How I Setup Angular + Node Projects

Posted on 10 April
by J Cole Morrison

Update: I've rewritten the article to be cleaner and setup for a long series of building an Express and Angular application. It covers everything this one does and a bit more. It also does this setup with Express 4.x instead of 3.x. Check it out here.

Update: This uses Express 3.X. Thanks to Jeff Ward for catching this. I'll rewrite it for 4.x very soon.

Mongo + Node setup... the helpful resources on how to do so were scarce.

wat. they're everywhere.

They are.. but the majority of them are little pet project examples full of my FAVORITE saying in tutorials:

"We're going to do this. You'd never do this in the real world, but we're just going to do this here."

Anyhow, depending on the level of interest, I'll continue walking through how I setup everything from accounts to security to payments to SSL. But for now, all this is is setting up the development environment. Just getting all the pieces can often turn into a nightmare, so I wanted to share how I do it. This setup covers the following:

1. Making the server nothing more than a REST service
2. Keeping all tools and generators easily updated and independent
3. Making deployment relatively simple.
4. Have unit tests available separately for the front-end. (if you're in to that type of thing)
5. Uses the magic of Yeoman and Generators for quick scaffolding.
6. Read #5 again. Because Yeoman is that cool.

Vital Tip for New MongoDB'ers How I Fight Procr

Prerequisites:

Before continuing, you should have node.js setup on your computer. I'm also doing this on Mac OSX because, for me at least, developing on windows is a nightmare. I'm also not going to talk about the basics of node, npm, express, or angular. Sorry, but there's PLENTY of those out there. I'll sprinkle in little lessons as I go if they seem relevant.

I'm also going to be doing a lot of commands in Terminal. So I hope you use that too...

The Tools

Before I state them, know that I put a great deal of thought into their selection. You could also read this as I wasted an exorbitant amount of time stressing out about which to choose. The biggest criteria in their selection is how often their maintained and their flexibility with adding other packages. I hate running into abandonware or projects that are overtly opinionated in what you should use.

1. Angular JS (obviously)
2. Yeoman - great scaffolder. saves TONS of time.
3. Generator-Angular
4. Express
5. Bower
6. Nodemon

This is all for now. If I wind up expanding on this tutorial, there's a ton more.

Getting Started

- Install Node.js

- Install Express

On the command line:

```
$npm install -g express
```

- Install Yeoman & Generator-Angular

On the command line:

```
$npm install -g yo

$npm install -g generator-angular
```

Generator-Angular is a tool within Yeoman. It takes care of setting up folder structure for you, getting SASS working, putting in bootstrap if you want it, creating routes... lots of goodies. Learn more about it here. In fact, Yeoman is kind of the reason I went through all of this trouble. I wanted to use it in my workflow.

- Install Bower

`$npm install -g bower`

This is Twitter's front-end package manager. It makes it easy to grab front-end assets likle bootstrap-sass and underscore.js.

- Install Nodemon

`$npm install -g nodemon`

This makes it so that you don't have to restart your node server everytime you change code in it.

Setup Folder Structure

Somewhere on your disk, just pick a folder that you want all of this magic to happen in. Inside of it make two folders: `client` and `server`. Also, get inside of your client folder. What it looks like on the command line:

```
$mkdir project-name
$cd project-name
$mkdir client
$mkdir server
$cd client
```

The client folder will house your angular and front-end code while the server folder will house your node and express code (among other things).

Scaffold out the Front End

Now we're going to setup our angular and front-end code using the amazingness of Yeoman. Do the following on the command line inside of your `client` folder:

```
$yo angular
```

Yeoman will ask you to supply it some options. Say yes to Sass/compass (because you're not crazy) and yes to bootstrap and yes to the sass version of bootstrap. Also accept all of the extra angular packages (resource, cookies, sanitize, route, etc) you'll need them. NPM will begin it's usual command line vomit of all the different packages.

Note: Sometimes the installation will have a brain fart here. If it does delete your client folder, remake it and do a `$bower cache clean` *and a* `$npm cache clean`. *From there, do* `$yo angular` *again. If that doesn't do it, make sure that your Node and Yeoman are up-to-date as well.*

To make sure that everything is working do the following on the command line while inside of your client folder:

```
$grunt serve
```

This will fire up the angular app and it will open up in your browser. What happens here is that Yeoman creates a bunch of temporary directories and serves them while you develop the app. However, once it's built, you'll use the command `grunt` to minify and optimize assets that it will put into a different directory.

Before we leave the client area we need to make one more change. Open up `client/Gruntfile.js` and change the `dist` output around line 25:

```
// Project settings
yeoman: {
  // configurable paths
  app: require('./bower.json').appPath || 'app',
  dist: '../server/dist'
},
```

This will make it so that Yeoman creates an optimized version of our app in the `../server/dist/` folder when we're ready.

Press `ctrl+c` to end yeoman.

Scaffold out the Back End

Now get into your server folder.

```
$cd ../server/
```

Use the `express` command to get a basic scaffolding inside of your server folder.

```
$express
$npm install
```

This is going to give you a variety of things, most of which we'll be deleting. I find this far more expedient vs typing out all of the scaffolding yourself. Half the time I'll forget a few of the express settings and wind up bashing my head against the computer thinking it's something else.

Open up your `server` folder and delete the `public` folder and the `views` folder. Create a folder inside of `server` called `dist`. Open up the file `server/app.js` and delete the following lines:

```
app.set('views', __dirname + '/views');
app.set('view engine', 'jade');
```

and also

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

and finally

```
// development only
if ('development' == app.get('env')) {
    app.use(express.errorHandler());
}

app.get('/', routes.index);
app.get('/users', user.list);
```

We delete these because we don't need a server templating engine (angular will be handling that), we'll have to setup our own public directories, and there's no need for those routes (yet).

Next we need to make it so that our express server serves files from our `client` directory when we're in "development" mode and from a `server/dist` directory when in "production" mode. Add the following lines to your `server/app.js` file:

```
/**
 * Development Settings
 */
```

```
if ('development' == app.get('env')) {
// This will change in production since we'll be using the dist
folder
// This covers serving up the index page
app.use(express.static(path.join(__dirname, '../client/.tmp')));
app.use(express.static(path.join(__dirname, '../client/app')));
app.use(express.errorHandler());
}

/**
 * Production Settings
 */
if('production' == app.get('env')) {
app.use(express.static(path.join(__dirname, '/dist')));
}
```

This is probably a little hairy at this point, so let's get through the rest of the setup and I'll explain what the point of this is. For now, all you need to know is that when we're in development we'll serve from a directory in the `client` folder and when we're in production we'll serve from the `server/dist` folder.

Testing the Development Server

Somewhere, some developer is probably going to puke at this method, but the simplicity is just too good for me to let go of. Do the following:

```
$cd ../client/
$grunt serve
```

This will start up the yeoman server. Inside of here you'll be able to work with SASS, add angular routes and assets available through the Generator

Angular path, manage dependencies with Bower, and all sorts of goodies. Now, press `cmd+t` to open up a new tab in terminal (so you'll have one terminal tab for your `client` and one for your `server`). Do the following:

```
$cd ../server/
$nodemon app.js
```

Express will fire up. Navigate to `localhost:3000` in your browser and you should see the same yeoman start page that you got via the `grunt serve` method.

Yes. You have two servers running (and two terminal tabs). I know it's weird, but I'll explain why this works out at the end. Just bear with me.

To shut things down, in your `client` terminal tab press `ctrl+c` to shut it down and in your `server` tab press `ctrl+c` to also shut it down.

Testing the Production Server

Back over in your `client` terminal tab do the following:

```
$grunt --force
```

This command tells Yeoman to optimize your app for production. You can see what is being optimized by reading through the `client/Gruntfile.js` file. The `--force` option tells it to ignore unit tests for now. If you want unit tests on, just use plain `grunt`.

Tab back over to your `server` tab and do the following:

```
$NODE_ENV=production nodemon app.js
```

This tells express to run your application in production mode and serves your newly optimized angular app from `server/dist` instead of your client folder.

This server folder is what you deploy when ready.

Note: You'll notice that the glyhpicons aren't showing up. That's because the Gruntfile isn't including the icons from bootstrap. We'll fix that later by switching out Glyhpicons for Font Awesome. Font Awesome is better anyway.

What the Hell Did I Just Do..

Alright, alright. So this was probably pretty weird so let me explain just what's going on and how I use it in workflows.

- I set things up in development mode (which is where I have the two terminal tabs open. One is in the `client` folder running `grunt serve` and the other is in the `server` folder running `nodemon app.js`).
- I develop my app with yeoman, SASS, and express. As you develop you watch what the Express server is doing. In otherwords, keep your browser open to `localhost:3000` or whatever Express is serving, not what Grunt /Yeoman is serving.
- When I'm done developing I'll do a `grunt` in my `client` terminal tab. This will optimize things and put them into the `server/dist` folder. I'll then deploy my `server` folder to my host and everything is good to go.
- When something like Angular or Express or Yeoman or Generator Angular needs an update, I do so with no stress.

So I'll generally have 3 terminal tabs open. 1 dedicated to the client running the `grunt serve`; 1 dedicated to running `yo:` commands for angular scaffolding and `bower` commands for adding in packages; and 1 dedicated to running the node server.

And That's How -I- Set Things Up

I'm not saying this is the most optimal or best way. This is just how **I** do it. I've fiddled with a ton of other setups, including using some Yeoman generators dedicated to express + angular, but most of them wind up getting abandonded or are just too opinionated in their use of packages. If you're already a user of Yeoman and Generators you've probably tried to get things working with Express already... and if so the utility of this method probably makes a lot more sense.

Next on the docket will be workflow with front-end code followed by setting up the REST services on the backend...assuming there's interest.

I'm sure there are some typos here and there, so if you see them, please put them in the comments so that I can correct ASAP.

About the author

J Cole Morrison

Sacramento, CA

http://jcolemorrison.com

Comments

26 Comments **start.jcolemorrison.com** 1 **Login**

Recommend **Share** **Sort by Best**

Join the discussion…

Jeff Ward • 2 years ago

This based on Express 3.

I started from scratch with updated everything and ran into a bunch of problems because of differences in the latest versions.

1) the express generator requires a separate install now (npm install -g generator-express), 'express' command does not generate anything on its own anymore.

2) express.errorHandler() has been decoupled. I used the default app.use(function(err, req, res, next){ ... that came with Express 4

3) You need to actually setup the server with a var server = app.listen(3000,callback()); , there is no default.

I had a pretty tough time grinding it down to work considering how simple it is, I think it needs a rewrite for Express 4. Thanks for the effort though!

Here is the barebones app.js I ended up with.
https://gist.github.com/jeffsc...

4 • Reply • Share ›

Cole Morrison Mod Jeff Ward • 2 years ago

Wow, thanks so much for that catch! I hadn't been keeping up with the express release cycles. I'll rewrite it for express 4 for sure.

• Reply • Share ›

Rajini Kumar • 7 months ago

I am using node version v0.10.37 and npm version 3.5.3. Which one I follow?
I followed this url:

When I run this command >> grunt serve - inside the client folder getting error

"Fatal error: Unable to find local grunt.
If you're seeing this message, either a Gruntfile wasn't found "

• Reply • Share ›

HaeckDesign • 2 years ago

After a couple tweaks (for v4 as you mentioned) - Got everything running smoothily. Very intuitive logic on your workflow! (...which might be the nerdiest sentence I've ever said and meant)

• Reply • Share ›

Sandeep Kalra • 2 years ago

sorry if I sound stupid as I am novice to all this.. but it looks like in the end, express is serving the webpages too (from the dist folder).. Grunt is serving on port 9000, which user never visits. please help me clarify this.

• Reply • Share ›

Cole Morrison Mod Sandeep Kalra • 2 years ago

Yep. Grunt just provides the server which you can visit, but what you really need that server for is all of the work it does (minification, crunching down sass files, etc).

• Reply • Share ›

Sandeep Kalra Cole Morrison • 2 years ago

Thanks for confirming. Is there a way that UI is served altogether separately (say by $http-server - the base server in angular-seed), and just call express for RESTful calls. Its been 2 weeks since I am looking for some example that can do so. I would really like to see if you can suggest some repo/work done similar to what I am looking for. Again, I am novice and have just started 3-4 weeks back, so, I am not knowing a lot. I am learning everyday little by little.

• Reply • Share ›

Wilmer E. López Rivas • 2 years ago

Nice article! Thanks

• Reply • Share ›

Ennio Wolsink • 2 years ago

This article has been very helpful for me, as I'm setting up my first ever *real* NodeJS/Express/Angular JS-project. I'll probably find my own way eventually, but in the meantime I'm glad to have stumbled upon this guide :)

• Reply • Share ›

Cole Morrison Mod Ennio Wolsink • 2 years ago

Awesome! The little series I have together is literally how I setup production projects.

2 • Reply • Share ›

Willson Mock • 2 years ago

Just wanted to say thanks for sharing your workflow!

• Reply • Share ›

Cole Morrison Mod Willson Mock • 2 years ago

No problem! This is one is outdated though, I'd do the updated version instead that works with the newest generator-angular and express:

http://start.jcolemorrison.com...

• Reply • Share ›

Matt Rust • 2 years ago

The only problem I have is the generated dynamically named images aren't being updated in my html or css, so I get 404 errors in production. This was my first time using yeoman and grunt at all, any ideas?

• Reply • Share ›

Cole Morrison Mod Matt Rust • 2 years ago

Hmm, just curious, after doing a build (grunt --force if you're not doing tests), check out your server/dist/images. Are they being put there? Also, check out your client/package.json. See if under "devDependencies" if this is there:

"grunt-contrib-imagemin": "~0.3.0",

That's the task that minifies the images.

I updated this tutorial for Express 4.x (Building an Angular and Express App Part 1), maybe give that a go if it doesn't work.

• Reply • Share ›

Matt Rust Cole Morrison • 2 years ago

I did have them in server/dist/images but the css and html didn't point to them, I did already do this on my own in Express 4, but it looks like you had walked through some things that I missed. Thanks.

• Reply • Share ›

Cole Morrison Mod Matt Rust • 2 years ago

No problem! So you got it working then?

• Reply • Share ›

srfrnk • 2 years ago

Nice article! I started with this not so long ago. Actually found some stuff was missing so I went and created a more inclusive generator - called it SiLi - you can check it at https://www.npmjs.org/package/... - would love to hear your comments about it.

• Reply • Share ›

Cole Morrison Mod srfrnk • 2 years ago

Very cool, I will definitely take a look at it!

• Reply • Share ›

srfrnk Cole Morrison • 2 years ago

Nice to know :)
I'm constantly updating it... Just now thinking about adding 'tedious' and 'busboy' - or maybe you think of better alternatives?

• Reply • Share ›

Elliot Fielstein • 2 years ago

Very nice approach, creating and configuring an express server while still using Yeoman's awesome scaffolding and build/test setup! I had no problems with this setup on Windows 7 using Node 10.x and Express 3. The only Windows-specific change was in setting the production environment at the command line for the build. I use PowerShell which requires setting $env:NODE_ENV="production". Otherwise, worked like a charm. Really look forward to the follow-up, integrating MongoDB and configuring routing on the backend. Great stuff, and thanks!!

• Reply • Share ›

Cole Morrison Mod Elliot Fielstein • 2 years ago

Not a problem! I should probably update this for Express 4.x first since I just realized I'm going to have to do that for a few applications I'm maintaining. :\

• Reply • Share ›

HHrvoje • 2 years ago

Article is great, but is there some showstopper for doing all this on windows? Node works great there, but didn't try anything else...

• Reply • Share ›

Cole Morrison Mod HHrvoje • 2 years ago

No, there isn't. Just making sure that you get all the right command line tools. I just didn't want to claim that the exact process I was using would work for windows.

• Reply • Share ›

amitava • 2 years ago

Excellent guide. I really like this approach of keeping both server and client separate. It's simple and clean.

One thing I would like to know how do you structure/organize your server-side code (mvc etc). Another blog post perhaps :)

• Reply • Share ›

Aaron Rosario • 2 years ago

Awesome setup guide! Only errors I saw were "this twitter" and "mor sense". I'd definitely enjoy seeing more of your process to try it out.

• Reply • Share ›

Cole Morrison Mod Aaron Rosario • 2 years ago

Ah. Thanks for the catch there! And for sure!

• Reply • Share ›

ALSO ON **START.JCOLEMORRISON.COM**

How to Try Out Angular in your Existing Rails App

1 comment • 6 months ago•

Allen Maxwell — Thanks for the post. Very helpful. I'd love to see how you'd use resources to pull data from a database …

Journaling a New Start(up)

1 comment • 2 years ago•

Aspire Subedi — inspirational, best of luck!

On Picking Your Next Idea

1 comment • 2 years ago•

Yer a Hacker Harry!

3 comments • 2 years ago•

Joshua Drake — Thank you for this article. It was a good reminder, yet gave a fresh perspective.

Steve — If you define magic as simply an arcane set of knowledge where you can create things, manipulate things, or …

Subscribe Add Disqus to your site Add Disqus Add Privacy