



# RESTful Web API Design with Node.js

---

Saleh Hamadeh

Video 2.8

*Saving Data in MongoDB*



## In this Video, we are going to take a look at...

- Turning our program into a web application:
  - Routes
  - Views
  - Static files
- Accessing MongoDB from Node.js:
  - Saving data
  - Loading data

## FOLDERS

### ▼ twitter-notes

▶ public

▶ views

authenticator.js

config.json

index.js

package.json

storage.js

1

4

```
1 var MongoClient = require('mongodb').MongoClient;
2 var database;
3
4 module.exports = {
5   connect: function() {
6     MongoClient.connect('mongodb://localhost:27017/twitter_notes', function(err,
7       if (err) {
8         return console.log("Error: " + err);
9       }
10
11       db.open(function(err, db) {
12         database = db;
13         console.log("Connected to database.");
14       });
15     });
16   },
17   connected: function() {
18     return typeof database !== 'undefined';
19   },
20   getFriends: function(userId, cb) {
21     var cursor = database.collection('friends').find({
22       for_user: userId
23     });
24
25     cursor.toArray(cb);
26   },
27   insertFriends: function(friends) {
28     database.collection('friends').insert(friends, function(err) {
```



```
1 var MongoClient = require('mongodb').MongoClient;
2 var database;
3
4 module.exports = {
5   connect: function() {
6     MongoClient.connect('mongodb://localhost:27017/twitter_notes', function(err,
7       if (err) {
8         return console.log("Error: " + err);
9       }
10
11       db.open(function(err, db) {
12         database = db;
13         console.log("Connected to database.");
14       });
15     });
16   },
17   connected: function() {
18     return typeof database !== 'undefined';
19   },
20   getFriends: function(userId, cb) {
21     var cursor = database.collection('friends').find({
22       for_user: userId
23     });
24
25     cursor.toArray(cb);
26   },
27   insertFriends: function(friends) {
28     database.collection('friends').insert(friends, function(err) {
```

```
1 <!DOCTYPE html>
2 <html>
3
4   <head>
5     <link rel="stylesheet" href="login.css">
6   </head>
7
8   <body>
9     <div>
10      <h1>Twitter Notes</h1>
11      <h2>Your personal friends diary</h2>
12      <a href="/auth/twitter"></a>
13    </div>
14  </body>
15
16 </html>
```



```
1 <!DOCTYPE html>
2 <html>
3
4 <head>
5   <link rel="stylesheet" href="style.css">
6 </head>
7
8 <body>
9   <div id="wrapper">
10     <div id="friends-column" class="column">
11       <h1>Your Friends</h1>
12       <ul id="friends">
13         <% friends.forEach(function(friend) { %>
14           <li uid="<%= friend.twitter_id %>" class="friend">
15             
16             <p>
17               <span class="name"><%= friend.name %></span>
18               <span class="screen-name">@<%= friend.screen_name %></span>
19             </p>
20             <p class="location"><%= friend.location %></p>
21           </li>
22         <% }); %>
23       </ul>
24     </div><!--
25     --><div id="notes-column" class="column">
26       <h1>Notes</h1>
27       <ul id="notes"></ul>
28     </div>
```



```
1 var url = require('url');
2 var express = require('express');
3 var querystring = require('querystring');
4 var async = require('async');
5 var authenticator = require('./authenticator');
6 var storage = require('./storage.js');
7 var config = require('./config');
8 var app = express();
9
10 // Connect to MongoDB
11 storage.connect();
12
13 // Set the view engine to ejs
14 app.set('view engine', 'ejs');
15
16 // Add cookie parsing functionality to our Express app
17 app.use(require('cookie-parser')());
18
19 // Take user to Twitter's login page
20 app.get('/auth/twitter', authenticator.redirectToTwitterLoginPage);
21
22 // This is the callback url that the user is redirected to after signing in
23 app.get(url.parse(config.oauth_callback).path, function(req, res) {
24     authenticator.authenticate(req, res, function(err) {
25         if (err) {
26             res.redirect('/login');
27         } else {
28             res.redirect('/');
```



```
16 // Add cookie parsing functionality to our Express app
17 app.use(require('cookie-parser')());
18
19 // Take user to Twitter's login page
20 app.get('/auth/twitter', authenticator.redirectToTwitterLoginPage);
21
22 // This is the callback url that the user is redirected to after signing in
23 app.get(url.parse(config.oauth_callback).path, function(req, res) {
24   authenticator.authenticate(req, res, function(err) {
25     if (err) {
26       res.redirect('/login');
27     } else {
28       res.redirect('/');
29     }
30   });
31 });
32
33 // Main page handler
34 app.get('/', function(req, res) {
35   if (!req.cookies.access_token || !req.cookies.access_token_secret || !req.cookies) {
36     return res.redirect('/login');
37   }
38
39   // If the app couldn't connect to the database, get data from Twitter's API
40   if (!storage.connected()) {
41     return renderMainPageFromTwitter(req, res);
42   }
43 }
```



```
47 }
48
49 if (friends.length > 0) {
50   console.log("Data loaded from MongoDB");
51
52   // Sort the friends alphabetically by name
53   friends.sort(function(a, b) {
54     return a.name.toLowerCase().localeCompare(b.name.toLowerCase());
55   });
56
57   // Render the main application
58   res.render('index', {
59     friends: friends
60   });
61 } else {
62   renderMainPageFromTwitter(req, res);
63 }
64 });
65 });
66
67 function renderMainPageFromTwitter(req, res) {
68   async.waterfall([
69     // Get friends' IDs
70     function(cb) {
71       var cursor = -1;
72       var ids = [];
73
```



```
138         location: friend.location,
139         profile_image_url: friend.profile_image_url
140     };
141 });
142
143 // Render the main application
144 res.render('index', {
145     friends: friends
146 });
147
148 // In the background, save the friends to MongoDB
149 if (storage.connected()) {
150     storage.insertFriends(friends);
151 }
152 });
153 }
154 });
155 }
156
157 // Show the login page
158 app.get('/login', function(req, res) {
159     res.render('login');
160 });
161
162 // Serve static files in public directory
163 app.use(express.static(__dirname + '/public'));
164
165 // Start listening for requests
```



```
var app = express();
9
10 // Connect to MongoDB
11 storage.connect();
12
13 // Set the view engine to ejs
14 app.set('view engine', 'ejs');
15
16 // Add cookie parsing functionality to our Express app
17 app.use(require('cookie-parser')());
18
19 // Take user to Twitter's login page
20 app.get('/auth/twitter', authenticator.redirectToTwitterLoginPage);
21
22 // This is the callback url that the user is redirected to after signing in
23 app.get(url.parse(config.oauth_callback).path, function(req, res) {
24   authenticator.authenticate(req, res, function(err) {
25     if (err) {
26       res.redirect('/login');
27     } else {
28       res.redirect('/');
29     }
30   });
31 });
32
33 // Main page handler
34 app.get('/', function(req, res) {
35   if (!req.cookies.access_token || !req.cookies.access_token_secret || !req.cookies
```



```
Index.js
1 {
2   "name": "twitter-notes",
3   "version": "1.0.0",
4   "description": "An application for leaving notes on Twitter friends and followers",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1"
8   },
9   "author": "Saleh Hamadeh",
10  "license": "ISC",
11  "dependencies": {
12    "async": "^1.5.0",
13    "cookie-parser": "^1.4.0",
14    "ejs": "^2.3.4",
15    "express": "^4.13.3",
16    "mongodb": "^2.1.2",
17    "oauth": "^0.9.14"
18  }
19 }
20
```

```
1 {
2   "name": "twitter-notes",
3   "version": "1.0.0",
4   "description": "An application for leaving notes on Twitter friends and followers",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \\\"Error: no test specified\\\" && exit 1"
8   },
9   "author": "Saleh Hamadeh",
10  "license": "ISC",
11  "dependencies": {
12    "async": "^1.5.0",
13    "cookie-parser": "^1.4.0",
14    "ejs": "^2.3.4",
15    "express": "^4.13.3",
16    "mongodb": "^2.1.2",
17    "oauth": "^0.9.14"
18  }
19 }
20
```



```
1 var url = require('url');
2 var express = require('express');
3 var querystring = require('querystring');
4 var async = require('async');
5 var authenticator = require('./authenticator');
6 var storage = require('./storage.js');
7 var config = require('./config');
8 var app = express();
9
10 // Connect to MongoDB
11 storage.connect();
12
13 // Set the view engine to ejs
14 app.set('view engine', 'ejs');
15
16 // Add cookie parsing functionality to our Express app
17 app.use(require('cookie-parser')());
18
19 // Take user to Twitter's login page
20 app.get('/auth/twitter', authenticator.redirectToTwitterLoginPage);
21
22 // This is the callback url that the user is redirected to after signing in
23 app.get(url.parse(config.oauth_callback).path, function(req, res) {
24     authenticator.authenticate(req, res, function(err) {
25         if (err) {
26             res.redirect('/login');
27         } else {
28             res.redirect('/');
```



```
41     return renderMainPageFromTwitter(req, res);
42 }
43
44 storage.getFriends(req.cookies.twitter_id, function(err, friends) {
45     if (err) {
46         return res.status(500).send(err);
47     }
48
49     if (friends.length > 0) {
50         console.log("Data loaded from MongoDB");
51
52         // Sort the friends alphabetically by name
53         friends.sort(function(a, b) {
54             return a.name.toLowerCase().localeCompare(b.name.toLowerCase());
55         });
56
57         // Render the main application
58         res.render('index', {
59             friends: friends
60         });
61     } else {
62         renderMainPageFromTwitter(req, res);
63     }
64 });
65 });
66
67 function renderMainPageFromTwitter(req, res) {
```



```
132 // Transform friends into the format that our application needs
133 friends = friends.map(function(friend) {
134     return {
135         twitter_id: friend.id_str,
136         for_user: req.cookies.twitter_id,
137         name: friend.name,
138         location: friend.location,
139         profile_image_url: friend.profile_image_url
140     };
141 });
142
143 // Render the main application
144 res.render('index', {
145     friends: friends
146 });
147
148 // In the background, save the friends to MongoDB
149 if (storage.connected()) {
150     storage.insertFriends(friends);
151 }
152 });
153 }
154 });
155 }
156
157 // Show the login page
158 app.get('/login', function(req, res) {
159     res.render('login');
```



```
141 //;
142
143 // Render the main application
144 res.render('index', {
145   friends: friends
146 });
147
148 // In the background, save the friends to MongoDB
149 if (storage.connected()) {
150   storage.insertFriends(friends);
151 }
152 });
153 }
154 });
155 }
156
157 // Show the login page
158 app.get('/login', function(req, res) {
159   res.render('login');
160 });
161
162 // Serve static files in public directory
163 app.use(express.static(__dirname + '/public'));
164
165 // Start listening for requests
166 app.listen(config.port, function() {
167   console.log("Listening on port " + config.port);
168 });
```



```
Index.js
141 // ...
142 // ...
143 // Render the main application
144 res.render('index', {
145   friends: friends
146 });
147
148 // In the background, save the friends to MongoDB
149 if (storage.connected()) {
150   storage.insertFriends(friends);
151 }
152 });
153 }
154 });
155 }
156
157 // Show the login page
158 app.get('/login', function(req, res) {
159   res.render('login');
160 });
161
162 // Serve static files in public directory
163 app.use(express.static(__dirname + '/public'));
164
165 // Start listening for requests
166 app.listen(config.port, function() {
167   console.log("Listening on port " + config.port);
168 });
```

```
lawn-143-215-115-165:Desktop saleh$ mongod --dbpath database/db/
```



```

lawn-143-215-115-165:Desktop saleh$ mongod --dbpath database/db/
2016-04-13T11:13:50.374-0400 I JOURNAL [initandlisten] journal dir=database/db/journal
2016-04-13T11:13:50.375-0400 I JOURNAL [initandlisten] recover : no journal files present, no recovery n
eeded
2016-04-13T11:13:50.455-0400 I JOURNAL [durability] Durability thread started
2016-04-13T11:13:50.455-0400 I JOURNAL [journal writer] Journal writer thread started
2016-04-13T11:13:50.456-0400 I CONTROL [initandlisten] MongoDB starting : pid=785 port=27017 dbpath=data
base/db/ 64-bit host=lawn-143-215-115-165.lawn.gatech.edu
2016-04-13T11:13:50.456-0400 I CONTROL [initandlisten]
2016-04-13T11:13:50.456-0400 I CONTROL [initandlisten] ** WARNING: soft rlimits too low. Number of files
is 256, should be at least 1000
2016-04-13T11:13:50.456-0400 I CONTROL [initandlisten] db version v3.0.7
2016-04-13T11:13:50.456-0400 I CONTROL [initandlisten] git version: 6ce7cbe8c6b899552dadd907604559806aa2
e9bd
2016-04-13T11:13:50.456-0400 I CONTROL [initandlisten] build info: Darwin mci-osx108-13.build.10gen.cc 1
2.3.0 Darwin Kernel Version 12.3.0: Sun Jan 6 22:37:10 PST 2013; root:xnu-2050.22.13~1/RELEASE_X86_64 x8
6_64 BOOST_LIB_VERSION=1_49
2016-04-13T11:13:50.456-0400 I CONTROL [initandlisten] allocator: system
2016-04-13T11:13:50.456-0400 I CONTROL [initandlisten] options: { storage: { dbPath: "database/db/" } }
2016-04-13T11:13:50.533-0400 I NETWORK [initandlisten] waiting for connections on port 27017

```



```
1 var MongoClient = require('mongodb').MongoClient;
2 var database;
3
4 module.exports = {
5   connect: function() {
6     MongoClient.connect('mongodb://localhost:27017/twitter_notes', function(err,
7       if (err) {
8         return console.log("Error: " + err);
9       }
10
11       db.open(function(err, db) {
12         database = db;
13         console.log("Connected to database.");
14       });
15     });
16   },
17   connected: function() {
18     return typeof database !== 'undefined';
19   },
20   getFriends: function(userId, cb) {
21     var cursor = database.collection('friends').find({
22       for_user: userId
23     });
24
25     cursor.toArray(cb);
26   },
27   insertFriends: function(friends) {
28     database.collection('friends').insert(friends, function(err) {
```



```
1 var url = require('url');
2 var express = require('express');
3 var querystring = require('querystring');
4 var async = require('async');
5 var authenticator = require('./authenticator');
6 var storage = require('./storage.js');
7 var config = require('./config');
8 var app = express();
9
10 // Connect to MongoDB
11 storage.connect();
12
13 // Set the view engine to ejs
14 app.set('view engine', 'ejs');
15
16 // Add cookie parsing functionality to our Express app
17 app.use(require('cookie-parser')());
18
19 // Take user to Twitter's login page
20 app.get('/auth/twitter', authenticator.redirectToTwitterLoginPage);
21
22 // This is the callback url that the user is redirected to after signing in
23 app.get(url.parse(config.oauth_callback).path, function(req, res) {
24     authenticator.authenticate(req, res, function(err) {
25         if (err) {
26             res.redirect('/login');
27         } else {
28             res.redirect('/');
```



```
1 var MongoClient = require('mongodb').MongoClient;
2 var database;
3
4 module.exports = {
5   connect: function() {
6     MongoClient.connect('mongodb://localhost:27017/twitter_notes', function(err,
7       if (err) {
8         return console.log("Error: " + err);
9       }
10
11       db.open(function(err, db) {
12         database = db;
13         console.log("Connected to database.");
14       });
15     });
16   },
17   connected: function() {
18     return typeof database !== 'undefined';
19   },
20   getFriends: function(userId, cb) {
21     var cursor = database.collection('friends').find({
22       for_user: userId
23     });
24
25     cursor.toArray(cb);
26   },
27   insertFriends: function(friends) {
28     database.collection('friends').insert(friends, function(err) {
```



```
1 var MongoClient = require('mongodb').MongoClient;
2 var database;
3
4 module.exports = {
5   connect: function() {
6     MongoClient.connect('mongodb://localhost:27017/twitter_notes', function(err,
7       if (err) {
8         return console.log("Error: " + err);
9       }
10
11     db.open(function(err, db) {
12       database = db;
13       console.log("Connected to database.");
14     });
15   });
16 },
17 connected: function() {
18   return typeof database !== 'undefined';
19 },
20 getFriends: function(userId, cb) {
21   var cursor = database.collection('friends').find({
22     for_user: userId
23   });
24
25   cursor.toArray(cb);
26 },
27 insertFriends: function(friends) {
28   database.collection('friends').insert(friends, function(err) {
```



```
51 console.log('Data loaded from MongoDB');
52
53 // Sort the friends alphabetically by name
54 friends.sort(function(a, b) {
55     return a.name.toLowerCase().localeCompare(b.name.toLowerCase());
56 });
57
58 // Render the main application
59 res.render('index', {
60     friends: friends
61 });
62 } else {
63     renderMainPageFromTwitter(req, res);
64 }
65 });
66
67 function renderMainPageFromTwitter(req, res) {
68     async.waterfall([
69         // Get friends' IDs
70         function(cb) {
71             var cursor = -1;
72             var ids = [];
73
74             // Get IDs by traversing the cursored collection
75             async.whilst(function() {
76                 return cursor !== 0;
77             }, function(cb) {
```



```
Index.js
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
```

storage.js

```
});

// Transform friends into the format that our application needs
friends = friends.map(function(friend) {
  return {
    twitter_id: friend.id_str,
    for_user: req.cookies.twitter_id,
    name: friend.name,
    location: friend.location,
    profile_image_url: friend.profile_image_url
  };
});

// Render the main application
res.render('index', {
  friends: friends
});

// In the background, save the friends to MongoDB
if (storage.connected()) {
  storage.insertFriends(friends);
}

});

});

}

// Show the login page
```

Index.js

storage.js

```
1 /
2
3
4
5
6
7
8     return console.log("Error: " + err);
9 }
10
11 db.open(function(err, db) {
12     database = db;
13     console.log("Connected to database.");
14 });
15 });
16 },
17 connected: function() {
18     return typeof database !== 'undefined';
19 },
20 getFriends: function(userId, cb) {
21     var cursor = database.collection('friends').find({
22         for_user: userId
23     });
24
25     cursor.toArray(cb);
26 },
27 insertFriends: function(friends) {
28     database.collection('friends').insert(friends, function(err) {
29         if (err) {
30             console.log("Cannot insert friends to database.");
31         }
32     });
33 }
34 }
```



```
Index.js      x
storage.js    x

130      });
131
132      // Transform friends into the format that our application needs
133      friends = friends.map(function(friend) {
134          return {
135              twitter_id: friend.id_str,
136              for_user: req.cookies.twitter_id,
137              name: friend.name,
138              location: friend.location,
139              profile_image_url: friend.profile_image_url
140          };
141      });
142
143      // Render the main application
144      res.render('index', {
145          friends: friends
146      });
147
148      // In the background, save the friends to MongoDB
149      if (storage.connected()) {
150          storage.insertFriends(friends);
151      }
152      });
153  }
154  });
155  }
156
157  // Show the login page
```



```
Index.js      x
storage.js    x

30      });
31    });
32
33    // Main page handler
34    app.get('/', function(req, res) {
35      if (!req.cookies.access_token || !req.cookies.access_token_secret || !req.cookies.access_token_expiry) {
36        return res.redirect('/login');
37      }
38
39      // If the app couldn't connect to the database, get data from Twitter's API
40      if (!storage.connected()) {
41        return renderMainPageFromTwitter(req, res);
42      }
43
44      storage.getFriends(req.cookies.twitter_id, function(err, friends) {
45        if (err) {
46          return res.status(500).send(err);
47        }
48
49        if (friends.length > 0) {
50          console.log("Data loaded from MongoDB");
51
52          // Sort the friends alphabetically by name
53          friends.sort(function(a, b) {
54            return a.name.toLowerCase().localeCompare(b.name.toLowerCase());
55          });
56
57          // Render the main application
```



```
Index.js  x  storage.js  x
7
8      return console.log("Error: " + err);
9    }
10
11    db.open(function(err, db) {
12      database = db;
13      console.log("Connected to database.");
14    });
15  });
16},
17connected: function() {
18  return typeof database !== 'undefined';
19},
20getFriends: function(userId, cb) {
21  var cursor = database.collection('friends').find({
22    for_user: userId
23  });
24
25  cursor.toArray(cb);
26},
27insertFriends: function(friends) {
28  database.collection('friends').insert(friends, function(err) {
29    if (err) {
30      console.log("Cannot insert friends to database.");
31    }
32  });
33}
34}
```

```
1 /
2
3
4
5
6
7
8     return console.log("Error: " + err);
9 }
10
11 db.open(function(err, db) {
12     database = db;
13     console.log("Connected to database.");
14 });
15 });
16 },
17 connected: function() {
18     return typeof database !== 'undefined';
19 },
20 getFriends: function(userId, cb) {
21     var cursor = database.collection('friends').find({
22         for_user: userId
23     });
24     cursor.toArray(cb);
25 },
26 insertFriends: function(friends) {
27     database.collection('friends').insert(friends, function(err) {
28         if (err) {
29             console.log("Cannot insert friends to database.");
30         }
31     });
32 });
33 }
34 }
```



```
Index.js  x  storage.js  x
1  /
2
3  (err) {
4      return console.log("Error: " + err);
5  }
6
7
8
9
10
11  db.open(function(err, db) {
12      database = db;
13      console.log("Connected to database.");
14  });
15  });
16  },
17  connected: function() {
18      return typeof database !== 'undefined';
19  },
20  getFriends: function(userId, cb) {
21      var cursor = database.collection('friends').find({
22          for_user: userId
23      });
24
25      cursor.toArray(cb);
26  },
27  insertFriends: function(friends) {
28      database.collection('friends').insert(friends, function(err) {
29          if (err) {
30              console.log("Cannot insert friends to database.");
31          }
32      });
33  }
34  }
```



```
38
39 // If the app couldn't connect to the database, get data from Twitter's API
40 if (!storage.connected()) {
41     return renderMainPageFromTwitter(req, res);
42 }
43
44 storage.getFriends(req.cookies.twitter_id, function(err, friends) {
45     if (err) {
46         return res.status(500).send(err);
47     }
48
49     if (friends.length > 0) {
50         console.log("Data loaded from MongoDB");
51
52         // Sort the friends alphabetically by name
53         friends.sort(function(a, b) {
54             return a.name.toLowerCase().localeCompare(b.name.toLowerCase());
55         });
56
57         // Render the main application
58         res.render('index', {
59             friends: friends
60         });
61     } else {
62         renderMainPageFromTwitter(req, res);
63     }
64 });
65 });
```



```
38
39 // If the app couldn't connect to the database, get data from Twitter's API
40 if (!storage.connected()) {
41     return renderMainPageFromTwitter(req, res);
42 }
43
44 storage.getFriends(req.cookies.twitter_id, function(err, friends) {
45     if (err) {
46         return res.status(500).send(err);
47     }
48
49     if (friends.length > 0) {
50         console.log("Data loaded from MongoDB");
51
52         // Sort the friends alphabetically by name
53         friends.sort(function(a, b) {
54             return a.name.toLowerCase().localeCompare(b.name.toLowerCase());
55         });
56
57         // Render the main application
58         res.render('index', {
59             friends: friends
60         });
61     } else {
62         renderMainPageFromTwitter(req, res);
63     }
64 });
65 });
```



```
38
39 // If the app couldn't connect to the database, get data from Twitter's API
40 if (!storage.connected()) {
41     return renderMainPageFromTwitter(req, res);
42 }
43
44 storage.getFriends(req.cookies.twitter_id, function(err, friends) {
45     if (err) {
46         return res.status(500).send(err);
47     }
48
49     if (friends.length > 0) {
50         console.log("Data loaded from MongoDB");
51
52         // Sort the friends alphabetically by name
53         friends.sort(function(a, b) {
54             return a.name.toLowerCase().localeCompare(b.name.toLowerCase());
55         });
56
57         // Render the main application
58         res.render('index', {
59             friends: friends
60         });
61     } else {
62         renderMainPageFromTwitter(req, res);
63     }
64 });
65 });
```



# Summary

- OAuth 1.0a
  - Parties
  - Process
- Twitter's REST API
  - Tweeting
  - Searching for tweets
  - Finding friends
- Async.js
- MongoDB

Next Section

Building a RESTful API