Constructing socnetapp in MEAN

with the book Write Modern Web Apps with the MEAN stack

Things start to get going in Chapter 4, with the angular posts.html having already been run and analyzed. So I copied it to the directory, and created package.json with the name socialapp, as per instructions right at the start of Chapter 4. It's important to use the “npm install –save {name}” syntax to make sure the right versions are installed according to which version of node is installed (or current via the “n” app).

The Social Networking Project: from static mockup to Angular

See posts.html

Building a Node.js API

Commit 4.01 basic provisioning with package.json

So I created a file “package.json” just for express and body-parser which I placed in the root directory. I also created a .gitignore, and committed the two files to a local git repo.

```
$ npm install --save express
$ npm install --save body-parser
$ node –version
v0.12.1
$ cat package.json
{
  "name": "socialapp",
  "dependencies": {
    "body-parser": "^1.12.2",
    "express": "^4.12.3"
  }
}
$ git commit -am "Initial commit"
[master (root-commit) db7185b] Initial commit
 3 files changed, 61 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 package.json
 create mode 100644 posts.html
```

Commit 4.02 bare bones server

Added bare bones starter server.js and ran:

```
$ node server.js
Server listening on 3000
```

Pointing my web browser at http://localhost:3000/api/posts produced in the browser:

```
[{"username":"dickeyxxx","body":"node rocks!"}]
```

I commited server.js:

```
$ git commit -am "bare bones server with express and body parser"
[master ba898d7] bare bones server with express and body parser
 1 file changed, 18 insertions(+)
 create mode 100644 server.js
```

Commit 4.03 including this doc

```
$ git commit -am "Added documentation with Libre Office doc"
[master 6bd8971] Added documentation with Libre Office doc
 3 files changed, 2 insertions(+)
 create mode 100644 doc/.~lock.constructing-socnetapp-in-MEAN.odt#
 create mode 100644 doc/constructing-socnetapp-in-MEAN.odt
```

Commit 4.04 creating POST endpoint and testing with curl

We add the app.post POST request and test it with curl. The book uses the following curl statement:

```
curl -v -H "Content-Type: application/json" -XPOST --data
"{\"username\":\"dickeyxxx\", \"body\":\"node rules!\"}" localhost:3000/api/posts
```

This works but requires the awkward, mistakes prone and hard to read escaped quotes \" .

I used the following:

```
$ curl -v -H "Content-Type: application/json" -XPOST --data
'{"username":"dickeyxxx", "body":"node rules!"}' localhost:3000/api/posts
```

which works fine.

The -v switch means verbose, the -H means header (advising that json is being sent), the -X indicates the type of request command, and --data heralds the data being included in the POST request.

The terminal output from the curl command is:

```
* Hostname was NOT found in DNS cache
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 3000 (#0)
> POST /api/posts HTTP/1.1
> User-Agent: curl/7.37.1
> Host: localhost:3000
> Accept: */*
> Content-Type: application/json
> Content-Length: 46
>
* upload completely sent off: 46 out of 46 bytes
< HTTP/1.1 201 Created
< X-Powered-By: Express
< Content-Type: text/plain; charset=utf-8
< Content-Length: 7
< ETag: W/"7-7d808e24"
< Date: Tue, 31 Mar 2015 16:37:43 GMT
< Connection: keep-alive
<
* Connection #0 to host localhost left intact
```

and the running server.js console log says:

```
post received!
dickeyxxx
node rules!
$ node server.js
Server listening on 3000
```

```
post received!
dickeyxxx
node rules!
express deprecated res.send(status): Use res.sendStatus(status) instead
server.js:20:7
```

We commit:

```
$ git commit -am "Created POST endpoint and tested with curl"
[master 0cf08d4] Created POST endpoint and tested with curl
 3 files changed, 7 insertions(+), 1 deletion(-)
 delete mode 100644 doc/.~lock.constructing-socnetapp-in-MEAN.odt#
 rewrite doc/constructing-socnetapp-in-MEAN.odt (94%)
```

Commit 4.04bis

Express gave a warning when it returned the successfully POST'd data:

```
express deprecated res.send(status): Use res.sendStatus(status) instead
server.js:20:7
```

So we make that change:

and commit:

```
$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#     modified:   server.js
#
no changes added to commit (use "git add" and/or "git commit -a")
Victors-MacBook-Air:socnetapp victorkane$ git diff
diff --git a/server.js b/server.js
index 390a03a..cda40a2 100644
--- a/server.js
+++ b/server.js
@@ -17,7 +17,7 @@ app.post('/api/posts', function (req, res) {
   console.log('post received!')
   console.log(req.body.username)
   console.log(req.body.body)
-  res.send(201)
+  res.sendStatus(201)
 })

 app.listen(3000, function () {
Victors-MacBook-Air:socnetapp victorkane$ git commit -am "replaced deprecated
express method"
[master 4a1a6eb] replaced deprecated express method
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Commit 4.05 MongoDB Models with Mongoose

At this point we must have mongodb installed. On my MacBook Air, that's done with homebrew. Perhaps you are running a VM or VSP and install using your operating system package manager.

First we make sure mongodb is running, maybe in a terminal we will leave open:

```
$ mongod
all output going to: /usr/local/var/log/mongodb/mongo.log
```

We leave that and in another terminal we execute mongo the MongoDB shell, mongo, which we execute and run through its paces just a bit:

```
$ mongo
MongoDB shell version: 2.2.2
connecting to: test
> show dbs
csbook-todos      0.203125GB
library_database  0.203125GB
lit2012     0.203125GB
local (empty)
nodebackbone      0.203125GB
> use csbook-todos
switched to db csbook-todos
> show collections
system.indexes
todos
> db.todos.find()
{ "title" : "Learn coffee script", "_id" : ObjectId("5107e05b3d90e3ef11000002"),
"created_at" : ISODate("2013-01-29T14:44:43.812Z"), "state" : "pending" }
{ "title" : "one", "_id" : ObjectId("5107e07ed8c140f511000001"), "created_at" :
ISODate("2013-01-29T14:45:18.129Z"), "state" : "pending" }
{ "title" : "this", "_id" : ObjectId("5107e13851aefb0e12000001"), "created_at" :
ISODate("2013-01-29T14:48:24.008Z"), "state" : "pending" }
{ "_id" : ObjectId("5107e0b9d378ecfe11000001"), "created_at" : ISODate("2013-01-
29T14:46:17.072Z"), "state" : "pending", "title" : "oneone" }
{ "title" : "yeah", "_id" : ObjectId("5107e23e6bd7452912000002"), "created_at" :
ISODate("2013-01-29T14:52:46.333Z"), "state" : "pending" }
{ "title" : "seven", "_id" : ObjectId("5107e4b705a5d6c017000001"), "created_at" :
ISODate("2013-01-29T15:03:19.201Z"), "state" : "pending" }
```

Now that we know it is running, we can connect to MongoDB from our app using the mongoose ODM (Object Document Mapper, “elegant mongodb object modeling for node.js”).

First we add mongoose to the project:

```
$ npm install --save mongoose
```

Commit 4.06 Make posts to mongoose

Next we need to add the database connection logic in a separate module db.js, to be accessed by a mongoose model module models/post.js to store the posts.

db.js:

```
var mongoose = require('mongoose')
mongoose.connect('mongodb://localhost/social', function () {
  console.log('mongodb connected')
})
module.exports = mongoose
```

models/post.js:

```
var db = require('../db')
var Post = db.model('Post', {
  username: { type: String, required: true },
  body:     { type: String, required: true },
  date:     { type: Date, required: true, default: Date.now }
```

```
})
module.exports = Post
```
The model Post can now be invoked inside our endpoint to create posts, by modifying the app.post method in server.js as follows:

```
var Post = require('./models/post')
app.post('/api/posts', function (req, res, next) {
  var post = new Post({
    username: req.body.username,
    body: req.body.body
  })
  post.save(function (err, post) {
    if (err) { return next(err) }
    res.json(201, post)
  })
})
```
We stop and execute the server:

```
$ node server.js
Server listening on 3000
mongodb connected
```
The endpoint can now be invoked using curl to create posts:

```
$ curl -v -H "Content-Type: application/json" -XPOST --data
'{"username":"dickeyxxx", "body":"node rules!"}' localhost:3000/api/posts
* Hostname was NOT found in DNS cache
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 3000 (#0)
> POST /api/posts HTTP/1.1
> User-Agent: curl/7.37.1
> Host: localhost:3000
> Accept: */*
> Content-Type: application/json
> Content-Length: 46
>
* upload completely sent off: 46 out of 46 bytes
< HTTP/1.1 201 Created
< X-Powered-By: Express
< Content-Type: application/json; charset=utf-8
< Content-Length: 120
< ETag: W/"78-b6a8cd6e"
< Date: Tue, 31 Mar 2015 22:07:58 GMT
< Connection: keep-alive
<
* Connection #0 to host localhost left intact
{"__v":0,"username":"dickeyxxx","body":"node
rules!","_id":"551b1abecd56d2fa0a955c6e","date":"2015-03-31T22:07:58.062Z"}
```
We can test our success using the mongo shell:

```
$ mongo social
MongoDB shell version: 2.2.2
connecting to: social
> db.posts.find()
{ "username" : "dickeyxxx", "body" : "node rules!", "_id" :
ObjectId("551b1abecd56d2fa0a955c6e"), "date" : ISODate("2015-03-31T22:07:58.062Z"),
"__v" : 0 }
```
Commit:

```
$ git commit -am "Added mongoose connection module, model module for posts,
modified app post endpoint to create posts and tested with mongo, documented"
[master f7a22c0] Added mongoose connection module, model module for posts, modified
app post endpoint to create posts and tested with mongo, documented
 4 files changed, 22 insertions(+), 5 deletions(-)
 create mode 100644 db.js
 rewrite doc/constructing-socnetapp-in-MEAN.odt (97%)
 create mode 100644 models/post.js
```

Commit 4.06bis Get posts from mongoose

We modify the app get endpoint to return all posts stored in mongodb:

```
app.get('/api/posts', function (req, res, next) {
  Post.find(function(err, posts) {
    if (err) { return next(err) }
    res.json(posts)
  })
})
```

To test this after resetting the server we point our browsers at http://localhost:3000/api/posts and see:

```
[{"username":"dickeyxxx","body":"node
rules!","_id":"551b1abecd56d2fa0a955c6e","__v":0,"date":"2015-03-
31T22:07:58.062Z"}]
```

And commit:

```
$ git commit -am "modified the app get endpoint to return all posts stored in
mongodb"
[master b3e0bb3] modified the app get endpoint to return all posts stored in
mongodb
 2 files changed, 5 insertions(+), 7 deletions(-)
 rewrite doc/constructing-socnetapp-in-MEAN.odt (88%)
```

Integrating Node with Angular

Twin objectives:

- Consume API with Angular
- Have node web server serve up the Angular app.listen

Commit 5.01 promises promises promises

The app.controller method in posts.html file was left as follows the last time we worked on it:

```
var app = angular.module('app', [])
app.controller('PostsCtrl', function ($scope) {
  $scope.addPost = function () {
    if ($scope.postBody) {
      $scope.posts.unshift({
        username: 'dickeyxxx',
        body: $scope.postBody
      })
    }
  }
  $scope.posts = [
    {
      username: 'dickeyxxx',
```

```
        body: 'Node rules!'
      },
      {
        username: 'jeffdickey',
        body: 'been trying out angular.js...'
      }
    ]
  })
```

In the first line we dependency inject $httm into the app controller.

Then we replace the stub data in $scope.posts with dynamic data consumed from the node API read in turn from the data persisted in the mongodb database:

```
app.controller('PostsCtrl', function ($scope, $http) {
  $scope.addPost = function () {
    if ($scope.postBody) {
      $scope.posts.unshift({
        username: 'dickeyxxx',
        body: $scope.postBody
      })
    }
  }
  $http.get('http://localhost:3000/api/posts')
  .success(function (posts) {
    $scope.posts = posts
  })
})
```

However, when we point our browsers to:

```
file:///Users/yourname/path_to/posts.html
```

We get a CORS (cross-origin resource sharing) error which you can see in your firebug or chrome console:

```
! Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote
resource at http://localhost:3000/api/posts. This can be fixed by moving the
resource to the same domain or enabling CORS.
```

“CORS is a security policy that browsers follow, instructing them to request data only from the same hostname the HTML is served from. There are ways to prevent this, but the best way for a MEAN app to do this is to serve posts.html from the API server.”

Commit 5.02 serve html from node

We add routing for '/' in server.js:

```
app.get('/', function (req, res) {
 res.sendfile('layouts/posts.html')
})
```

and after moving posts.html to a subdirectory layouts and resetting the server,

```
$ mkdir layouts
$ mv posts.html layouts
```

point our browsers at:

http://localhost:3000/

and it woiks!:

<< image 5.02a.png >>

```
$ git commit -am "serving posts.html through node"
[master 52cdcc3] serving posts.html through node
 3 files changed, 4 insertions(+), 57 deletions(-)
 rewrite doc/constructing-socnetapp-in-MEAN.odt (97%)
 delete mode 100644 posts.html
```