# Create Your Own Yeoman Generator

[Yeoman's](http://yeoman.io/) generators are what give its platform flexibility, allowing you to reuse the same tool for any kind of project you may be working on, JavaScript or otherwise. See also the enormous library with [community generators](http://yeoman.io/community-generators.html). Sometimes though, you may have some specific setup that you like to employ in your own projects and for situations like this, it makes sense to abstract all the boilerplate into your own generator.

In this article, we will be building a Yeoman generator that will allow us to build a single page site with AngularJS or JQuery, SASS, Bootstrap and 'Modernizr'.

## Getting Set Up

I am going to assume you have [Node.js](http://nodejs.org/) setup, if not, you can grab the installation from [here](http://nodejs.org/download/). Besides that, we will need to have [Yeoman](http://yeoman.io/) and [Bower](http://bower.io/) installed as well as the generator ([**yo**](https://www.npmjs.com/package/yo)) for creating generators. You can accomplish this via the following commands to **npm**:

|  |  |
| --- | --- |
|  | **npm install -g -bower**  **npm install -g yo**  **npm install -g generator-generator** |

First, create a folder within which you'll write your generator. This folder must be named generator-name (where name is the name of your generator). This is important, as Yeoman relies on the file system to find available generators.

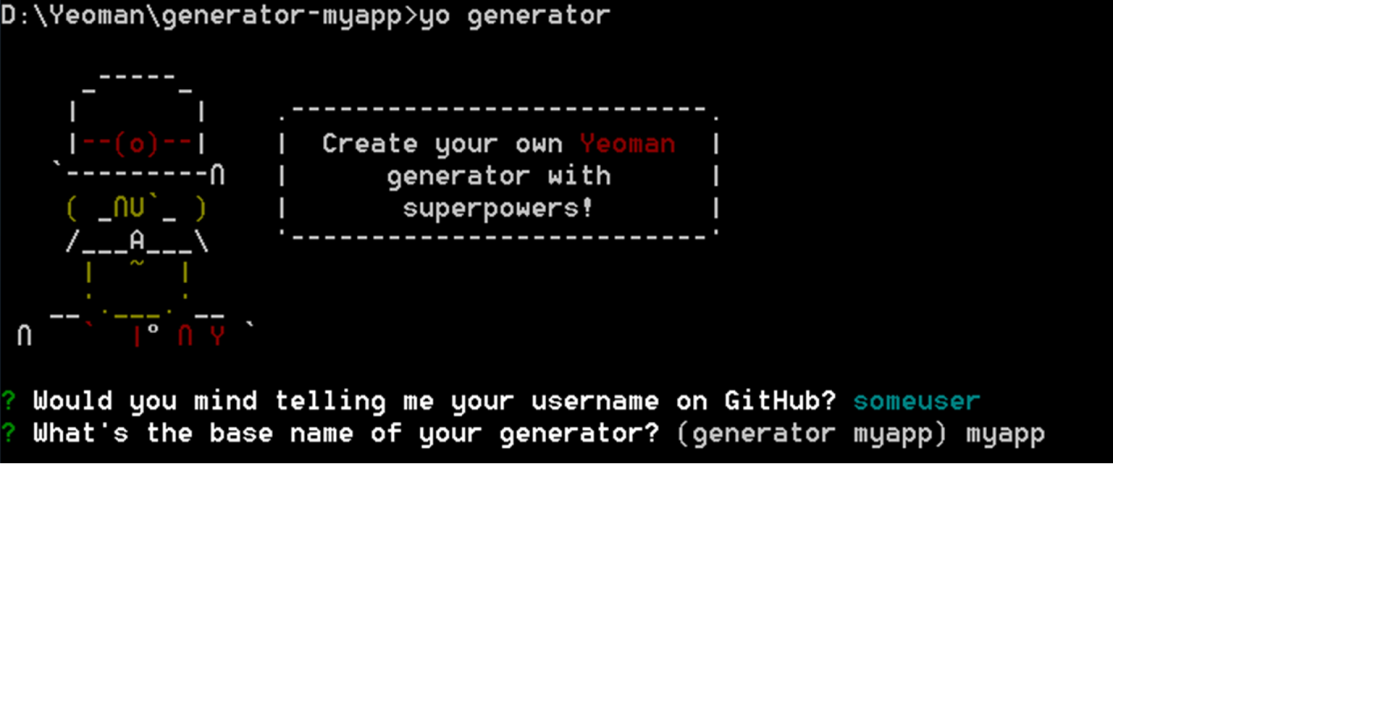
**mkdir generator-myapp**

**cd generator-myapp**

And generate in this folder your basic generator:

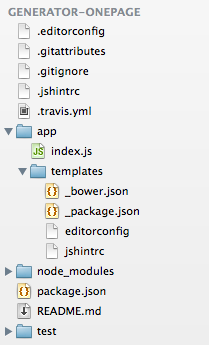
|  |  |
| --- | --- |
|  | **yo generator** |

This will start the generator and ask you some questions like the project name and your GitHub account. I am going to name my generator: **myapp**.



## The File Structure

These are the files generated by the command, it will all make sense in just a moment.



The first couple of files are dotfiles for various things like Git and Travis CI, we have apackage.json file for the generator itself, a readme file and a folder for tests.

The .yo-rc.json  is very important to locate the root of the project.

The generators folder holds each sub-generator. Each folder has the same name as the sub-generator name. In our case we have one (default) generator app, that is called with command:

**yo myapp**. Each sub-generator has an **index.js** file, that contains the entry point for the generator, and a templates folder that contains the template files for the boilerplate (for generating the actual scaffolding).

The **test** folder holds the tests for the yeoman generator.

Now you have a default generator ready. So before modifying it to add our custom features, we will test this base generator. Since you’re developing the generator locally, and haven’t published it yet, you have to symlink your local module as a global module, using npm. Why?

This is handy for installing your own stuff, so that you can work on it and test it iteratively without having to continually rebuild. Run this command in your generators root directory (the root folder is where you can find the: .yo-rc.json ).

**npm link**

Make a new folder on your filesystem (i.e. outputfolder) for your application and test your generator. You can scaffold your very own web app now:

**mkdir testmyapp**

**cd testmyapp**

**yo myapp**

A new scaffolding will be generated in your outputfolder: **testmyapp**

# How to Script your Generator

Yeoman generator-generator generated a base structure for our generator, now we will check how to customize it and add our own features in the generator by the following steps:

1. Setup the actual generator object
2. Initializing the generator
3. Getting user inputs
4. Options and arguments
5. Adding custom templates
6. Write the generator specific files
7. Install npm and Bower
8. Scaffold your app
9. Unittest

# Setup the actual generator object

The index.js file needs to export the actual generator object (MyappGenerator) which will get run by Yeoman. I am going to clear everything inside the actual generator so we can start from scratch, here is what the file should look like after that:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | |  | | --- | | 'use strict';  var fs = require('fs');  var path = require('path');  var yeoman = require('yeoman-generator');  var yosay = require('yosay');  var chalk = require('chalk');  var wiredep = require('wiredep');    var MyappGenerator = yeoman.generators.Base.extend({    });    module.exports = MyappGenerator; | | |  |

A Yeoman Generator can extend from the Base generator.

All that this code in the **index.js** is doing is creating the generator object: ‘MyappGenerator’ and exporting it out, Yeoman will actually retrieve the exported object and run it. The way it runs, is by first calling the constructor method to set the object up and then it will go through all the methods you create on this object (in the order you created them) and run them one at a time. Some method names have priority in this generator. The available priorities are in order:

1. initializing - Your initialization methods (checking current project state, getting configs, etc)
2. prompting - Where you prompt users for options (where you'd call this.prompt())
3. configuring - Saving configurations and configure the project (creating .editorconfig files and other metadata files)
4. default
5. writing - Where you write the generator specific files (routes, controllers, etc)
6. conflicts - Where conflicts are handled (used internally)
7. install - Where installation are run (npm, Bower)
8. end - Called last, cleanup, say *good bye*, etc

I refer to the [running context](http://yeoman.io/authoring/running-context.html) for more information about how methods are run and in which context.

# Initializing the generator

The first thing you have to do is initializing your generator with the **package.json** file. This file is a Node.js module manifest. You can find this file in the root folder of the generator ( the root folder is the place where.yo-rc.json is located).

|  |
| --- |
| The package.json file must contain the following:{  "name": "generator-myapp",  "version": "0.0.0",  "description": "Scaffold out a front-end web app",  "license": "BSD",  "repository": "petereijgermans11/generator-myapp",  "author": {  "name": "",  "email": "",  "url": "https://github.com/petereijgermans11"  },  "main": "app/index.js",  "engines": {  "node": ">=0.10.0",  "npm": ">=1.3"  },  "scripts": {  "test": "mocha --reporter spec"  },  "files": [  "app"  ],  "keywords": [  "yeoman-generator",  "web",  "app",  "front-end",  "h5bp",  "modernizr",  "jquery",  "angular",  "gulp"  ],  "dependencies": {  "chalk": "^1.0.0",  "wiredep": "^2.2.2",  "yeoman-generator": "^0.18.5",  "yeoman-assert": "^2.0.0",  "yosay": "^1.0.0"  },  "peerDependencies": {  "yo": ">=1.0.0",  "generator-mocha": ">=0.1.0",  "gulp": ">=3.5.0"  },  "devDependencies": {  "mocha": "\*"  }  } |

How this package.json file is composed I refer to [package.json](https://docs.npmjs.com/files/package.json)

I use the following dependencies:

* [**chalk**](https://www.npmjs.com/package/chalk) **:** to log a coloured message with Yeoman
* [wiredep](https://www.npmjs.com/package/wiredep)**: for** injecting Bower components to your HTML/SCSS files.
* **yeoman-generator** see[yeoman](http://github.com/yeoman/generator)
* **yeoman-assert :** Assert utility from [yeoman](http://github.com/yeoman/generator)
* [**yosay**](https://www.npmjs.com/package/yosay)**:** Tell Yeoman what to say in the console

I configured the following peerDependencis which are like plugins:

* [**yo**](https://www.npmjs.com/package/yo): CLI tool for running Yeoman generators
* Generator-[mocha](http://mochajs.org/): a generator for the mocha test-framework
* [Gulp](http://gulpjs.com/): a front-end build tool

And finally I defined the devDependency on the mocha-test test-framework

To initialize the package.json file, we will add the *initializing* method to the index.js:

|  |
| --- |
| 'use strict';  var fs = require('fs');  var path = require('path');  var yeoman = require('yeoman-generator');  var yosay = require('yosay');  var chalk = require('chalk');  var wiredep = require('wiredep');    var MyappGenerator = yeoman.generators.Base.extend({  **initializing: function () {**  **this.pkg = require('../../package.json');**  **}**  });    module.exports = MyappGenerator; |

# Getting user input

Yeoman allows you to add questions to your generator so that you can receive input from the user and customize the behavior appropriately. We are going to have the following questions in our generator:

* 'Would you like AngularJS or JQuery ?
* 'What more front-end frameworks would you like? (SASS, Bootstrap or 'Modernizr').

To accomplish this, we will add the ***prompting*** function to the **index.js**, that will prompt the user for this info and then store the results on our generator object (MyappGenerator) itself:

|  |
| --- |
| 'use strict';  var fs = require('fs');  var path = require('path');  var yeoman = require('yeoman-generator');  var yosay = require('yosay');  var chalk = require('chalk');  var wiredep = require('wiredep');   var MyappGenerator = yeoman.generators.Base.extend({  **initializing: function () {**  **this.pkg = require('../../package.json');**  **},**  **prompting**: function () {  var done = this.async();  if (!this.options['skip-welcome-message']) {  this.log(yosay('Out of the box I include HTML5 Boilerplate, AngularJS, jQuery and a Gulpfile.js to build your app.'));  }  var prompts = [{  type: 'checkbox',  name: 'mainframeworks',  message: **'Would you like AngularJS or JQuery ?**',  choices: [{  name: 'Angular',  value: 'includeAngular',  checked: true  }, {  name: 'JQuery',  value: 'includeJQuery',  checked: true  }]  },  {  type: 'checkbox',  name: 'features',  message: **'What more front-end frameworks would you like ?**',  choices: [{  name: 'Sass',  value: 'includeSass',  checked: true  }, {  name: 'Bootstrap',  value: 'includeBootstrap',  checked: true  }, {  name: 'Modernizr',  value: 'includeModernizr',  checked: true  }]  }  ];  this.prompt(prompts, function (answers) {  var features = answers.features;  var mainframeworks = answers.mainframeworks;  var hasFeature = function (feat) {  return features.indexOf(feat) !== -1;  };  var hasMainframeworks = function (mainframework) {  return mainframeworks.indexOf(mainframework) !== -1;  };  // manually deal with the response, get back and store the results.  // we change a bit this way of doing to automatically do this in the self.prompt() method.  this.includeSass = hasFeature('includeSass');  this.includeBootstrap = hasFeature('includeBootstrap');  this.includeModernizr = hasFeature('includeModernizr');  this.includeAngular = hasMainframeworks('includeAngular');  this.includeJQuery = hasMainframeworks('includeJQuery');  done();  }.bind(this));  }  });    module.exports = MyappGenerator; |

The first line inside the function sets a **done** variable from the object's asyncmethod. Yeoman tries to run your methods in the order that they are defined, but if you run no async code, the function will exit before the actual work gets completed and Yeoman will start the next function early. To get around this, you can call theasync method, which will return a callback and then Yeoman will wait to go on to the next function until that callback gets executed.

Next, we defined a list of prompts, each prompt has a type, a name and a message. If no type was specified, it will default to ‘input' which is for standard text entry.

With the array of questions ready, we can pass it to the prompt method along with a callback function. The first parameter of the callback function is the list of answers received back from the user. We then attach those answers onto our generator object (referenced by ‘this’) and call the **done** method to go on to the next function in the generator object.

# Options and arguments

The user can pass options to generator (index.js) from the command line:

yo myapp --skip-install-message

In our case we like to have options for the following:

**- `--skip-welcome-message`**

Skips Yeoman's greeting before displaying options.

**- `--skip-install-message`**

Skips the message displayed after scaffolding has finished and before the dependencies are being installed.

**- `--skip-install`**

Skips the automatic execution of `Bower` and `npm` after scaffolding has finished.

**- `--test-framework=<framework>`**

Defaults to [mocha](http://mochajs.org/). Can be switched for another supported testing framework like [jasmine](http://jasmine.github.io/).

In our generator the arguments and options should be defined in the **constructor** method:

|  |
| --- |
| var MyappGenerator = yeoman.generators.Base.extend({  **constructor**: function () {  yeoman.generators.Base.apply(this, arguments);  this.option('test-framework', {  desc: 'Test framework to be invoked',  type: String,  defaults: 'mocha'  });  this.option('skip-welcome-message', {  desc: 'Skips the welcome message',  type: Boolean  });  this.option('skip-install', {  desc: 'Skips the installation of dependencies',  type: Boolean  });  this.option('skip-install-message', {  desc: 'Skips the message after the installation of dependencies',  type: Boolean  });  }, |

# Adding custom templates

All I have to do in this section is create all the template files in the: generators/<sub-generator>/**templates** folder. So, the first thing I want to do is to create the following template files in de: generators/<sub-generator>/**templates** folder:

* Gulpfile.js
* \_package.json
* Bowerrc
* Bower.json
* Gitignore
* Gitattributes
* Jshintrc
* Editorconfig
* Favicon.ico, apple-touch-icon.png and robots.txt
* Index.html
* Main.css
* Main.scss
* Main.js

### Gulpfile.js

I want to use Gulp as build system.

For Gulp I need to define a gulpfile.js. I use EJS-styled placeholders in this template file, which will be filled in by our Yeoman-generator at runtime.

### Create the /generator-myapp/generators/app/templates/gulpfile.js containing:

|  |
| --- |
| /\*global -$ \*/  'use strict';  // generated on <%= (new Date).toISOString().split('T')[0] %> using <%= pkg.name %> <%= pkg.version %>  var gulp = require('gulp');  var $ = require('gulp-load-plugins')();  var browserSync = require('browser-sync');  var reload = browserSync.reload;  gulp.task('styles', function () {<% if (includeSass) { %>  gulp.src('app/styles/\*.scss')  .pipe($.sourcemaps.init())  .pipe($.sass({  outputStyle: 'expanded',  precision: 10,  includePaths: ['.']  }).on('error', $.sass.logError))<% } else { %>  return gulp.src('app/styles/\*.css')  .pipe($.sourcemaps.init())<% } %>  .pipe($.postcss([  require('autoprefixer-core')({browsers: ['last 1 version']})  ]))  .pipe($.sourcemaps.write())  .pipe(gulp.dest('.tmp/styles'))  .pipe(reload({stream: true}));  });  function jshint(files) {  return function () {  return gulp.src(files)  .pipe(reload({stream: true, once: true}))  .pipe($.jshint())  .pipe($.jshint.reporter('jshint-stylish'))  .pipe($.if(!browserSync.active, $.jshint.reporter('fail')));  };  }  gulp.task('jshint', jshint('app/scripts/\*\*/\*.js'));  gulp.task('jshint:test', jshint('test/spec/\*\*/\*.js'));  gulp.task('html', ['styles'], function () {  var assets = $.useref.assets({searchPath: ['.tmp', 'app', '.']});  return gulp.src('app/\*.html')  .pipe(assets)  .pipe($.if('\*.js', $.uglify()))  .pipe($.if('\*.css', $.csso()))  .pipe(assets.restore())  .pipe($.useref())  .pipe($.if('\*.html', $.minifyHtml({conditionals: true, loose: true})))  .pipe(gulp.dest('dist'));  });  gulp.task('images', function () {  return gulp.src('app/images/\*\*/\*')  .pipe($.if($.if.isFile, $.cache($.imagemin({  progressive: true,  interlaced: true,  // don't remove IDs from SVGs, they are often used  // as hooks for embedding and styling  svgoPlugins: [{cleanupIDs: false}]  }))  .on('error', function(err){ console.log(err); this.end; })))  .pipe(gulp.dest('dist/images'));  });  gulp.task('fonts', function () {  return gulp.src(require('main-bower-files')({  filter: '\*\*/\*.{eot,svg,ttf,woff,woff2}'  }).concat('app/fonts/\*\*/\*'))  .pipe(gulp.dest('.tmp/fonts'))  .pipe(gulp.dest('dist/fonts'));  });  gulp.task('extras', function () {  return gulp.src([  'app/\*.\*',  '!app/\*.html'  ], {  dot: true  }).pipe(gulp.dest('dist'));  });  gulp.task('clean', require('del').bind(null, ['.tmp', 'dist']));  gulp.task('serve', ['styles', 'fonts'], function () {  browserSync({  notify: false,  port: 9000,  server: {  baseDir: ['.tmp', 'app'],  routes: {  '/bower\_components': 'bower\_components'  }  }  });  // watch for changes  gulp.watch([  'app/\*.html',  'app/scripts/\*\*/\*.js',  'app/images/\*\*/\*',  '.tmp/fonts/\*\*/\*'  ]).on('change', reload);  gulp.watch('app/styles/\*\*/\*.<%= includeSass ? 'scss' : 'css' %>', ['styles']);  gulp.watch('app/fonts/\*\*/\*', ['fonts']);  gulp.watch('bower.json', ['wiredep', 'fonts']);  });  gulp.task('serve:dist', function () {  browserSync({  notify: false,  port: 9000,  server: {  baseDir: ['dist']  }  });  });  gulp.task('serve:test', function () {  browserSync({  notify: false,  open: false,  port: 9000,  ui: false,  server: {  baseDir: 'test'  }  });  gulp.watch([  'test/spec/\*\*/\*.js',  ]).on('change', reload);  gulp.watch('test/spec/\*\*/\*.js', ['jshint:test']);  });  // inject bower components  gulp.task('wiredep', function () {  var wiredep = require('wiredep').stream;  <% if (includeSass) { %>  gulp.src('app/styles/\*.scss')  .pipe(wiredep({  ignorePath: /^(\.\.\/)+/  }))  .pipe(gulp.dest('app/styles'));  <% } %>  gulp.src('app/\*.html')  .pipe(wiredep({<% if (includeSass && includeBootstrap) { %>  exclude: ['bootstrap-sass'],<% } %>  ignorePath: /^(\.\.\/)\*\.\./  }))  .pipe(gulp.dest('app'));  });  gulp.task('build', ['jshint', 'html', 'images', 'fonts', 'extras'], function () {  return gulp.src('dist/\*\*/\*').pipe($.size({title: 'build', gzip: true}));  });  gulp.task('default', ['clean'], function () {  gulp.start('build');  }); |

The gulpfile supports the following:

* CSS Autoprefixing: a postprocessor for CSS appropriate for all  browsers.
* Automagically compile Sass/.scss files with [libsass](http://libsass.org/)
* Minifies all your .css and .js files and html-files
* Map compiled CSS to source stylesheets with source maps
* Built-in preview server with [BrowserSync](http://www.browsersync.io/). BowerSync watch all files and update connected browsers if a change occurs in your
* Automagically lint your scripts via [jshint](https://www.npmjs.com/package/gulp-jshint)
* Image optimization
* Automagically wire-up dependencies installed with [Bower](http://bower.io/)
* Inject Bower components to your HTML/SCSS files via the [wiredep](https://www.npmjs.com/package/wiredep) task.
* Use the .tmp directory mostly for compiling assets like SCSS files. It has precedence over app, so if you had an app/index.html template compiling to .tmp/index.html, your application would point to .tmp/index.html, which is what we want.

For more information on what this generator can do for you, take a look at the*‘*gulp-plugins’used in our *package.json* in the next section. As you might have noticed, gulp plugins (the ones that begin with gulp-) don't have to be require()'d. They are automatically picked up by [gulp-load-plugins](https://github.com/jackfranklin/gulp-load-plugins) and available through the **$** variable.

### \_package.json

### Create the /generator-myapp/generators/app/templates/\_package.json containing:

|  |
| --- |
| {  "private": true,  "engines": {  "node": ">=0.12.0"  },  "devDependencies": {  "autoprefixer-core": "^5.1.8",  "browser-sync": "^2.2.1",  "del": "^1.1.1",  "gulp": "^3.8.11",  "gulp-cache": "^0.2.8",  "gulp-csso": "^1.0.0",  "gulp-if": "^1.2.5",  "gulp-imagemin": "^2.2.1",  "gulp-jshint": "^1.9.2",  "gulp-load-plugins": "^0.8.1",  "gulp-minify-html": "^1.0.0",  "gulp-postcss": "^5.0.0",<% if (includeSass) { %>  "gulp-sass": "^2.0.0",<% } %>  "gulp-size": "^1.2.1",  "gulp-sourcemaps": "^1.5.0",  "gulp-uglify": "^1.1.0",  "gulp-useref": "^1.1.1",  "jshint-stylish": "^1.0.1",  "main-bower-files": "^2.5.0",  "opn": "^1.0.1",  "wiredep": "^2.2.2"  }  } |

### Bowerrc

I use [Bower](http://bower.io/) as the the web package manager.

The default place Bower will install its dependencies is *./bower-components*

Create the /generator-myapp/generators/app/templates/bowerrc containing:

|  |
| --- |
| {  "directory": "bower\_components"  } |

**Bower.json**

Packages are defined by a manifest file bower.json.

Create the /generator-myapp/generators/app/templates/bower.json containing:

|  |
| --- |
| {  "name": "package",  "version": "0.0.0",  "dependencies": {}  } |

### Gitignore

### Create the /generator-myapp/generators/app/templates/gitignore containing:

|  |
| --- |
| node\_modules  dist  .tmp  .sass-cache  bower\_components  test/bower\_components |

### Gitattributes

### Create the /generator-myapp/generators/app/templates/gitattributes containing:

|  |
| --- |
| \* text=auto |

### Jshintrc

I use [jshint](https://www.npmjs.com/package/gulp-jshint) to enable warnings in the Javascript.

### Create the /generator-myapp/generators/app/templates/jshintrc containing:

|  |
| --- |
| {  "browser": true,  "node": true,  "esnext": true,  "bitwise": true,  "camelcase": true,  "curly": true,  "eqeqeq": true,  "immed": true,  "indent": 2,  "latedef": true,  "newcap": true,  "noarg": true,  "quotmark": "single",  "undef": true,  "unused": true,  "strict": true,  "angular": true  } |

### Index.html

I want to define a single page for my scaffolding. I use [**EJS-styled**](http://www.embeddedjs.com/) placeholders (<% … %>) in this template file, which will be filled in by our Yeoman-generator at runtime.

On the other hand I have inserted placeholders in the index.html for injecting Bower dependencies using the [wiredep](https://www.npmjs.com/package/wiredep)-plugin. These placeholders have the following syntax:

<html>

<head>

*<!-- bower:css -->*

*<!-- endbower -->*

</head>

<body>

*<!-- bower:js -->*

*<!-- endbower -->*

</body>

</html>

### Create the /generator-myapp/generators/app/templates/index.html containing:

|  |
| --- |
| <!doctype html>  <html<% if (includeModernizr) { %> class="no-js"<% } %> lang="">  <head>  <meta charset="utf-8">  <meta name="description" content="">  <meta name="viewport" content="width=device-width, initial-scale=1">  <title><%= appname %></title>  <link rel="apple-touch-icon" href="apple-touch-icon.png">  <!-- Place favicon.ico in the root directory -->  <!-- build:css styles/vendor.css -->  <!-- bower:css -->  <!-- endbower -->  <!-- endbuild -->  <!-- build:css styles/main.css -->  <link rel="stylesheet" href="styles/main.css">  <!-- endbuild -->  <% if (includeModernizr) { %>  <!-- build:js scripts/vendor/modernizr.js -->  <script src="/bower\_components/modernizr/modernizr.js"></script>  <!-- endbuild --><% } %>  </head>  <body>  <!--[if lt IE 10]>  <p class="browserupgrade">You are using an <strong>outdated</strong> browser. Please <a href="http://browsehappy.com/">upgrade your browser</a> to improve your experience.</p>  <![endif]-->  <% if (includeBootstrap) { %>  <div class="container">  <div class="header">  <ul class="nav nav-pills pull-right">  <li class="active"><a href="#">Home</a></li>  <li><a href="#">About</a></li>  <li><a href="#">Contact</a></li>  </ul>  <h3 class="text-muted"><%= appname %></h3>  </div>  <div class="jumbotron">  <h1>Hello!</h1>  <p class="lead">Gulp scaffolding app.</p>  <p><a class="btn btn-lg btn-success" href="#">Button!</a></p>  </div>  <div class="row marketing">  <div class="col-lg-6">  <h4>HTML5 Boilerplate</h4>  <p>HTML5 Boilerplate is a professional front-end template for building fast, robust, and adaptable web apps or sites.</p>  <% if (includeAngular) { %>  <h4>AngularJS</h4>  <p>You have AngujarJS</p>  <% } %>  <% if (includeJQuery) { %>  <h4>JQuery</h4>  <p>You have JQuery</p>  <% } %>  <% if (includeSass) { %>  <h4>Sass</h4>  <p>Sass is the most mature, stable, and powerful professional grade CSS extension language in the world.</p>  <% } %>  <h4>Bootstrap</h4>  <p>Sleek, intuitive, and powerful mobile first front-end framework for faster and easier web development.</p><% if (includeModernizr) { %>  <h4>Modernizr</h4>  <p>Modernizr is an open-source JavaScript library that helps you build the next generation of HTML5 and CSS3-powered websites.</p>  <% } %>  </div>  </div>  <div class="footer">  <p>Footer placeholder</p>  </div>  </div>  <% } else { %>  <div class="hero-unit">  <h1>Hello!</h1>  <p>You now have</p>  <ul>  <% if (includeAngular) { %>  <li>AngujarJS</li>  <% } %>    <% if (includeJQuery) { %>  <li>JQuery</li>  <% } %>  <li>HTML5 Boilerplate</li><% if (includeSass) { %>  <li>Sass</li><% } %><% if (includeModernizr) { %>  <li>Modernizr</li><% } %>  </ul>  </div>  <% } %>  <!-- Google Analytics: change UA-XXXXX-X to be your site's ID. -->  <script>  (function(b,o,i,l,e,r){b.GoogleAnalyticsObject=l;b[l]||(b[l]=  function(){(b[l].q=b[l].q||[]).push(arguments)});b[l].l=+new Date;  e=o.createElement(i);r=o.getElementsByTagName(i)[0];  e.src='https://www.google-analytics.com/analytics.js';  r.parentNode.insertBefore(e,r)}(window,document,'script','ga'));  ga('create','UA-XXXXX-X');ga('send','pageview');  </script>  <!-- build:js scripts/vendor.js -->  <!-- bower:js -->  <!-- endbower -->  <!-- endbuild -->  </body>  </html> |

### Main.css

I need to define default styling.

Create the /generator-myapp/generators/app/templates/main.css containing:

|  |
| --- |
| <% if (includeBootstrap) { %>.browserupgrade {  margin: 0.2em 0;  background: #ccc;  color: #000;  padding: 0.2em 0;  }  /\* Space out content a bit \*/  body {  padding-top: 20px;  padding-bottom: 20px;  }  /\* Everything but the jumbotron gets side spacing for mobile first views \*/  .header,  .marketing,  .footer {  padding-left: 15px;  padding-right: 15px;  }  /\* Custom page header \*/  .header {  border-bottom: 1px solid #e5e5e5;  }  /\* Make the masthead heading the same height as the navigation \*/  .header h3 {  margin-top: 0;  margin-bottom: 0;  line-height: 40px;  padding-bottom: 19px;  }  /\* Custom page footer \*/  .footer {  padding-top: 19px;  color: #777;  border-top: 1px solid #e5e5e5;  }  .container-narrow > hr {  margin: 30px 0;  }  /\* Main marketing message and sign up button \*/  .jumbotron {  text-align: center;  border-bottom: 1px solid #e5e5e5;  }  .jumbotron .btn {  font-size: 21px;  padding: 14px 24px;  }  /\* Supporting marketing content \*/  .marketing {  margin: 40px 0;  }  .marketing p + h4 {  margin-top: 28px;  }  /\* Responsive: Portrait tablets and up \*/  @media screen and (min-width: 768px) {  .container {  max-width: 730px;  }  /\* Remove the padding we set earlier \*/  .header,  .marketing,  .footer {  padding-left: 0;  padding-right: 0;  }  /\* Space out the masthead \*/  .header {  margin-bottom: 30px;  }  /\* Remove the bottom border on the jumbotron for visual effect \*/  .jumbotron {  border-bottom: 0;  }  }<% } else { %>body {  background: #fafafa;  font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;  color: #333;  }  .hero-unit {  margin: 50px auto 0 auto;  width: 300px;  font-size: 18px;  font-weight: 200;  line-height: 30px;  background-color: #eee;  border-radius: 6px;  padding: 60px;  }  .hero-unit h1 {  font-size: 60px;  line-height: 1;  letter-spacing: -1px;  }  .browserupgrade {  margin: 0.2em 0;  background: #ccc;  color: #000;  padding: 0.2em 0;  }<% } %> |

### Main.scss

### When I want to support SASS, I need a file containing scss for my styling.

### Create the /generator-myapp/generators/app/templates/main.scss containing:

|  |
| --- |
| <% if (includeBootstrap) { %>$icon-font-path: '../fonts/';  // bower:scss  // endbower  .browserupgrade {  margin: 0.2em 0;  background: #ccc;  color: #000;  padding: 0.2em 0;  }  /\* Space out content a bit \*/  body {  padding-top: 20px;  padding-bottom: 20px;  }  /\* Everything but the jumbotron gets side spacing for mobile first views \*/  .header,  .marketing,  .footer {  padding-left: 15px;  padding-right: 15px;  }  /\* Custom page header \*/  .header {  border-bottom: 1px solid #e5e5e5;  /\* Make the masthead heading the same height as the navigation \*/  h3 {  margin-top: 0;  margin-bottom: 0;  line-height: 40px;  padding-bottom: 19px;  }  }  /\* Custom page footer \*/  .footer {  padding-top: 19px;  color: #777;  border-top: 1px solid #e5e5e5;  }  .container-narrow > hr {  margin: 30px 0;  }  /\* Main marketing message and sign up button \*/  .jumbotron {  text-align: center;  border-bottom: 1px solid #e5e5e5;  .btn {  font-size: 21px;  padding: 14px 24px;  }  }  /\* Supporting marketing content \*/  .marketing {  margin: 40px 0;  p + h4 {  margin-top: 28px;  }  }  /\* Responsive: Portrait tablets and up \*/  @media screen and (min-width: 768px) {  .container {  max-width: 730px;  }  /\* Remove the padding we set earlier \*/  .header,  .marketing,  .footer {  padding-left: 0;  padding-right: 0;  }  /\* Space out the masthead \*/  .header {  margin-bottom: 30px;  }  /\* Remove the bottom border on the jumbotron for visual effect \*/  .jumbotron {  border-bottom: 0;  }  }<% } else { %>// bower:scss  // endbower  body {  background: #fafafa;  font-family: 'Helvetica Neue', Helvetica, Arial, sans-serif;  color: #333;  }  .hero-unit {  margin: 50px auto 0 auto;  width: 300px;  font-size: 18px;  font-weight: 200;  line-height: 30px;  background-color: #eee;  border-radius: 6px;  padding: 60px;  h1 {  font-size: 60px;  line-height: 1;  letter-spacing: -1px;  }  }  .browserupgrade {  margin: 0.2em 0;  background: #ccc;  color: #000;  padding: 0.2em 0;  }<% } %> |

### Main.js

### Create the /generator-myapp/generators/app/templates/main.js containing:

|  |
| --- |
| /\* jshint devel:true \*/  console.log('Hello!'); |

### Robot.txt

### Create the /generator-myapp/generators/app/templates/robot.txt containing:

|  |
| --- |
| # robotstxt.org/  User-agent: \*  Disallow: |

# Write the generator specific files

In this section I finally parse and copy the templates from ‘**/generator-myapp/generators/app/templates/**’ to the ‘outputfolder’. The outputfolder is the folder where the user want to generate his scaffolding.

These actions are performed in the the ‘**writing method’** in the **index.js** file.

The **index.js** should contain the following:

|  |
| --- |
| var MyappGenerator = yeoman.generators.Base.extend({  prompting:{  …………  },  **writing:** {  gulpfile: function () {  this.template('gulpfile.js');  },  packageJSON: function () {  this.template('\_package.json', 'package.json');  },  git: function () {  this.fs.copy(  this.templatePath('gitignore'),  this.destinationPath('.gitignore')  );  this.fs.copy(  this.templatePath('gitattributes'),  this.destinationPath('.gitattributes')  );  },  bower: function () {  var bower = {  name: this.\_.slugify(this.appname),  private: true,  dependencies: {}  };  if (this.includeBootstrap) {  var bs = 'bootstrap' + (this.includeSass ? '-sass' : '');  bower.dependencies[bs] = '~3.3.1';  }  if (this.includeModernizr) {  bower.dependencies.modernizr = '~2.8.1';  }    if (this.includeAngular) {  bower.dependencies.angular = '~1.3.15';  }    if (this.includeJQuery) {  bower.dependencies.jquery = '~2.1.1';  }  this.fs.copy(  this.templatePath('bowerrc'),  this.destinationPath('.bowerrc')  );  this.write('bower.json', JSON.stringify(bower, null, 2));  },  jshint: function () {  this.fs.copy(  this.templatePath('jshintrc'),  this.destinationPath('.jshintrc')  );  },    h5bp: function () {  this.fs.copy(  this.templatePath('favicon.ico'),  this.destinationPath('app/favicon.ico')  );  this.fs.copy(  this.templatePath('apple-touch-icon.png'),  this.destinationPath('app/apple-touch-icon.png')  );  this.fs.copy(  this.templatePath('robots.txt'),  this.destinationPath('app/robots.txt')  );  },  mainStylesheet: function () {  var css = 'main';  if (this.includeSass) {  css += '.scss';  } else {  css += '.css';  }  this.copy(css, 'app/styles/' + css);  },  writeIndex: function () {  this.indexFile = this.src.read('index.html');  this.indexFile = this.engine(this.indexFile, this);  // wire Bootstrap plugins  if (this.includeBootstrap) {  var bs = '/bower\_components/';  if (this.includeSass) {  bs += 'bootstrap-sass/assets/javascripts/bootstrap/';  } else {  bs += 'bootstrap/js/';  }  this.indexFile = this.appendScripts(this.indexFile, 'scripts/plugins.js', [  bs + 'affix.js',  bs + 'alert.js',  bs + 'dropdown.js',  bs + 'tooltip.js',  bs + 'modal.js',  bs + 'transition.js',  bs + 'button.js',  bs + 'popover.js',  bs + 'carousel.js',  bs + 'scrollspy.js',  bs + 'collapse.js',  bs + 'tab.js'  ]);  }  this.indexFile = this.appendFiles({  html: this.indexFile,  fileType: 'js',  optimizedPath: 'scripts/main.js',  sourceFileList: ['scripts/main.js']  });  this.write('app/index.html', this.indexFile);  },  app: function () {  this.copy('main.js', 'app/scripts/main.js');  }  }, |

This ‘writing’ method supports the following:

* gulpfile: Parse the gulpfile.js and copy it to the outputfolder
* packageJSON : Copy package.json to the outputfolder
* bower: Add dependencies to the bower.json and copy it to the outputfolder
* mainStylesheet : Copy the desired stylesheet
* writeIndex: Wire the bootstrapplugins and the main.js at the end of the index.html.
* app: copy the main.js file

# Install npm and bower

Once you've run your generators, you'll often want to run [npm](https://www.npmjs.com/) and [Bower](http://bower.io/) to install any additional dependencies your generators require. In this section I will use the entered ‘options’ (see section: Options end Arguments) via the commandline.

In our generator (**index.js**) the installation of the dependencies should be defined in the **install** method:

|  |
| --- |
| var MyappGenerator = yeoman.generators.Base.extend({  **install: function () {**  var howToInstall =  '\nAfter running ' +  chalk.yellow.bold('npm install & bower install') +  ', inject your' +  '\nfront end dependencies by running ' +  chalk.yellow.bold('gulp wiredep') +  '.';  if (this.options['skip-install']) {  this.log(howToInstall);  return;  }  this.installDependencies({  skipMessage: this.options['skip-install-message'],  skipInstall: this.options['skip-install']  });  this.on('end', function () {  var bowerJson = this.dest.readJSON('bower.json');  // wire Bower packages to .html  wiredep({  bowerJson: bowerJson,  directory: 'bower\_components',  exclude: ['bootstrap-sass', 'bootstrap.js'],  ignorePath: /^(\.\.\/)\*\.\./,  src: 'app/index.html'  });  if (this.includeSass) {  // wire Bower packages to .scss  wiredep({  bowerJson: bowerJson,  directory: 'bower\_components',  ignorePath: /^(\.\.\/)+/,  src: 'app/styles/\*.scss'  });  }  // ideally we should use composeWith, but we're invoking it here  // because generator-mocha is changing the working directory  // https://github.com/yeoman/generator-mocha/issues/28  this.invoke(this.options['test-framework'], {  options: {  'skip-message': this.options['skip-install-message'],  'skip-install': this.options['skip-install']  }  });  }.bind(this));  } |

This install method supports the following:

* Call installDependencies() to run both npm and bower
* After the installation, I use the ‘end-queue’, to wire the Bower packages in the index.html en main.scss.
* And last but not least I install the desired test-framework. The default is the mocha-testframework.

# Scaffold your app

After all these work, run this command in your generators root directory (the root folder is where you can find the: .yo-rc.json ).

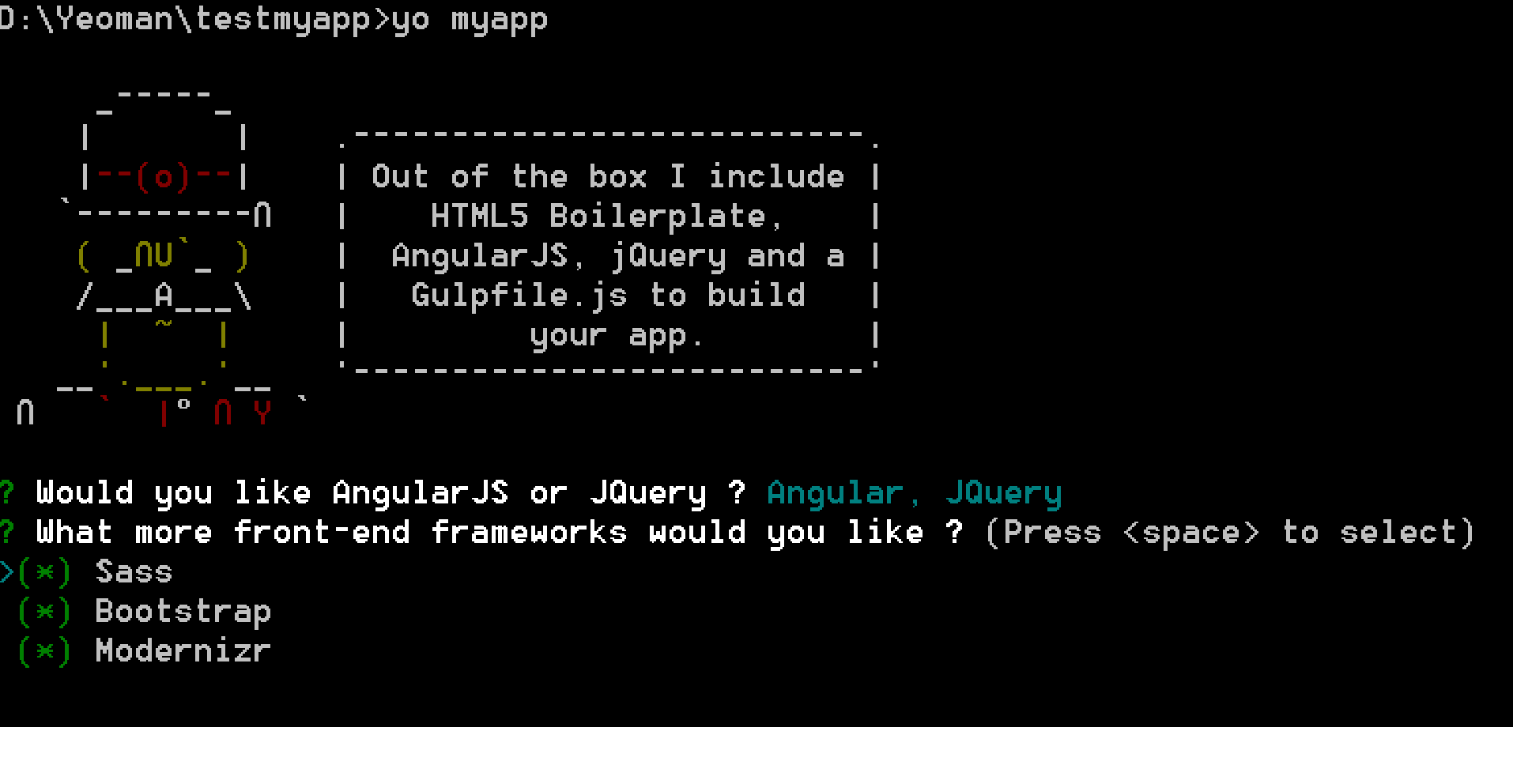
**npm link**

Make a new folder on your filesystem and scaffold your very own web app:

**mkdir testmyapp**

**cd testmyapp**

**yo myapp**

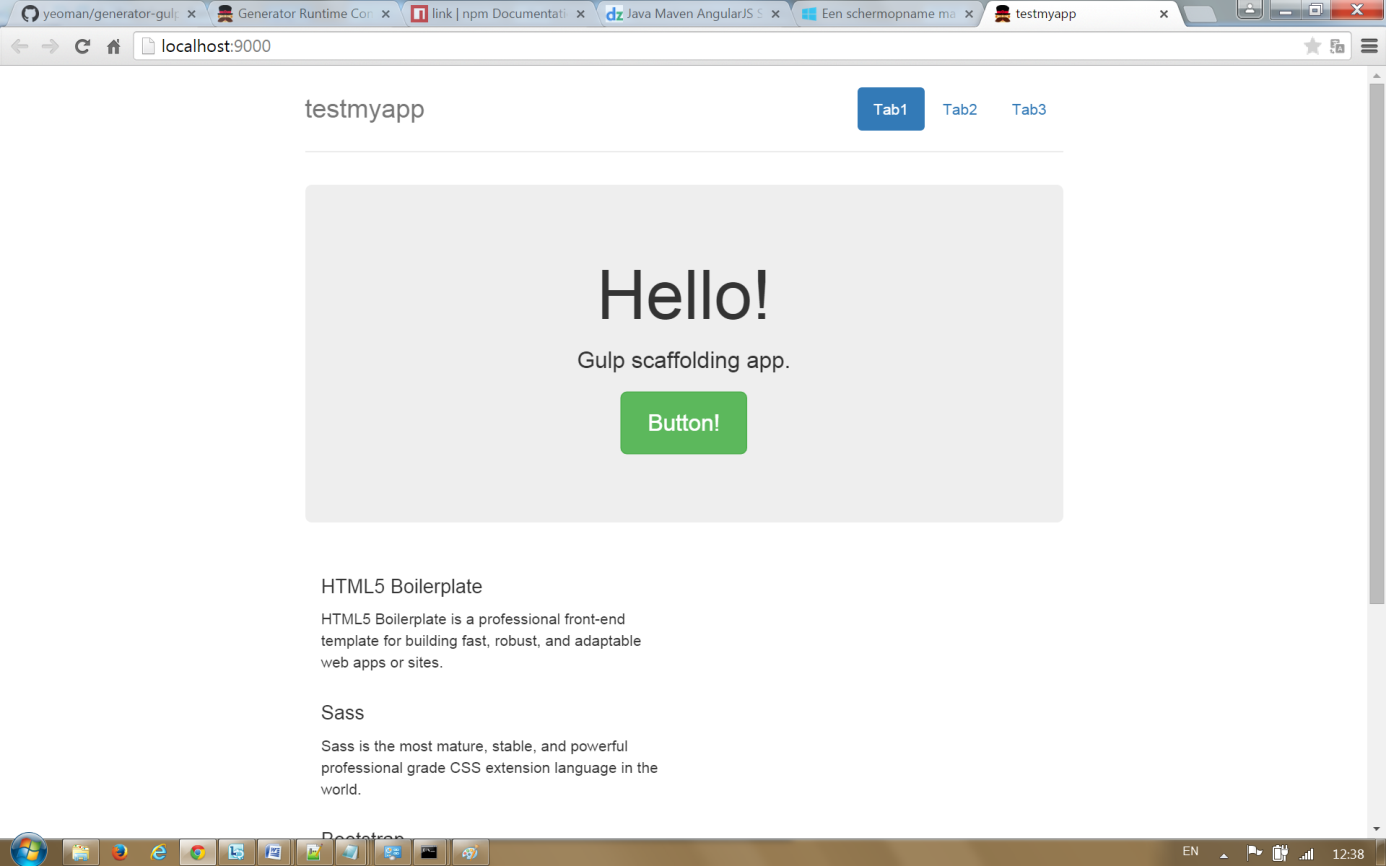


A new scaffolding will be generated in the outputfolder: testmyapp and the dependencies are installed.

To start developing, run:

**gulp serve**

This will fire up a local web server, open **http://localhost:9000** in your default browser and watch files for changes, reloading the browser automatically via [LiveReload](https://github.com/intesso/connect-livereload).

****

To make a production-ready build of the app, run:

**gulp**

# Unittest

To test your app, run:

**gulp serve:test**

# Conclusion

In this article, we covered a lot of the common features but there are still [more interesting options](http://yeoman.github.io/generator/) to check out. As you can probably tell, there is a bit of boilerplate required when building a generator, but that is kind of the point, you get it all done once and then you're able to use it throughout all your applications.

Yeoman is a great tool designed for front-end web developers. It helps you start new projects on the right foot and is a very powerful addition to every front-end developer’s arsenal.

**Related topics**

* [generator list](http://yeoman.io/generators/)
* [API documentation](http://yeoman.github.io/generator/)
* [tuts+ tutorial](http://code.tutsplus.com/tutorials/build-your-own-yeoman-generator--cms-20040)
* [create-and-publish-a-yeoman-generator](http://techiedreams.com/create-and-publish-a-yeoman-generator/)