Example: MongoDB Ruby Driver with GridFS

Load Example Script/Class

1. Load Example Script with GridfsLoader class

   ```
   $ irb
   > require './gridfs_loader'
    => true
   ```

2. Get Connection to MongoDB

   ```
   > GridfsLoader.mongo_client
   creating connection mongodb://localhost:27017 test
    => #<Mongo::Client:0x16705380 cluster=localhost:27017>
   ```

Import a File (into GridFS)

1. Locate Example File to Import

   ```
    $ ls -ltrh
    total 7.5M
   -rw-rw-r--. 1 jim jim 313K Nov  9 04:11 image1.jpg
   -rw-rw-r--. 1 jim jim 7.2M Nov  9 04:11 image2.jpg
   -rw-rw-r--. 1 jim jim 1.3K Nov 11 01:14 gridfs_loader.rb
   ```

2. Load the Operating System (OS) File

   ```
   > os_file=File.open("./image1.jpg", "rb")
    => #<File:./image1.jpg>
   ```

3. Create a GridFS File from OS File

   Note that we are not passing in any file description properties at this time. All info will be derived form the raw file itself.

   ```
   > grid_file = Mongo::Grid::File.new(os_file.read)
     => #<Mongo::Grid::File:0x16281840 filename=>
   ```

4. Inspect the GridFS File

   ```
   > grid_file.methods
    => [:chunk_size, :content_type, :filename, :id, :md5, :upload_date, :chunks, :data, :info
   ```

   MongoDB assigns an `id` we can use later to get the data from GridFS.

   ```
   > grid_file.id
    => BSON::ObjectId('5642e168e301d09ce9000000')
   ```

   Content type defaults to `binary/octet-stream`

   ```
   > grid_file.content_type
    => "binary/octet-stream"
   ```

   Filename is `nil` because this is not derived from the `os_file` or data and we did not pass in file description properties in when we created the file.

   ```
   > grid_file.filename
    => nil
   ```

MongoDB calculates an md5 hash from the data.

```
> grid_file.md5
 => #<Digest::MD5: 3468ca1c23cc13ac6af493c4642cc72a>
```

MongoDB timestamps the creation of the file at the time the MongoDB File was created. It is not yet technically in MongoDB/GridFS yet.

```
> grid_file.upload_date
 => 2015-11-11 06:32:29 UTC
```

Much of what was shown above enacsulated in the file's info object.

```
> grid_file.info
 => #<Mongo::Grid::File::Info:0x21616940
chunk_size=261120
filename=
content_type=binary/octet-stream
id=5642e168e301d09ce9000000
md5=3468ca1c23cc13ac6af493c4642cc72a>
```

Chunks are broken up into ~255KB

```
> grid_file.chunk_size
 => 261120
```

Our 313K OS File was broken up into two (2) chunks of up to 255K each.

```
> grid_file.chunks.count
 => 2
> grid_file.chunks
 => [#<Mongo::Grid::File::Chunk:0x0000000293a6c8 ... "n"=>0}>,
     #<Mongo::Grid::File::Chunk:0x0000000293a330 ... "n"=>1}>]
```

5. Write the GridFS File into GridFS

```
> c=GridfsLoader.mongo_client
creating connection mongodb://localhost:27017 test
 => #<Mongo::Client:0x20670940 cluster=localhost:27017>
> r=c.database.fs.insert_one(grid_file)
 => BSON::ObjectId('5642e168e301d09ce9000000')
```

Export File (from GridFS)

1. Find the GridFS File (by ID)

```
> stored_file=c.database.fs.find_one(:_id => BSON::ObjectId('5642e168e301d09ce9000000'))
 => #<Mongo::Grid::File:0x19799600 filename=>
```

2. Create an Output File to Write To

```
> os_file2=File.open("./exported_copy.jpg","wb")
 => #<File:./exported_copy.jpg>
```

3. Write data to File

```
> stored_file.chunks.size
 => 2
> stored_file.chunks.reduce([]) { |x,chunk| os_file2 << chunk.data.data }
 => #<File:./exported_copy.jpg>
```

4. Locate New OS File Copy

```
-rw-rw-r--. 1 jim jim 313K Nov  9 04:11 image1.jpg
-rw-rw-r--. 1 jim jim 7.2M Nov  9 04:11 image2.jpg
-rw-rw-r--. 1 jim jim 1.3K Nov 11 01:14 gridfs_loader.rb
-rw-rw-r--. 1 jim jim 313K Nov 11 02:17 exported_copy.jpg
```

Adding File Info/Description

1. Create a description hash to pass into GridFS initialize

```
> description={}
 => {}
```

2. Assign a `filename` (this is a standard property)

```
> description[:filename]="myfile.jpg"
 => "myfile.jpg"
```

3. Assign a `content_type` (this is a standard property)

```
> description[:content_type]="image/jpeg"
 => "image/jpeg"
```

4. Assign some custom properties. This must go in metadata (metadata is a standard property but what is within metadata is custom)

```
> description[:metadata]={:author=>"kiran", :topic=>"nice spot"}
 => {:author=>"kiran", :topic=>"nice spot"}
```

5. Pass the file properties into initialize with OS File with data bytes

```
> grid_file = Mongo::Grid::File.new(os_file.read, description)
 => #<Mongo::Grid::File:0x22644620 filename=myfile.jpg>
```

6. Notice the standard properties showing up in the info object

```
> grid_file.info
 => #<Mongo::Grid::File::Info:0x22644480
      chunk_size=261120
      filename=myfile.jpg
      content_type=image/jpeg
      id=5642f149e301d09ce9000009
      md5=3468ca1c23cc13ac6af493c4642cc72a>
```

7. Write data and file properties into GridFS

```
> r=c.database.fs.insert_one(grid_file)
 => BSON::ObjectId('5642f149e301d09ce9000009')
```

Find Files (Grid::File)

- Query standard properties at document root

```
> c.database.fs.find_one(:contentType=>'image/jpeg', :filename=>'myfile.jpg')
 => #<Mongo::Grid::File:0x20470500 filename=myfile.jpg>
```

- Query custom metadata properties using nested property dot (".") syntax

```
> c.database.fs.find_one(:"metadata.author"=>"kiran", :"metadata.topic"=>{:$regex=>"spot"})
 => #<Mongo::Grid::File:0x18411820 filename=myfile.jpg>
```

Find File Properties (Ruby Hash)

- Query standard properties at document root

```
> require 'pp'
> pp c.database.fs.find(:contentType=>'image/jpeg', :filename => "myfile.jpg").first
{"_id"=>BSON::ObjectId('5642f149e301d09ce9000009'),
 "chunkSize"=>261120,
 "uploadDate"=>2015-11-11 07:41:50 UTC,
 "contentType"=>"image/jpeg",
 "filename"=>"myfile.jpg",
 "metadata"=>{"author"=>"kiran", "topic"=>"nice spot"},
 "length"=>307797,
 "md5"=>"3468ca1c23cc13ac6af493c4642cc72a"}
```

- Query custom metadata properties using nested property dot (".") syntax

```
> pp c.database.fs.find(:"metadata.author"=>"kiran", :"metadata.topic"=>{:$regex=>"spot"}).first
{"_id"=>BSON::ObjectId('5642f149e301d09ce9000009'),
 "chunkSize"=>261120,
 "uploadDate"=>2015-11-11 07:41:50 UTC,
 "contentType"=>"image/jpeg",
 "filename"=>"myfile.jpg",
 "metadata"=>{"author"=>"kiran", "topic"=>"nice spot"},
 "length"=>307797,
 "md5"=>"3468ca1c23cc13ac6af493c4642cc72a"}
```

Deleting Files from GridFS

- Delete a specific File

```
> id=c.database.fs.find(:"metadata.author"=>"kiran").first[:_id]
 => BSON::ObjectId('5642f149e301d09ce9000009')
> r=c.database.fs.find(:_id=>id).delete_one
 => #<Mongo::Operation::Result:22441840 documents=[{"ok"=>1, "n"=>1}]>
> r.deleted_count
 => 1
> r.n
 => 1
```

- Delete all Files

```
> r=c.database.fs.find.delete_many
 => #<Mongo::Operation::Result:18398180 documents=[{"ok"=>1, "n"=>6}]>
```

Last Updated: 2016-02-21