias de herança profunda e ampla utilização de tipos genéricos [STUDYTRAILS, 2014].

O **log4j** é uma biblioteca *open source* da Apache que possui três componentes principais que são os: *loggers*, *appenders* e *layouts* onde estes três tipos de componentes trabalham juntos para permitir que os desenvolvedores possam registrar mensagens de acordo com o tipo de mensagens e nível, e serve também para controlar em tempo de execução como essas mensagens serão formatadas e onde serão apresentadas [GÜLCÜ, 2012].

O **log4mongo-java** é uma biblioteca de código aberto que inclui vários Log4J appenders para armazenar mensagens de log em uma collection MongoDB, que são: MongoDbAppender que serve para armazenar ou registrar eventos numa forma BSONified, ExtendedMongoDbAppender que é uma extensão do MongoDbAppender, mas que permite adicionar elementos de nível superior e por último MongoDbPatternLayoutAppender que suporta dados de log em um formato personalizado [ATLASSIAN, 2015].

O **mongo-java-driver** é uma biblioteca *open source* da MongoDB University para o banco de dados orientado a documentos projetados para facilidade de desenvolvimento e de escala, oferece alta performance e alta disponibilidade, contendo mongodb-driver, mongodb-driver-core, e bson [MONGODB, 2015]. Com todos esses recursos dito pelos projetos que foram inseridos no "Generous" que foi possível começar a desenvolver esse projeto, deixando as suas funcionalidades o mais genérica possível para serem utilizadas por outros desenvolvedores. Para começar a utilizar os recursos dele deve-se configurar programaticamente algumas informações a respeito do banco de dados MongoDB. Feito essas configurações as funções básicas já estarão acessíveis. Nesta versão do "Generous" obrigatoriamente o usuário deverá passar o login e senha do banco de dados.

```
String HOST="localhost";
String PORT="27017";
String DB_NAME="GenerousDB";
String DB_LOGIN="Generous";
String DB_PASSWORD="Generous123";

// Essa classe é responsável por gerar e alterar arquivos properties.
FileProperties properties = new FileProperties();
// Função para alterar os valores do Banco MongoDB dentro do arquivo connect.properties.
properties.setValuesConnect(HOST, PORT, DB_NAME, DB_LOGIN, DB_PASSWORD);
```


 Unisanta Science and Technology **(89-95) p.p. ISSN 2317-1316**

Há a possibilidade também de recuperar ou alterar informação de uma propriedade em específico ou de todas, e limpar todos os valores de uma propriedade no arquivo ou apenas um.

```
// Informações das propriedades relacionadas ao Banco de Dados do MongoDB.
System.out.println(FileProperties.propertiesConnect.HOST.getValue());
System.out.println(FileProperties.propertiesConnect.PORT.getValue());
System.out.println(FileProperties.propertiesConnect.DB_NAME.getValue());
System.out.println(FileProperties.propertiesConnect.DB_LOGIN.getValue());
System.out.println(FileProperties.propertiesConnect.DB_PASSWORD.getValue());

 localhost
 27017
 GenerousDB
 Generous
 Generous123
// Função responsável por alterar uma informação de uma propriedade específica
// dentro do arquivo connect.properties.
FileProperties.propertiesConnect.HOST.setValue("NewLocalhost");
FileProperties.propertiesConnect.PORT.setValue("20000");
FileProperties.propertiesConnect.DB_NAME.setValue("NewGenerousDB");
FileProperties.propertiesConnect.DB_LOGIN.setValue("NewLogin");
FileProperties.propertiesConnect.DB_PASSWORD.setValue("NewPassword");

System.out.println(FileProperties.propertiesConnect.HOST.getValue());
System.out.println(FileProperties.propertiesConnect.PORT.getValue());
System.out.println(FileProperties.propertiesConnect.DB_NAME.getValue());
System.out.println(FileProperties.propertiesConnect.DB_LOGIN.getValue());
System.out.println(FileProperties.propertiesConnect.DB_PASSWORD.getValue());

 NewLocalhost
 20000
 NewGenerousDB
 NewLogin
 NewPassword
// Função para recuperar todas as informações do Arquivo connect.properties.
System.out.println(new FileProperties().getAllValuesConnect());

[HOST: NewLocalhost, PORT: 20000, DB_NAME: NewGenerousDB, DB_LOGIN: NewLogin, DB_PASSWORD: NewPassword]


// Função responsável por limpar todas as informações contidas no arquivo connect.properties.
new FileProperties().resetValuesConnect();
System.out.println(new FileProperties().getAllValuesConnect());

[HOST: null, PORT: null, DB_NAME: null, DB_LOGIN: null, DB_PASSWORD: null]

// Função para alterar uma propriedade.
FileProperties.propertiesConnect.HOST.setValue("localhost");
FileProperties.propertiesConnect.PORT.setValue("27017");
System.out.println(new FileProperties().getAllValuesConnect());
[HOST: localhost, PORT: 27017, DB_NAME: null, DB_LOGIN: null, DB_PASSWORD: null]
```

Outra função importante que o "Generous" oferece é a opção do desenvolvedor escolher o *log* (histórico de utilização) em um arquivo de *log*, no banco ou nos dois arquivos.


```
String hostname="localhost";
String port="27017";
String databaseName="GenerousDB";
String collectionName="Log";
String userName="Generous";
String password="Generous123";
String conversionPattern="{\"Milisegundos\": \"%r\",\"Classe\": \"%C\" ,\"Thread\": \"%t\" ,"
        + "\"Data\": \"%d{dd/MM/yyyy}\" , \"Hora\": \"%d{HH:mm:ss}\" ,\"Local\": \"%l\", "
        + "\"Prioridade\": \"%p\", \"Mensagem\": \"%m\"}";
String pathFile = "log/app.log";
String conversionPatternLog="Milisegundos desde o início do programa: %r %nClasse: "
        + "%C %nThread: %t %nData: %d{dd/MM/yyyy} %nHora: %d{HH:mm:ss} %nLocal: %l "
        + "%nPrioridade: %p %nMensagem: %m %n-----------------------------------------%n";
FileProperties properties = new FileProperties();
// Função para alterar as informações do arquivo log4j.properties referente a Log no Banco de Dados.
properties.setValuesLogDatabase(hostname, port, databaseName, collectionName, userName,
        password, conversionPattern);
// Função para alterar as informações do arquivo log4j.properties referente a Log em um Arquivo.
properties.setValuesLogFile(pathFile, conversionPatternLog);
// Função para alterar todas as propriedades do arquivo log4j.properties.
properties.setAllValuesLog(hostname, port, databaseName, collectionName, userName, password,
        conversionPattern, pathFile, conversionPatternLog);
```

Podendo também recuperar ou alterar informação de uma propriedade em específico ou de todas, e limpar todos os valores ou apenas um de *log* na base ou em um arquivo.

```
// Função responsável por recuperar uma informação de uma propriedade específica
// dentro do arquivo log4j.properties.
System.out.println(FileProperties.propertiesLog.MongoDB_databaseName.getValue());
System.out.println(FileProperties.propertiesLog.MongoDB_hostName.getValue());
System.out.println(FileProperties.propertiesLog.MongoDB_port.getValue());
System.out.println(FileProperties.propertiesLog.MongoDB_collectionName.getValue());
System.out.println(FileProperties.propertiesLog.MongoDB_userName.getValue());
System.out.println(FileProperties.propertiesLog.MongoDB_password.getValue());
System.out.println(FileProperties.propertiesLog.MongoDB_conversionPattern.getValue());
System.out.println(FileProperties.propertiesLog.FileOut_pathFile.getValue());
System.out.println(FileProperties.propertiesLog.FileOut_conversionPattern.getValue());

GenerousDB
localhost
27017
Log
Generous
Generous123
{"Milisegundos": "%r","Classe": "%C" ,"Thread": "%t" ,"Data": "%d{dd/MM/yyyy}" , "Hora": "%d{HH:mm:ss}" ,
log/app.log
Milisegundos desde o início do programa: %r %nClasse: %C %nThread: %t %nData: %d{dd/MM/yyyy} %nHora: %d{H
// Função responsável por alterar uma informação de uma propriedade específica
// dentro do arquivo log4j.properties.
FileProperties.propertiesLog.MongoDB_databaseName.setValue("NewDatabase");
FileProperties.propertiesLog.MongoDB_hostName.setValue("NewLocalhost");
FileProperties.propertiesLog.MongoDB_port.setValue("20000");
FileProperties.propertiesLog.MongoDB_collectionName.setValue("NewLog");
FileProperties.propertiesLog.MongoDB_userName.setValue("NewUser");
FileProperties.propertiesLog.MongoDB_password.setValue("NewPassword");
FileProperties.propertiesLog.MongoDB_conversionPattern.setValue("{\"Milisegundos\": \"%r\"}");
FileProperties.propertiesLog.FileOut_pathFile.setValue("newlog/newapp.log");
FileProperties.propertiesLog.FileOut_conversionPattern.setValue("início do programa: %r %n");
```


Unisanta Science and Technology **(89-95) p.p. ISSN 2317-1316**

```
  // Função responsável por recuperar uma informação de uma propriedade específica
  // dentro do arquivo log4j.properties.
  System.out.println(FileProperties.propertiesLog.MongoDB_databaseName.getValue());
  System.out.println(FileProperties.propertiesLog.MongoDB_hostName.getValue());
  System.out.println(FileProperties.propertiesLog.MongoDB_port.getValue());
  System.out.println(FileProperties.propertiesLog.MongoDB_collectionName.getValue());
  System.out.println(FileProperties.propertiesLog.MongoDB_userName.getValue());
  System.out.println(FileProperties.propertiesLog.MongoDB_password.getValue());
  System.out.println(FileProperties.propertiesLog.MongoDB_conversionPattern.getValue());
  System.out.println(FileProperties.propertiesLog.FileOut_pathFile.getValue());
  System.out.println(FileProperties.propertiesLog.FileOut_conversionPattern.getValue());

NewDatabase
NewLocalhost
20000
NewLog
NewUser
NewPassword
{"Milisegundos": "%r"}
newlog/newapp.log
início do programa: %r %n

// Função responsável por recuperar as informações contidas no
// arquivo log4j.properties referente a Log no Banco de Dados.
System.out.println(new FileProperties().getValuesLogDatabase());

hostname: NewLocalhost,
port: 20000,
databaseName: NewDatabase,
collectionName: NewLog,
userName: NewUser,
password: NewPassword,
conversionPattern: {"Milisegundos": "%r"}

// Função responsável por recuperar as informações contidas no
// arquivo log4j.properties referente a Log em um Arquivo.
System.out.println(new FileProperties().getValuesLogFile());

pathFile: newlog/newapp.log,
conversionPattern: início do programa: %r %n

// Função para retornar todos os valores dentro do arquivo log4j.properties.
System.out.println(new FileProperties().getAllValuesLog());

LogDatabase[
hostname: NewLocalhost,
port: 20000,
databaseName: NewDatabase,
collectionName: NewLog,
userName: NewUser,
password: NewPassword,
conversionPattern: {"Milisegundos": "%r"}
]
LogFile[
pathFile: newlog/newapp.log,
conversionPattern: início do programa: %r %n
]
```



```
// Função responsável por limpar as informações contidas no
// arquivo log4j.properties referente a Log no Banco de Dados.
new FileProperties().resetValuesLogDatabase();
System.out.println(new FileProperties().getValuesLogDatabase());

hostname: null,
port: null,
databaseName: null,
collectionName: null,
userName: null,
password: null,
conversionPattern: null

// Função responsável por limpar as informações contidas no
// arquivo log4j.properties referente a Log em um Arquivo.
new FileProperties().resetValuesLogFile();
System.out.println(new FileProperties().getValuesLogFile());

pathFile: null,
conversionPattern: null

// Função responsável por limpar todas as informações contidas no arquivo log4j.properties.
new FileProperties().resetAllLogs();
System.out.println(new FileProperties().getAllValuesLog());

LogDatabase[
hostname: null,
port: null,
databaseName: null,
collectionName: null,
userName: null,
password: null,
conversionPattern: null
]
LogFile[
pathFile: null,
conversionPattern: null
]
```

No "Generous" existem também dois métodos *save* onde cada um recebe um tipo de parâmetro diferente para salvar registros na base de dados, *find*, *findOne*, *findAll* e *getRegister*, onde cada um desses possui uma maneira diferente para localizar registros, dois métodos *update* em que um recebe duas Strings e outro dois objetos do tipo BacicDBObject para atualizar registros e *delete* que também possui dois métodos onde um recebe um objeto de uma classe e outro uma *String* para excluir registros da base. A maneira como são utilizados esses métodos poderão ser analisados pelos os usuários desta API através de um Javadoc (É um utilitário fornecido pela Sun Microsystems junto ao JDK para gerar documentação de códigos Java em formato *HTML* a partir de comentários no código fonte Java) com todas as informações detalhadas referentes a estes métodos comentados acima [ORACLE, 2004].

Para finalizar, existe uma classe chamada *Utils* que dispõe de funcionalidades que desenvolvedores costumam utilizar, *convertToJson* (converte um objeto em *JSON*), *convertToObject* (converte um *JSON* para objeto da classe passada por parâmetro) e *Log* (que possui níveis de log para criar histórico de utilização).

4. Resultados

Sem dúvida a utilização das API's tanto em projetos de desenvolvedores ou em empresas já são constantes, devido as suas vantagens.

> Uma API contém os meios com os quais o desenvolvedor do software fornece acesso aos dados ou processos de um programa para outro. A Documentação de API lista os métodos e exigências para os processos de interface e de

> troca de dados. Com o crescimento da *internet* de aplicativos para dispositivos móveis e do projeto orientado ao serviço dos aplicativos, as API tornaram-se onipresentes. (JOHNSON, 2014).

O "Generous" vem atendendo as atuais necessidades de projetos paralelos como o próprio trabalho de conclusão de curso na área de Sistemas de Informação mesmo sendo apenas um projeto experimental. O diferencial desta API é que o desenvolvedor que a utiliza consegue de forma intuitiva configurar e utilizar suas funções básicas sem grandes problemas. Além disso, não é preciso a utilização de *annotations* para definir que uma classe é uma representação de uma tabela do banco de dados MongoDB, vários métodos para alteração de informações referentes ao banco, *log* no banco ou em um arquivo em tempo de execução, o usuário não precisa se preocupar com CRUD uma vez que todas essas funcionalidades já estão inclusas nesta API [XAVIER, 2010].

É importante destacar também que de acordo com as necessidades de se encontrar uma API com estas características é confirmado que foi possível atingir o objetivo geral e específico conforme pesquisas.

5. Conclusão

De acordo com testes experimentais em projetos pessoais o "Generous" se mostrou eficiente atendendo bem as expectativas esperadas, mas a pretensão é de tornar esse projeto *open source* para que a comunidade desenvolvedora possa contribuir e torná-lo ainda maior e realizar alguns ajustes que ainda não foram explorados nesta primeira versão. Tendo uma API com uma única interface e que consiga realizar as funções de CRUD sem a necessidade de utilizar *annotations*, *log* em um arquivo ou em um banco de dados MongoDB que poderá ser utilizado como auditoria e conversão de objeto para formato *JSON* ou vice-versa facilitou o trabalho dos desenvolvedores na hora de codificarem programas. Portanto, esta linha de pesquisa foi satisfatória tanto na aprendizagem com a linguagem de programação Java, conceitos para se ter uma API amigável e o próprio banco de dados MongoDB quanto no próprio desenvolvimento do "Generous" conseguindo tornar o projeto uma realidade.

Referencias

ATLASSIAN. **Log4mongo for Java**. 2015. Disponível em: <https://log4mongo.atlassian.net/wiki/display/LOG4MONGO/Log4mongo+for+Java>. Acesso em: 08 abr. 2015.

BERNARD, E., GRINOVERO, S., MORLING, G.r, D'ALTO, D. **Hibernate OGM Reference Guide**. 2015. Disponível em: <http://docs.jboss.org/hibernate/ogm/4.1/reference/en-US/html/> Acesso em: 10 abr. 2015.

CANALTECH. **O que é API?**. 2014, Disponível em: <http://canaltech.com.br/o-que-e/software/O-que-e-API/>. Acesso em: 18 abr. 2015.

GÜLCÜ, C.. **The complete log4j manual**. 2012. Disponível em: <https://logging.apache.org/log4j/1.2/manual.html>. Acesso em: 06 abr. 2015.

JOHNSON, G.. **Definição e classificação por tipo da Interface de Programação de Aplicativos (API)**. 2014. Disponível em: <http://blog.magicsoftware.com.br/definicao-e-classificacao-por-tipo-da-interface-de-programacao-de-aplicativos-api/>. Acesso em: 25 abr. 2015.

MONGODB. **The MongoDB 3.0 Manual**. 2015. Disponível em: <http://docs.mongodb.org/manual/>. Acesso em: 11 abr. 2015.

ORACLE. **Javadoc Tool**. 2004. Disponível em: <http://www.oracle.com/technetwork/articles/java/index-jsp-135444.html>. Acesso em: 26 abr. 2015.

STUDYTRAILS. **Java Google Json (Gson) Introduction**. 2014. Disponível em: <https://code.google.com/p/google-gson/>. Acesso em: 06 abr. 2015.

UNIVERSIA BRASIL. **Saiba o que é open source e conheça a sua importância**. 2014. Disponível em: <http://noticias.universia.com.br/destaque/noticia/2014/04/01/1092713/saiba-e-open-source-e-conheca-sua-importancia.html>. Acesso em: 21 abr. 2015.

XAVIER, D. W.. **Tutorial completo.** 2010. Disponível em: <http://www.tiexpert.net/programacao/java/annotations.php>. Acesso em: 17 abr. 2015.

