
gridfs-stream

Easily stream files to and from MongoDB GridFS.

Please note

gridfs-stream v1.x uses Stream2 API from nodejs v0.10 (and the mongodb v2.x driver). It provides more robust and easier to use streams. If for some reason you need nodejs v0.8 streams, please switch to the gridfs-stream 0.x branch

Description

```
1 var mongo = require('mongodb');
2 var Grid = require('gridfs-stream');
3
4 // create or use an existing mongodb-native db instance
5 var db = new mongo.Db('yourDatabaseName', new mongo.Server("127.0.0.1",
6   27017));
7 var gfs = Grid(db, mongo);
8
9 // streaming to gridfs
10 var writestream = gfs.createWriteStream({
11   filename: 'my_file.txt'
12 });
13 fs.createReadStream('/some/path').pipe(writestream);
14
15 // streaming from gridfs
16 var readstream = gfs.createReadStream({
17   filename: 'my_file.txt'
18 });
19 //error handling, e.g. file does not exist
20 readstream.on('error', function (err) {
21   console.log('An error occurred!', err);
22   throw err;
23 });
24
25 readstream.pipe(response);
```

Alternatively you could read the file using an `_id`. This is often a better option, since filenames don't have to be unique within the collection. e.g.

```
1 var readstream = gfs.createReadStream({
2   _id: '50e03d29edfdc00d34000001'
3 });
```

Created streams are compatible with other Node streams so piping anywhere is easy.

install

```
1 npm install gridfs-stream
```

use

```
1 var mongo = require('mongodb');
2 var Grid = require('gridfs-stream');
3
4 // create or use an existing mongodb-native db instance.
5 // for this example we'll just create one:
6 var db = new mongo.Db('yourDatabaseName', new mongo.Server("127.0.0.1",
7   27017));
8
9 // make sure the db instance is open before passing into `Grid`
10 db.open(function (err) {
11   if (err) return handleError(err);
12   var gfs = Grid(db, mongo);
13   // all set!
14 })
```

The `gridfs-stream` module exports a constructor that accepts an open mongodb-native db and the mongodb-native driver you are using. *The db must already be opened before calling `createWriteStream` or `createReadStream`.*

Now we're ready to start streaming.

createWriteStream

To stream data to GridFS we call `createWriteStream` passing any options.

```
1 var writestream = gfs.createWriteStream([options]);
2 fs.createReadStream('/some/path').pipe(writestream);
```

Options may contain zero or more of the following options, for more information see GridStore:

```
1 {
2   _id: '50e03d29edfdc00d34000001', // a MongoDB ObjectId
3   filename: 'my_file.txt', // a filename
4   mode: 'w', // default value: w
5 }
```

```
6 //any other options from the GridStore may be passed too, e.g.:
7
8 chunkSize: 1024,
9 content_type: 'plain/text', // For content_type to work properly,
    set "mode"-option to "w" too!
10 root: 'my_collection',
11 metadata: {
12     ...
13 }
14 }
```

Events

The `writeStream` is a fully compliant Stream2 Writable Stream, it emits all the associated events (`drain`, `finish`, `pipe`, `unpipe`, `error`), as well as additional special events (`open`, `close`).

`finish` is emitted after the file has been completely written to GridFS.

`open` is emitted after the GridStore is successfully opened.

`close` is emitted after the GridStore is successfully closed, which means the file is fully written to GridFS, and the file object is passed as the first argument.

```
1 writestream.on('close', function (file) {
2   // do something with `file`
3   console.log(file.filename);
4 });
```

Methods

The `writeStream` has additional methods:

`destroy([err])`: Destroy the `writeStream` as soon as possible: stop writing incoming data, close the `_store`. An `error` event will be emitted, as well as a `close` event. It's up to you to cleanup the GridStore if it's not desired to keep half written files in GridFS (the `close` event returns a GridStore `file` which can be used to delete the file, or mark it failed).

createReadStream

To stream data out of GridFS we call `createReadStream` passing any options, at least an `_id` or `filename`.

```
1 var readstream = gfs.createReadStream(options);
2 readstream.pipe(response);
```

See the options of `createWriteStream` for more information.

To get partial data with `createReadStream`, use `range` option. e.g.

```
1 var readstream = gfs.createReadStream({
2   _id: '50e03d29edfdc00d34000001',
3   range: {
4     startPos: 100,
5     endPos: 500000
6   }
7 });
```

removing files

Files can be removed by passing options (at least an `_id` or `filename`) to the `remove()` method.

```
1 gfs.remove(options, function (err, gridStore) {
2   if (err) return handleError(err);
3   console.log('success');
4 });
```

See the options of `createWriteStream` for more information.

check if file exists

Check if a file exist by passing options (at least an `_id` or `filename`) to the `exist()` method.

```
1 gfs.exist(options, function (err, found) {
2   if (err) return handleError(err);
3   found ? console.log('File exists') : console.log('File does not exist');
4 });
```

See the options of `createWriteStream` for more information.

accessing file metadata

All file meta-data (file name, upload date, contentType, etc) are stored in a special mongodb collection separate from the actual file data. This collection can be queried directly:

```
1 var gfs = Grid(conn.db);
2 gfs.files.find({ filename: 'myImage.png' }).toArray(function (err,
3   files) {
4   if (err) ...
5   console.log(files);
```

```
5    })
```

Alternatively you can use the `gfs.findOne`-shorthand to find a single file

```
1    gfs.findOne({ _id: '54da7b013706c1e7ab25f9fa'}, function (err, file)
2        {
3            console.log(file);
4        });
```

using with mongoose

```
1    var mongoose = require('mongoose');
2    var Grid = require('gridfs-stream');
3
4    var conn = mongoose.createConnection(..);
5    conn.once('open', function () {
6        var gfs = Grid(conn.db, mongoose.mongo);
7
8        // all set!
9    })
```

You may optionally assign the driver directly to the `gridfs-stream` module so you don't need to pass it along each time you construct a grid:

```
1    var mongoose = require('mongoose');
2    var Grid = require('gridfs-stream');
3    Grid.mongo = mongoose.mongo;
4
5    var conn = mongoose.createConnection(..);
6    conn.once('open', function () {
7        var gfs = Grid(conn.db);
8
9        // all set!
10   })
```

LICENSE