SpringData MongoDB GridFsTemplate to save, retrieve, delete binary files (Image, Text files)

MongoDB provides GridFS for storing and retrieving large files with size limit of 16MB. So in the tutorial, JavaSampleApproach will show you how to work with **SpringData Mongo GridFsTemplate** to save, retrieve, delete binary files (Image, Text files).

Related posts:
– Bootstrap Image + MongoDB GridFsTemplate + SpringBoot RestAPI
– Spring MongoOperations to access MongoDB
– MongoDB – Model One-to-One, One-to-Many Relationships Embedded Documents | SpringBoot

Contents [hide]

I. Technologies

– Java: 1.8
– Maven: 3.3.9
– Spring Tool Suite: Version 3.8.4.RELEASE
– Spring Boot: 1.5.7.RELEASE
– MongoDB: version v3.4.9

II. SpringData Mongo GridFsTemplate

SpringData MongoDb provides `GridFsTemplate` class to work with **GridFs** file, details:

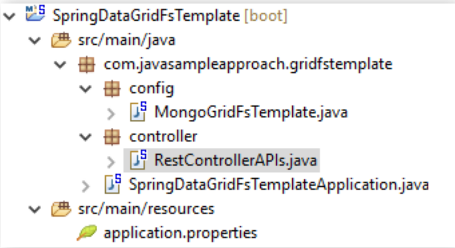
```
public class GridFsTemplate implements GridFsOperations, ResourcePatternResolver {
...

}
```

`GridFsTemplate` implement `interface GridFsOperations` that is the collection of operations to store and read files from MongoDB GridFS.

For convenient working, SpringBoot provides `spring-boot-starter-data-mongodb` dependency that is the collection of {`mongodb-driver`, `spring-data-mongodb`}.

III. Practice

In the tutorial, we create a SpringBoot project as below:



```
SpringDataGridFsTemplate [boot]
  src/main/java
    com.javasampleapproach.gridfstemplate
      config
        MongoGridFsTemplate.java
      controller
        RestControllerAPIs.java
      SpringDataGridFsTemplateApplication.java
  src/main/resources
    application.properties
```

Step to do:
– Create SpringBoot project
– Configure GridFsTemplate
– Implement RestAPIs to work with GridFs files
– Run & check results

1. Create SpringBoot project

Using **SpringToolSuite**, create a SpringBoot project, then add dependencies:

```
...

<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-mongodb</artifactId>
</dependency>
```

2. Configure GridFsTemplate

Using `AbstractMongoConfiguration` to configure `GridFsTemplate`:

```
package com.javasampleapproach.gridfstemplate.config;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.mongodb.config.AbstractMongoConfiguration;
import org.springframework.data.mongodb.gridfs.GridFsTemplate;

import com.mongodb.Mongo;
import com.mongodb.MongoClient;

@Configuration
public class MongoGridFsTemplate  extends AbstractMongoConfiguration{

  @Value("${jsa.mongo.address}")
  private String mongoAddress;

  @Value("${jsa.mongo.database}")
  private String mongoDatabase;

  @Bean
  public GridFsTemplate gridFsTemplate() throws Exception {
      return new GridFsTemplate(mongoDbFactory(), mappingMongoConverter());
  }

  @Override
  protected String getDatabaseName() {
    return mongoDatabase;
  }

  @Override
  public Mongo mongo() throws Exception {
    return new MongoClient(mongoAddress);
  }
}
```

Open **application.properties** file, add configuration:

```
jsa.mongo.address=127.0.0.1
jsa.mongo.database=jsa-mongodb
```

3. Implement RestAPIs to work with GridFs files

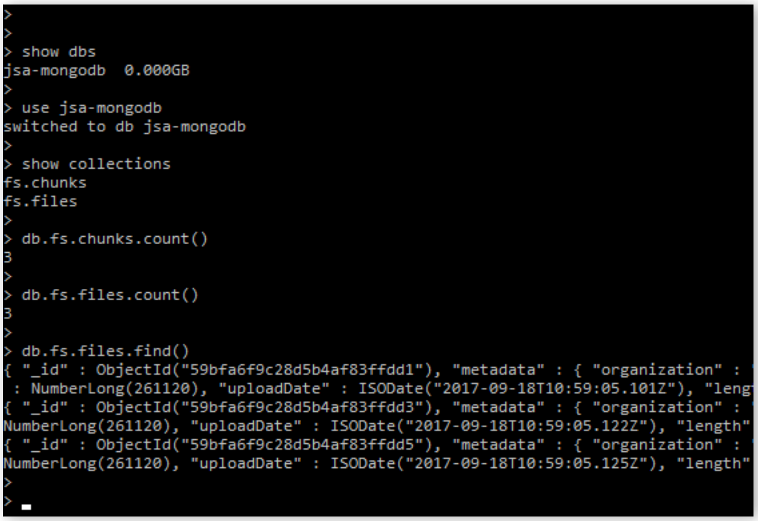
Using `GridFsOperations` to implement 4 RestfulAPIs { `/api/save`, `/api/retrieve/imagefile`, `/api/retrieve/textfiles`, `/api/delete/image` }:

```
package com.javasampleapproach.gridfstemplate.controller;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.mongodb.core.query.Criteria;
import org.springframework.data.mongodb.core.query.Query;
import org.springframework.data.mongodb.gridfs.GridFsOperations;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.mongodb.BasicDBObject;
import com.mongodb.DBObject;
import com.mongodb.gridfs.GridFSDBFile;

@RestController
@RequestMapping("/api")
public class RestControllerAPIs {

  @Autowired
  GridFsOperations gridOperations;

  // this variable is used to store ImageId for other actions like: findOne or delete
  private String imageFileId = "";

  @GetMapping("/save")
  public String saveFiles() throws FileNotFoundException {
    // Define metaData
    DBObject metaData = new BasicDBObject();
    metaData.put("organization", "JavaSampleApproach");

    /**
     * 1. save an image file to MongoDB
     */

    // Get input file
    InputStream iamgeStream = new FileInputStream("D:\\JSA\\jsa-logo.png");

    metaData.put("type", "image");

    // Store file to MongoDB
    imageFileId = gridOperations.store(iamgeStream, "jsa-logo.png", "image/png", metaData).getId().toString();
    System.out.println("ImageFileId = " + imageFileId);

    /**
     * 2. save text files to MongoDB
     */

    // change metaData
    metaData.put("type", "data");

    // Store files to MongoDb
    gridOperations.store(new FileInputStream("D:\\JSA\\text-1.txt"), "text-1.txt", "text/plain", metaData);
    gridOperations.store(new FileInputStream("D:\\JSA\\text-2.txt"), "text-2.txt", "text/plain", metaData);

    return "Done";
  }

  @GetMapping("/retrieve/imagefile")
  public String retrieveImageFile() throws IOException{
    // read file from MongoDB
    GridFSDBFile imageFile = gridOperations.findOne(new Query(Criteria.where("_id").is(imageFileId)));

    // Save file back to local disk
    imageFile.writeTo("D:\\JSA\\retrieve\\jsa-logo.png");

    System.out.println("Image File Name:" + imageFile.getFilename());

    return "Done";
  }

  @GetMapping("/retrieve/textfiles")
  public String retrieveTextFiles(){
    /**
     * get all data files then save to local disk
     */

    // Retrieve all data files
    List<GridFSDBFile> textFiles = gridOperations.find(new Query(Criteria.where("metadata.type").is("data")));

    // Save all back to local disk
    textFiles.forEach(file->{

      try {
        String fileName = file.getFilename();

        file.writeTo("D:\\JSA\\retrieve\\"+ fileName);

        System.out.println("Text File Name: " + fileName);

      } catch (IOException e) {
        e.printStackTrace();
      }
    });

    return "Done";
  }

  @GetMapping("/delete/image")
  public String deleteFile(){
    // delete image via id
    gridOperations.delete(new Query(Criteria.where("_id").is(imageFileId)));

    return "Done";
  }
}
```

4. Run & check results

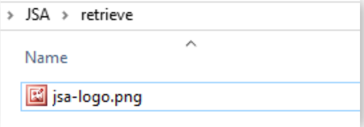
Build and Run the SpringBoot project with commandlines: { `mvn clean install`, `mvn spring-boot:run` }.

– Make a saving request: `http://localhost:8080/api/save`



```
>
>
> show dbs
jsa-mongodb  0.000GB
>
> use jsa-mongodb
switched to db jsa-mongodb
>
> show collections
fs.chunks
fs.files
>
> db.fs.chunks.count()
3
>
> db.fs.files.count()
3
>
> db.fs.files.find()
{ "_id" : ObjectId("59bfa6f9c28d5b4af83ffdd1"), "metadata" : { "organization" :
 : NumberLong(261120), "uploadDate" : ISODate("2017-09-18T10:59:05.101Z"), "leng
{ "_id" : ObjectId("59bfa6f9c28d5b4af83ffdd3"), "metadata" : { "organization" :
NumberLong(261120), "uploadDate" : ISODate("2017-09-18T10:59:05.122Z"), "length"
{ "_id" : ObjectId("59bfa6f9c28d5b4af83ffdd5"), "metadata" : { "organization" :
NumberLong(261120), "uploadDate" : ISODate("2017-09-18T10:59:05.125Z"), "length"
>
> 
```

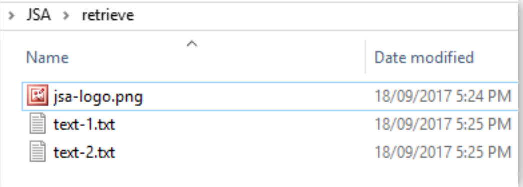
– Make a retrieving image request: `http://localhost:8080/api/retrieve/imagefile`



› JSA › retrieve

Name
jsa-logo.png

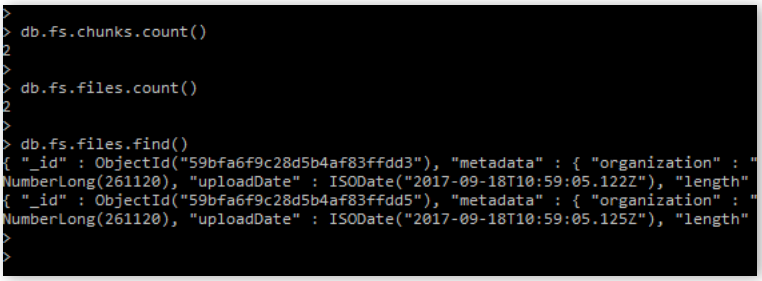
– Make a retrieving textfiles request: `http://localhost:8080/api/retrieve/textfiles`



› JSA › retrieve

Name	Date modified
jsa-logo.png	18/09/2017 5:24 PM
text-1.txt	18/09/2017 5:25 PM
text-2.txt	18/09/2017 5:25 PM

– Make a delete request: `http://localhost:8080/api/delete/image`



```
>
> db.fs.chunks.count()
2
>
> db.fs.files.count()
2
>
> db.fs.files.find()
{ "_id" : ObjectId("59bfa6f9c28d5b4af83ffdd3"), "metadata" : { "organization" : "
NumberLong(261120), "uploadDate" : ISODate("2017-09-18T10:59:05.122Z"), "length"
{ "_id" : ObjectId("59bfa6f9c28d5b4af83ffdd5"), "metadata" : { "organization" : "
NumberLong(261120), "uploadDate" : ISODate("2017-09-18T10:59:05.125Z"), "length"
>
>
```

IV. Sourcecode

SpringDataGridFsTemplate

By grokonez | September 18, 2017.

Related Posts

- Amazon S3 – Upload/Download files with SpringBoot Amazon S3 application.
- NodeJS – GridFS APIs Upload/Download Files to MongoDB – Mongoose
- Bootstrap Image + MongoDB GridFsTemplate + SpringBoot RestAPI
- Spring Boot + Vue.js example | Spring Data MongoDB + RestApi CRUD
- NodeJS – use Mongoose to save Files/Images to MongoDB
- Spring Boot + React Redux + MongoDb CRUD example
- Kotlin – SpringBoot MongoDB – Model One-to-One, One-to-Many Relationships Embedded Documents
- Kotlin – SpringData MongoRepository to interact with MongoDB
- Angular 4 + Spring Boot + MongoDB CRUD example
- MongoDB – Model One-to-One, One-to-Many Relationships Embedded Documents | SpringBoot

Post Tags

file system | mongo gridfs | MongoDB | spring boot | spring data

4 thoughts on “SpringData MongoDB GridFsTemplate to save, retrieve, delete binary files (Image, Text files)”

David

May 6, 2018 at 3:24 pm

Hi.
I'm trying to use your code.

saveFiles works ok and create documents in mongo, but retrieveImageFile give me an error in the next line:

GridFSDBFile imageFile = gridOperations.findOne(new Query(Criteria.where("_id").is(imageFileId)));

Type mismatch: cannot convert from GridFSFile to GridFSDBFile

Tried to change GridFSDBFile to GridFSFile but then imageFile.writeTo doesn't work.

Any idea?
Thanks

Charlie

June 6, 2018 at 4:45 pm

I have the same issue as david.

JavaSampleApproach

June 7, 2018 at 2:09 am

May be you are working with Spring Boot 2.0.x that GridFS operations are mowing towards Resource.
You can try with a solution below:

```
gridFsTemplate.getResource("your-file-name.txt")
```

Regards,
JSA

Bharath.k

December 17, 2018 at 10:22 am

When i try to do read or write i am getting com.mongodb.MongoQueryException: Query failed with error code 13 and error message 'there are no users authenticated' on server

grokonez

Home | Privacy Policy | Contact Us | Our Team

FOLLOW US

ABOUT US

We are passionate engineers in software development by Java Technology & Spring Framework. We believe that creating little good thing with specific orientation everyday can make great influence on the world someday.