Deleting Users

This lesson shows how to implement the Delete Users operation. It goes over all four aspects of the implementation including HTML, typescript, UserService, and back-end code.

WE'LL COVER THE FOLLOWING

Delete Users: HTML Code

Just as it was for updating users, the Deleting users feature is intertwined with the table of users.

If you take another look at the table of users you will notice that there is `Delete` button for each row, i.e., each user.

```
<div class="col-sm-2">
    <button class="btn btn-default" (click)="deleteUser(user)">Delete</button>
</div>
```

/mean_frontend/src/app/users/users.component.html

Delete Users: TypeScript Code

Once this button is clicked, the `deleteUser()` function is called.

```
deleteUser(user:User)
  {
    this.userService.deleteUser(user)
    .subscribe(
      data => {
        this.users.splice(this.users.indexOf(user), 1);
        console.log("User deleted!");


      })
  }
```

/mean_frontend/src/app/users/users.component.ts

The information flow is the same as in previous examples: data from *HTML* is passed down to the `deleteUser()` function, the function passes it to the `userService`, and the service, uses an *HTTP* request to transmit it to the back-end.

Delete Users: Service Implementation

In the background this is managed like this:

```
@Injectable()
export class UserService {

    constructor(private http: Http) {
    }

    deleteUser(user:User): Observable<any>{
        return this.http.post("http://localhost:3000/deleteUser", { id: user._id })
        .map((res:any) => {
            return res.json();
        })
        .catch((error:any) => {
            return Observable.throw(error.json ? error.json().error : error || 'Server error'
        });
    }
}
```

/mean_frontend/src/app/services/users.service.ts

Delete Users: Backend Code

The back-end uses Mongoose features to manipulate data in the database:

```
router.post('/deleteUser', function(req, res, next) {
  User.remove({_id : req.body.id}, function(err) {
    if (err) {
      console.log("not removed!");
      res.status(400);
      res.send();
    }

    console.log("removed!");
    res.send({status: 'ok'});
  });
});
```

/mean_backend/routes/index.js

Implementation

Now, let’s execute the code for the *Deleting users* operation below.

```
#!/usr/bin/env node

/**
 * Module dependencies.
 */

var app = require('../app');
var debug = require('debug')('mongodbnode:server');
var http = require('http');
var mongoose = require('mongoose');

var mongoDB = 'mongodb://127.0.0.1/blog';
mongoose.connect(mongoDB, {
  useMongoClient: true
});

//Get the default connection
var db = mongoose.connection;

//Bind connection to error event (to get notification of connection errors)
db.on('error', console.error.bind(console, 'MongoDB connection error:'));

/**
 * Get port from environment and store in Express.
 */

var port = normalizePort(process.env.PORT || '3000');
app.set('port', port);

/**
 * Create HTTP server.
 */

var server = http.createServer(app);

/**
 * Listen on provided port, on all network interfaces.
 */

server.listen(port);
server.on('error', onError);
server.on('listening', onListening);

/**
 * Normalize a port into a number, string, or false.
 */

function normalizePort(val) {
  var port = parseInt(val, 10);
  if (isNaN(port)) {
    // named pipe

    return val;
  }

  if (port >= 0) {
    // port number
    return port;
  }

  return false;
}

/**
 * Event listener for HTTP server "error" event.
 */

function onError(error) {
  if (error.syscall !== 'listen') {
    throw error;
  }

  var bind = typeof port === 'string'
    ? 'Pipe ' + port
    : 'Port ' + port;

  // handle specific listen errors with friendly messages
  switch (error.code) {
    case 'EACCES':
      console.error(bind + ' requires elevated privileges');
      process.exit(1);
      break;
    case 'EADDRINUSE':
      console.error(bind + ' is already in use');
      process.exit(1);
      break;
    default:
      throw error;
  }
}

/**
 * Event listener for HTTP server "listening" event.
 */

function onListening() {
  var addr = server.address();
  var bind = typeof addr === 'string'
    ? 'pipe ' + addr
    : 'port ' + addr.port;
  debug('Listening on ' + bind);
}
```

> Try inserting a few users and then click the delete button to remove them.

This was the complete implementation, including the front-end and the back-end of the *Delete Users* operation. In the next lesson, we will run the whole application.