Exercise (Instructions): Express Generator | Coursera

Objectives and Outcomes

In this exercise, you will use the Express generator to scaffold out an Express application. Thereafter you will modify the application to support REST API making use of the Node modules that you developed as part of the first assignment. At the end of this exercise, you will be able to:

- Generating an Express application using the express-generator
- Modify the Express application to support the REST API by adding routes

Installing express-generator

- Install *express-generator* by typing the following at the prompt:

```
     npm install express-generator -g
```

Use sudo if you are using an OSX or Linux machine.

Scaffolding an Express Application

- To scaffold out an Express application, type the following at the prompt:

```
     express node-express-gen
```

- Next, move to the *node-express-gen* folder. Type the following at the command prompt to install all the Node modules

```
     npm install
```

- You can start the Express server by typing the following at the prompt:

```
     npm start
```

- Now, copy the *dishRouter.js*, *promoRouter.js* and *leaderRouter.js* from your first assignment (node-express folder) to the routes folder within the Express application that you just scaffolded out.
- Furthermore, copy the index.html and aboutus.html file from the *node-express/public* folder to the public folder in your new project.
- Then, open the app.js file and then update the code in there as follows:

```
var express = require('express');
var path = require('path');
var favicon = require('serve-favicon');
var logger = require('morgan');
var cookieParser = require('cookie-parser');
var bodyParser = require('body-parser');

var routes = require('./routes/index');
var users = require('./routes/users');
var dishRouter = require('./routes/dishRouter');
var promoRouter = require('./routes/promoRouter');
var leaderRouter = require('./routes/leaderRouter');

var app = express();

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'jade');

// uncomment after placing your favicon in /public
//app.use(favicon(path.join(__dirname, 'public', 'favicon.ico')));

app.use(logger('dev'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(cookieParser());

app.use(express.static(path.join(__dirname, 'public')));

app.use('/', routes);
app.use('/users', users);
app.use('/dishes',dishRouter);
app.use('/promotions',promoRouter);
app.use('/leadership',leaderRouter);

// catch 404 and forward to error handler
app.use(function(req, res, next) {
  var err = new Error('Not Found');
  err.status = 404;
  next(err);
});
```

```
// error handlers
// development error handler
// will print stacktrace
if (app.get('env') === 'development') {
  app.use(function(err, req, res, next) {
    res.status(err.status || 500);
    res.render('error', {
      message: err.message,
      error: err
    });
  });
}

// production error handler
// no stacktraces leaked to user
app.use(function(err, req, res, next) {
  res.status(err.status || 500);
  res.render('error', {
    message: err.message,
    error: {}
  });
});

module.exports = app;
```

- Save the changes and run the server. You can then test the server by sending requests and observing the behavior.

Conclusions

In this exercise, you learnt to scaffold out an Express application using the express-generator. Thereafter you were able to build a REST API server.