Scaling Angular: Enterprise SOA on the MEAN Stack

Levent Gurses
@gursesl

April 2015

Agenda

App Structure

Development Practices

Angular 2.0 & Beyond

Q & A

What is MEAN?

- Mongo
- Express
- Angular
- Node

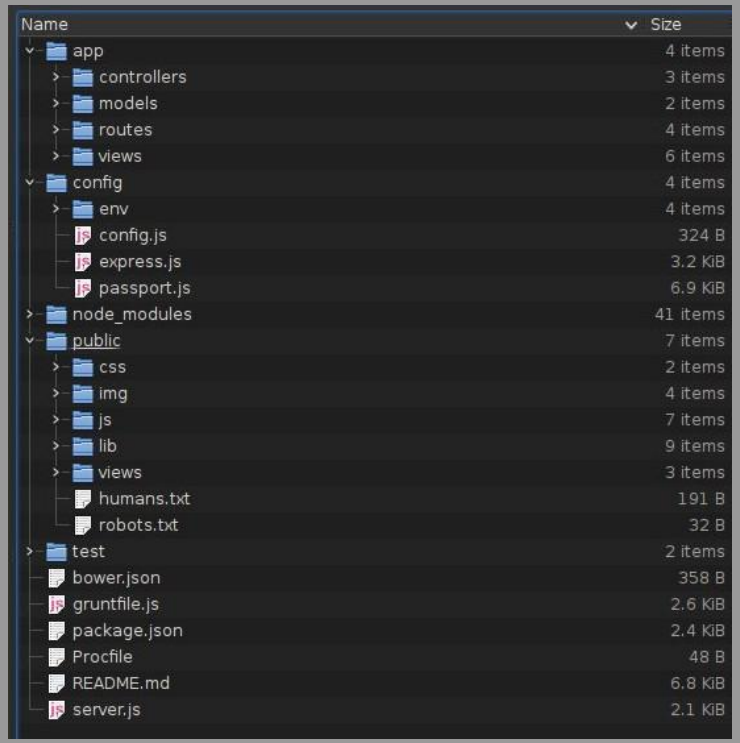
Let’s Define Scalability

- **Development practices** - as complexity grows your app should be able to maintain a coherent structure
- **Maintainability** - new developers should be able to learn and maintain the system relatively quickly
- **Performance** - as the load increases your app should adjust itself to handle the load accordingly

Full-Stack Apps

- Server code
  - Routes
  - Modules
  - ...
- Client code
  - Routes
  - Modules
  - ...
- Configuration files
- Build files

Full-Stack Folder Structure



Generators & Skeletons

- **Yeoman**
  - yo meanstack
- **GitHub**
  - MEAN Stack
  - Mean.io
  - DSTRoot Skeleton

Angular Folder Structure

Recursive structure, the base unit of which contains a module definition file (**app.js**), a controller definition file (**app-controller.js**), a unit test file (**app-controller_test.js**), an HTML view (**index.html** or **app.html**) and some styling (**app.css**) at the top level, along with directives, filters, services, protos, e2e tests, in their own subdirectories.

Components & Subsections

Elements of an application are grouped either under a "**components**" directory (for common elements reused elsewhere in the app), or under meaningfully-named directories for recursively-nested **sub-sections** that represent structural "view" elements or routes within the app

Components

- A components directory contains **directives, services, filters, and related files**
- Common data (images, models, utility files, etc.) might also live under components (e.g. components/lib/), or it can be stored externally
- Components can have subcomponent directories, including appended naming where possible
- Components may contain module definitions, if appropriate

Sub-Sections

- These top-level (or nested) directories contains ***only*** **templates (.html, .css), controllers, and module definitions**
- Sub-level child sub-sections are stamped using the same unit template (i.e., a section is made up of templates, controllers, and module definitions), going deeper and deeper as needed to reflect the inheritance of elements in the UI
- For a very simple app, with just one controller, sub-section directories might not be needed, since everything is defined in the top-level files
- Sub-sections may or may not have their own modules, depending on how complex the code is

Module Definitions

- In general, **'angular.module('foo')'** should be called only once. Other modules and files can depend on it, but they should never modify it
- Module definition can happen in the main module file, or in subdirectories for sections or components, depending on the application's needs

Naming Conventions

- Each filename should describe the file's purpose by including the component or view sub-section that it's in, and the type of object that it is as part of the name. For example, a datepicker directive would be in **components/datepicker/datepicker-directive.js**
- Controllers, Directives, Services, and Filters, should include controller, directive, service, and filter in their name
- File names should be lowercase, following the existing JS Style recommendations. HTML and css files should also be lowercase.
- Unit tests should be named ending in _test.js, hence "**foo-controller_test.js**" and "**bar-directive_test.js**"
- Partial templates should be named foo.html rather than foo.ng. (This is because, in practice, most of the templates in an Angular app tend to be partials.)

Simple App Example

Consider a very simple app with one directive and one service:

```
sampleapp/                  the client app and its unit tests live under this directory
   app.css
   app.js
   app-controller.js
   app-controller_test.js
   components/          module def'n, services, directives, filters, and related
      foo/                   files live here. "foo" describes what the module does.
            foo.js       The 'foo' module is defined here.
                  foo-directive.js
                  foo-directive_test.js
                  foo-service.js
                  foo-service_test.js
   index.html
```

Simple App Example

Or, in the case where the directive and the service are unrelated, we'd have:

```
sampleapp/
  app.css
  app.js
  app-controller.js
  app-controller_test.js
  components/
    bar/                                  "bar" describes what the service does
      bar.js
      bar-service.js
      bar-service_test.js
    foo/                                  "foo" describes what the directive does
      foo.js
      foo-directive.js
      foo-directive_test.js
   index.html
```

Complicated App Example

```
sampleapp/
        app.css
        app.js                                  top-level configuration, route def'ns for the app
        app-controller.js
        app-controller_test.js
        components/
                adminlogin/
                        adminlogin.css                  styles only used by this component
                        adminlogin.js                optional file for module definition
                        adminlogin-directive.js
                        adminlogin-directive_test.js
                  private-export-filter/
                          private-export-filter.js
                          private-export-filter_test.js
                userlogin/
                        somefilter.js
                        somefilter_test.js
                        userlogin.js
                        userlogin.css
                        userlogin.html
                        userlogin-directive.js
                        userlogin-directive_test.js
                        userlogin-service.js
                        userlogin-service_test.js
        index.html
        subsection1/
                subsection1.js
                subsection1-controller.js
                  subsection1-controller_test.js
                  subsection1_test.js
                  subsection1-1/
                          subsection1-1.css
                          subsection1-1.html
                          subsection1-1.js
                          subsection1-1-controller.js
                        subsection1-1-controller_test.js
                  subsection1-2/
        subsection2/
                subsection2.css
                subsection2.html
                subsection2.js
                subsection2-controller.js
                subsection2-controller_test.js
        subsection3/
                  subsection3-1/
                          etc...
```

Testing

- Unit tests - Jasmine - mock data - $httpBackend
- AATs - Protractor - mock data - $httpBackend
  - Requires a browser
- Integration tests - real data
- Karma runner
- Coverage

Build and Deployment

- npm package management
- grunt - serial
- gulp - parallel (10X faster)
- NodeJS server requires little configuration
- Deployment means moving compiled files to **/public** folder

Compilation

- CoffeeScript
- CSS - LESS, SASS, STYLUS
- HTML Templates - Jade, Moustache
- JS minification
- CSS, HTML, Image minification

Continuous Integration

- Every check-in triggers
  - Compilation
  - Unit tests
  - Functional tests
  - Integration tests
  - JSHint, JSLint
  - Istanbul (Code coverage tool)
- TeamCity, Jenkins

Continuous Delivery

- Deploy to [production] multiple times a day
- Progression over CI
- Virtualization
- Scripted infrastructure w/ failover
  - Puppet, Chef, Vagrant
  - Docker, Fig, Consul
- Unlimited environments
- CD pipeline

Cloud Infrastructure

- Scripted infrastructure
- Amazon EC2 - custom Node environments
- Heroku - ready to use Node servers
- Parse.com
- MongoLabs

Security

- CORS
- NodeJS is a secure, new-gen application server
- Passport - Node module for auth/authorization

Enterprise SOA

- Careful planning
- Development cycle
- Front-end dictates the protocol (REST, JSON)
- One team
- Full-stack development

Angular 2.0 & Beyond

- Radical changes to Angular
  - Controllers
  - Services
  - Directives
  - DI
- To be released in Q3 2015

Web Components

- Custom Elements - Enables the extension of HTML through custom tags
- HTML Imports - Enables packaging of various resources (HTML, CSS, JS, etc.)
- Template Element - Enables the inclusion of inert HTML in a document
- Shadow DOM - Enables encapsulation of DOM and CSS

In closing...

- Project structure important to scale apps
- Development practices key to success
- Each App is Different
- Experiment Until You Find A Formula That Works For You

Thank you.

Resources

- http://www.movel.co
- http://www.meetup.com/mobile-dc
- http://www.meetup.com/full-stack-dc/
- http://www.meetup.com/javascript-dc/
- http://www.meetup.com/startup-dc/
- http://www.movel.co/events/introduction-es6-tutorial/
- http://www.meetup.com/polymer-dc/
- https://angularjs.org/
- http://mean.io/