Working With Images And MongoDB

Presented to
Washington, DC MongoDB Users Group
by Bob Cochran

email: r2cochran2@gmail.com
Wednesday, August 20, 2014

What We Will Discuss

- Adding images to a MongoDB 2.6 collection using a Node.js and Express application.
- Retrieving images from MongoDB collections and displaying them on web pages.

Required Software

- Minimum MongoDB version of 2.4.x. This presentation was tested on MongoDB 2.6.3.
- Node.js v0.10.29 or higher, compiled from source or installed from (downloaded) binary releases from nodejs.org. I compile Node from source code on both the Mac OS X and Linux platforms.
- Express.js version 4.

Optional Express Middleware

- Node-multiparty or other middleware for extracting data from multipart web forms.

The Goal

- Add images to MongoDB collections
- Query for images from MongoDB collections
- We are not using GridFS. Instead, we are storing JPEG images directly into collections as binary fields.

Programming Language

- Node.js is an implementation of asynchronous JavaScript. Tonight's discussion centers on Node and the supported 'mongodb' Node driver.
- I found Node very difficult to develop with for this project.
- Node learning curve is difficult.

Adding Images to MongoDB

- Read the image to be added to the MongoDB collection. Often, you want to read a number of images and process them in one task.
- Two ways to read a file in Node. Read asynchronously, with Node's “fs.createReadStream()” method, or read synchronously with the “fs.readFileSync()” method.

Adding Images to MongoDB

- Asynchronous reads are non-blocking: Node.js has a thread running separately from your code which performs the read operation. When the operation either fails or completes, a callback event is fired. Your code processes the file reading results, including the data, in the callback.

Adding Images To MongoDB

Anatomy of an asynchronous file read with ReadStream

- call fs.createReadStream() for the file to be read.
- process the 'open', 'data', and 'end' events emitted by readStream.
- readStream opens, reads, streams, closes the file.

Adding Images To MongoDB

Check the size of the image file to ensure it does not exceed 16 megabytes. MongoDB documents are limited to 16 Mb in size. File size information is available through fs.fstat.

Adding Images To MongoDB

Anatomy of a synchronous file read

Call fs.readFileSync() with the name of the file to be read as a parameter.

Adding Images To MongoDB

Synchronous reads are “blocking”: all other operations will wait until the file is either read successfully, or the read fails. There is a noticeable delay while the file is read. If the encoding option is specified, you get a string, otherwise you get a buffer.

Adding Images To MongoDB

- Depending on how you elect to read the image file, you have either a buffer or a string representation of the image.
- The image has to be converted to a binary JSON (called BSON) object.
- This is done with the Binary method of the mongodb Node driver.

Using the mongodb.Binary driver

```
var MongoBinData =
require('mongodb').Binary
var image = new
MongoBinData(data)
```

If *data* is the name of the buffer containing your photo, then *image* is the binary JSON object representation of that buffer.

Insert Photo To MongoDB

Add the new document containing the BSON-ified image to your collection, using collection.save().

```
db.collection(“articles”, function (error, collection) {
        collection.save({
                        “img” : image,
                    “author” : iauthor}, { w:1 }, callback);
})
```

Image Saved To Collection

- collection.save() either inserts the image as a part of a new document, or updates the existing document contents if the action is for a matching underscore id key field (“_id”).
- In the code shown here, the method simply inserts new documents. The _id key is generated for us automatically.

ReadStreams Are Trouble

- Node's ReadStreams do not seem to work for me.
- When an image is later queried, it does not display on a web page. The image is corrupted.
- Possible causes: incorrectly chained callbacks; insufficient time allowed for callbacks to fire; or the “end” event is emitted before the complete contents of the image file are read in and added to the buffer.
- Complete image may not be stored in the collection.

Synchronous Reads Work

fs.readFileSync returns a buffer with the contents of the image file.

When queries are later done, image contents display correctly when added to a web page.

Synchronous reads are “blocking”, meaning other operations halt until the file is read. Processing numerous images will be slow.

Two MongoDB Databases

Database: 'images'

Collection: 'demoimages'

- Documents contain images added using Node's fs.createReadStream() method

Database: 'roberts'

Collection: 'articles'

- Documents contain images added using Node's fs.readFileSync() method

Collection Fields

demoimages

- _id
- fn
- image

articles

- _id
- idt
- author
- content
- im_typ
- im_len
- im_name
- img

Query For The Images

Now that we have images in our MongoDB collections, we can query for them and do things with them – for example, display them on a web page. Consider an http request captured by Express.js middleware which triggers a query to the database, followed by an http response containing image data.

Act On The HTTP Query String

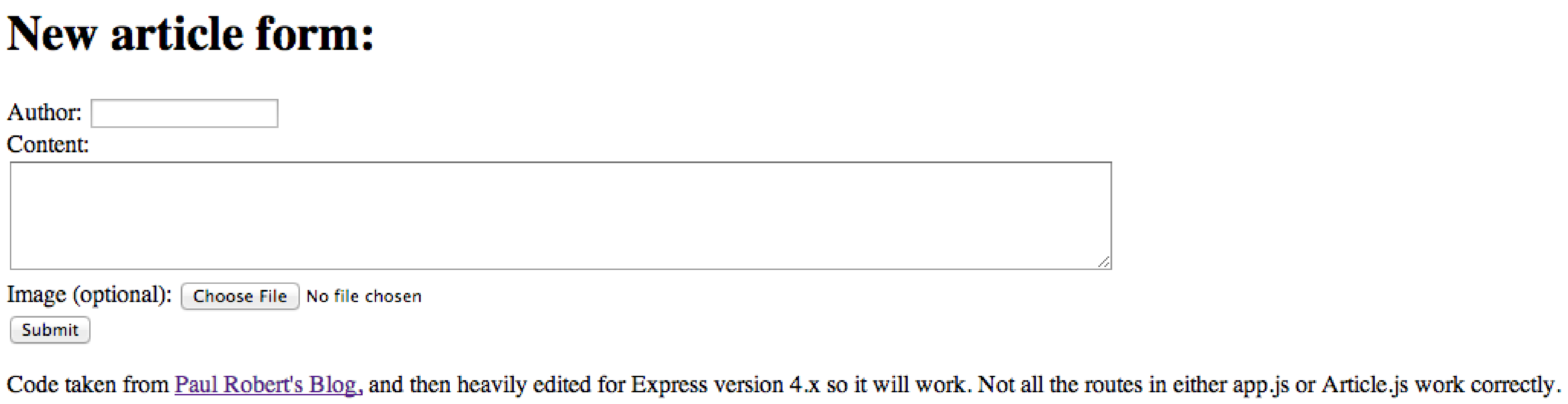
Express.js is middleware code that helps process incoming http requests. A request might be for differing web page 'views' which depend on the URL string the user types into the browser. Express processes the http request, and sends back an http response of some sort.

Save Images Using A Form

Suppose the user types:

http://localhost:3000/

In a browser address bar. This is saying: “send a request to the web server that is listening on port 3000 of this machine. I would like to upload an image to MongoDB using a web form.”



New article form:

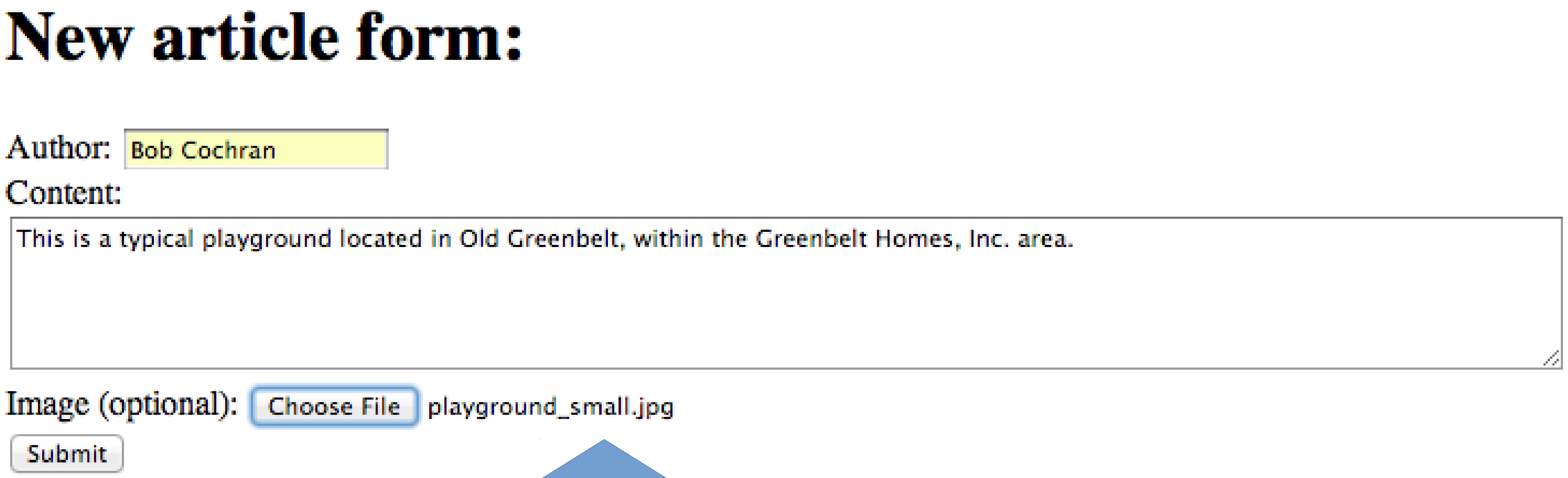
Author:

Content:

Image (optional):

Code taken from Paul Robert's Blog, and then heavily edited for Express version 4.x so it will work. Not all the routes in either app.js or Article.js work correctly.

We Can Add An Image!



New article form:

Author: Bob Cochran

Content:

This is a typical playground located in Old Greenbelt, within the Greenbelt Homes, Inc. area.

Image (optional): Choose File playground_small.jpg

Submit

Acknowledgements

I am so grateful to the following people:

Julie Paparelli and Gerard Williams, sign language interpreters.

Rachel Channon and Robert Weinstock for reviewing these slides.

Zeki Mokhtarzada, founder and CTO of Webs, for sponsoring sign language interpreters.

Caleb Harris for accepting my offer to speak.