Singular

Singular is a Bootstrap Admin theme powered by AngularJS.
This document describe the structure and use of the components included with the theme.

> Before requesting support, please read carefully and look for information on the topic that you think is related to the issue. There are big chances you could find the answer here and it would save you a lot of time!

Contact Support > (http://themeforest.net/user/geedmo#contact)

Getting started

Since AngularJs applications load the html via http requests (html views, partials, json data, etc), to get started with this theme you need a webserver. You can use the built-in server by using the gulp command or download XAMPP (https://www.apachefriends.org/index.html) to load the app from a local webserver. Any other options like nodejs, iis, etc is also allowed. Make sure to choose a server that you fill comfortable with the configuration system.

Quick Tips

- Find an existing asset and modify it to learn how it works.
- Explore the sources for ideas and sample code.
- Use Firebug or Chrome Developer Tools to find bugs on your website.
- Getting an error message? Someone might have seen it too, try a quick Google search for a fix.

Structure

Project folders organization

```
`-- app/
|   `-- css/
|   `-- img/
|   `-- js/
|   `-- langs/
|   `-- pages/
|   `-- views/
|   `-- vendor/
`-- master/
|   `-- jade/
|   |   `-- views/
|   |   |   `-- partials/
|   `-- js/
|   |   `-- modules/
|   |   |   `-- forms/
|   |   |   `-- tables/
|   |   |   `-- etcs ..
|   `-- less/
|   |   `-- app/
|   |   `-- bootstrap/
|   `-- gulpfile . js
|   `-- package . json
|   `-- bower . json
`-- server/
|   `-- * . json
`-- index . html
```

Folders

app/ folder

This folder contains the compiled files. This files are ready to deploy on your server.

- pages/ This folder contains the compiled html files for the single pages (out of the app).
- views/ This folder contains the compiled html files for the views and partials used for the app.
    - partials/ Contains compiles html files that are used in views
- langs/ This folder contains the json files use for translation.
- css/ Contains the static css files generated for the app
- js/ Contains the concatenated javascript source
- img/ Contains the theme images

master/ folder

This folder contains the source files that you can (optionally) compile to get the static version (html, css, js).

- jade/ This folder contains JADE files that are compiled into static html via gulp tasks. -
- less/ This folder contains the LESS files for the core styles and bootstrap styles.
    - app/ contains the LESS source with the app components
    - bootstrap/ contains the LESS source files for Bootstrap
- js/ Here you will find pure JS files. All this files are concatenated into the file app.js.
    - modules/ contains all controllers, directives, etc previously separated in folder using the modular pattern.
    - custom/ here you will put your own custom controllers, directives, etc

vendor/ folder

This folder contains the vendor files used to include plugins and other components. This folder is handled via bower so you can remove or upgrade the vendor components using such tool.

server/ folder

This folder contains server side files (only json included in the package) Curretnly this folder contains only files used for the charts, calendar and sidebar components.

Usage

Customize

To add your custom code you can follow this guideline

Static files

Go to folder app/ to find all html, js and css files.
Create a file custom.js and put it into the folder app/js, then linkit from the index.html file.
Create a file custom.css and put it into the folder app/js, then linkit from the index.html file after the app.css.

Source files

Source files are located under the folder master/ and requires gulp to compile them into static files.
To add custom js, create your files into the folder master/js/custom. All this files will be concatenated after all files into the folder modules.
Your custom LESS should be @imported at the bottom of the file app.less.
Finally, jade files can be added directly into the folder master/jade and they will be compiled automatically to its html version into the folder app/views.

Servers

Here are some resources you might find useful according to the server you want to use

- ExpressJS
  http://briantford.com/blog/angular-express (http://briantford.com/blog/angular-express)
- Visual Studio
  http://stackoverflow.com/questions/19516829/allow-loading-of-json-files-in-visual-studio-express-2013-for-web
  (http://stackoverflow.com/questions/19516829/allow-loading-of-json-files-in-visual-studio-express-2013-for-web)
- Django
  http://django-angular.readthedocs.org/en/latest/integration.html (http://django-angular.readthedocs.org/en/latest/integration.html)
- Ruby on Rails
  http://dillonbuchanan.com/programming/ruby-on-rails-angularjs-resources/ (http://dillonbuchanan.com/programming/ruby-on-rails-angularjs-resources/)

Build

Tools

> This is only required if you want to work with JADE, LESS and modular JS. Otherwise, you don't need to deal with those language to work with the theme.

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications.

Gulp is a task manager, you can define different tasks to run certain commands. Those commands does the compilation job for JADE and LESS, and concats the JS files.

Bower is a depency manager, it works by fetching and installing packages from all over, taking care of hunting, finding, downloading, and saving the stuff you're looking for. ower keeps track of these packages in a manifest file, bower.json.

The package includes under the master/ folder the file gulpfile.js and package.json to install the required components to compile the source files.

The bower.json file is included under the root folder.

Installing tools

Follow this steps to start installing this tools

- To install node and npm, go to http://nodejs.org/ (http://nodejs.org/)
- Make sure both command node and npm are in the enviroment variable PATH
- Run npm install -g bower

Compiling sources

- Open a terminal, go to master/ folder and run the command npm install. This command will install gulp and all dependencies.
- Finally run gulp to start the task manager

If everything goes fine, you should see the messages in the terminal telling you that most the task are done ok.

- To install vendor dependencies, run bower install

Tasks:

- `gulp dev` will start the task manager with livereload so any change in the source will trigger a page reload. Make sure you have the LiveReload plugin for Chrome.
- `gulp dev:server` will to start the task manager with livereload and also serve the application directly from nodejs. Your localserver will be accessible at http://localhost:3000 (http://localhost:3000)
- `gulp vendor:base` will generate the base.js file with the required scripts to load the page. Requires bower install
- `gulp vendor:copy` will copy all files in file vendor.json to the app/vendor folder. Requires bower install

Note: You can use tasks defined in gulp directly from command line. For example, `gulp dev webserver` or just `gulp webserver`

Note2: All customization can be done by editing the content of the file gulpfile.js

Javascript

The main Javascript is divided in two main files that controls the app

base.js: contains the scripts to start the application. It is generated automatically from vendor folder and files included are defined in file gulpfile.js (variable vendorBaseScripts)

app.js: contains the modules used in the application (controllers, directive, etc). It's generated by concatenating the files in the master/js folder.

JADE

Jade is a terse language for writing HTML templates. This files are not served directly to browser, they are used only to generate static HTML files.
To serve this files you will need a server like ExpressJS (http://expressjs.com/guide/using-template-engines.html)

LESS

LESS files for the theme components and for Bootstrap compiles into the file app.css. This file contains the bootstrap styles at the beginning and the application custom styles after them so any custom styles overrides the default Bootstrap style.

There's also a file generated automatically called apprtl.css which is automatically generated with the same styles but inverted for RTL layouts.

Vendor

Vendor folder

Vendor script dependencies are managed by bower. Just run bower install in folder master/ and all dependencies will be installed.

Most of the time bower downloads many files that are not necessary. Because of that, there's a task `vendor:copy` that will copy all files required by the app from the bower_components folder to the app/vendor folder.
This files are configured in file vendor.json which contains the path for all necessary files required by the app components. Those files are usually required via the lazy load module but you can include fonts, svg, etc.

Vendor base

The vendor base script, base.js is generated by the task `vendor:base` . This task concats and compress all files that needs to be loaded when the app is requested for first time.
All files included here are defined in file gulpfile.js

Features

Layout

The following layout markup representation is in fact divided into views but this code will give a good perspective of the final organization after the app is rendered:

```
<html>
 <head>
  #metas and css
 </head>
 <body>
  <section class="app-container" data-ui-view>

    #start include from app.html

     <header ng-include="'app/views/partials/top-navbar.html'"> ...
      # top navbar markup

     <aside ng-include="'app/views/partials/sidebar.html'"> ..
      # sidebar markup

     <section>
      <div class="app" data-ui-view>
       # view markup
      </div>
     </section>

     <footer ng-include="'app/views/partials/footer.html'"> ...
      # footer markup

    #end include from app.html

  </section>

  #scripts

 </body>
</html>
```

Layout can be changed via the following classed applied to the body tag

.layout-fixed : Makes navbars become fixed while the user can scoll only content

.layout-boxed : Limits the width of the main wrapper element

.aside-collapsed : Condenses the sidebar showing only icons

.aside-toggle : used internally for mobiles to hide the sidebar off screen

Lazy Load

This app requires only the necessary scripts according to the view that is loaded saving tons unnecessary request to the server.

The lazy load is handled by a custom core function based on the plugin ocLazyLoad (https://github.com/ocombe/ocLazyLoad)

To configure the lazy scripts, you need to edit the constants `appDependencies` (constants.js)
Then edit the app configuration (config.js) where you will find the routes configuration and add or edit the params using the special function `requireDeps` which handles the scripts request order for the current route.

RTL

The Right To Left (RTL) version of the app is handled by the file `apprtl.css` . This file is generated using a tool called css-flip (https://github.com/twitter/css-flip) which inverts most the css properties to change the page orientation. To make use of this tool you can download the standalone version or just compile the source with gulp. There's a global property `$rootScope.isRTL` used to detect when the site is in RTL mode.

Routes

This app uses for routing the AngularUI Router (https://github.com/angular-ui/ui-router) with nested states making more simple to manage the routing system and load resource in cascade.

This routes are defined in the file config.js

Translation

The translation system uses the AngularUI Translate (https://github.com/angular-translate/angular-translate) module.
This modules simplifies the translation system by loading translate references from a JSON file and replacing the content where the reference has been used.

Examples

```
<h3 ui-translate="reference.NAME">Text that will be replaced</h3>

<h3>{{ 'reference.NAME' | translate }}</h3>

<a href="#" title="{{ 'reference.NAME' | translate }">Link</a>
```

The JSON files with translation refernces are located in the folder app/langs and it is named using the corresponding locale ID for a language. Learn More (https://docs.angularjs.org/guide/i18n)

Sidebar

The sidebar is created dynamically from a JSON file.
Such file is called `sidebar-items.json` and it's located under the folder server.

JSON properties format:

To create a single item

```
[
 {
 "text":     "Item text"  ,          // replaced by translate reference
 "sref":     "app.dashboard"  ,       // the state name of the target route
 "icon":     "icon-classname"  ,      // the icon full classname
 "translate": "sidebar.ITEM"       // the translation reference
 }...
]
```

To create a heading or separator

```
[
 {
 "text":     "Item text"  ,          // replaced by translate reference
 "translate": "sidebar.ITEM"  ,       // the translation reference
 "type": "heading or separator"     // only heading needs the above props
 }...
]
```

To make the sidebar works properly it requires the following

- SidebarController
- SidebarDirective
- SidebarMenuService

> When the json file has a wrong format an alert message will be displayed.

Markdown Docs

This documentation is loaded from a Markdown source using Flatdoc (http://ricostacruz.com/flatdoc/) plugin. The menu and the content is generated automatically from the .md file and styled directly from custom css.

Via the `flatdoc` directive you can use it like this

```
<flatdoc src="path/to/readme.md">
  <!-- generated by directive -->
  <div role='flatdoc'>
    <div role='flatdoc-menu' ui-scrollfix></div>
    <div role='flatdoc-content'></div>
  </div>
  <!-- end directive -->
</flatdoc>
```

Themes

Themes are handled via background classes (bg-*) you can watch them here
Those classes will style all elements inside allowing to change the app style in the Angular way.

How it works?

The SettingsController defines an array of element to apply the `.bg-*` classes combination.
This array is used to generate the color combination for each theme using styles defined in the `settings.less` file. Once a color combination is clicked, Angular updates the model settings with the current combination.
This combination is then saved within the global `$rootScope.app` configuration to be restored later.

Directives

Bootstrap UI

This item include all directives from Angular BootstrapUI (http://github.com/api).

Custom

This item include the following directives.

`[href]`

Disables empty links default behavior

`[animate-enabled]`

Enable or disables ngAnimate for the element with the directive

`[reset-key]`

Removes a key from the browser storage via element click

`[flatdoc]`

Creates the flatdoc markup and initializes the plugin

`[toggle-fullscreen]`

Toggle the fullscreen mode on/off

`[masked]`

Initializes the masked inputs

`[sidebar]`

Wraps the sidebar and handles collapsed state

`[climacon]`

Include any animated weather icon from Climacon

`[sparkline]`

SparkLines Mini Charts

`[check-all]`

Tables check all checkbox

`[toggle-state]`

Toggle a classname from the body tag. Useful to change a state that affects globally the entire layout or more than one item.
Elements must have [toggle-state="CLASS-NAME-TO-TOGGLE"]. Use [no-persist] to avoid saving the sate in browser storage.

`[bootstrap-slider]`

Initializes the jQuery UI slider controls

`[vector-map]`

Init jQuery Vector Map plugin

[flot]

Initializes the Flot chart plugin and handles data refresh

[wysiwyg]

Initializes the Wysiwyg editor

[portlet]

Drag and drop any panel based on jQueryUI portlets

[scrollable]

Make a content box scrollable

Constants

appDependencies

Defines the scripts path that will be used with the lazy load manager.

Format:

```
// Put here all jQuery script (and not angular js)
scripts: {
 'friendly-name' : [ 'path/to/plugin.js' , 'path/to/plugin.css' , '...' ],
   ...
 }
// Put here all angular js modules that needs to be instantiated
modules: {
   {
   name: 'toaster' , files: [ 'path/to/module.js' , 'path/to/module.css' , '...' ]
   },
   ...
 }
```

Learn more by this constant by looking into the file config.js

appColors

Defines the brand colors used in the css accessible from JS

```
App . controller ( 'ExampleCtrl' , [ 'appColors' , function ( colors ) {
 console . log ( colors . primary );
 // prints #5d9cec
 }]);
```

This constant is used together with the service colors to provide access from the $scope to each color by its name

Example

```
<div sparkline data-bar-color="{{colorByName('primary')}}" ></div>
```

appMediaquery

Defines the media queries used in the css accessible from JS

```
App . controller ( 'ExampleCtrl' , [ 'appMediaquery' , function ( mq ) {
 console . log ( mq [ 'mobile' ] );
 // prints 480
 }]);
```

Credits

Angular (https://angularjs.org/)
Angular Docs (https://docs.angularjs.org/guide/)
ocLazyLoad (https://github.com/ocombe/ocLazyLoad)
uiRouter (https://github.com/angular-ui/ui-router)
uiTranslate (https://github.com/angular-translate/angular-translate)
uiBootstrap (http://angular-ui.github.io/bootstrap/)
Toaster (https://github.com/jirikavi/AngularJS-Toaster)
Angular Loading Bar (http://chieffancypants.github.io/angular-loading-bar/)
Bootstrap (http://getbootstrap.com/)
jQuery (http://jquery.com/)
Animate.css (http://daneden.github.io/animate.css/)
Chosen (https://github.com/harvesthq/chosen/)
BS Datetime Picker (http://tarruda.github.io/bootstrap-datetimepicker/)
FlotCharts (http://www.flotcharts.org/)
Angular Gmap (https://github.com/dylanfprice/angular-gm-bower)
Angular Knob (https://github.com/yunlzheng/angular-knob)
Angular UI Utils (https://github.com/angular-ui/ui-utils)
Animated Climacons (https://github.com/noahblon/animated-climacons)
AngularUI Calendar (https://github.com/angular-ui/ui-calendar/)
Bootstrap Wysiwyg (https://github.com/mindmup/bootstrap-wysiwyg)
Marked (https://github.com/chjj/marked)
Modernizr (http://modernizr.com/)
MomentJs (http://momentjs.com/)
Bootstrap Slider (http://www.eyecon.ro/bootstrap-slider)
Sparkline (http://omnipotent.net/jquery.sparkline/#s-about)
slimSCroll (http://rocha.la/jQuery-slimScroll)
FullCalendar (http://arshaw.com/fullcalendar/docs/)
InputMask (https://github.com/RobinHerbots/jquery.inputmask)
jVectorMap (http://jvectormap.com/)
FlatDoc (https://github.com/rstacruz/flatdoc)
jQueryUI (http://jqueryui.com/sortable/)
ngTable (https://github.com/esvit/ng-table)
ScreenFull (https://github.com/sindresorhus/screenfull.js)

Icons

Feather Icons (http://colebemis.com/feather/)
Font Awesome (http://fortawesome.github.io/Font-Awesome/)
Climacons (http://adamwhitcroft.com/climacons/)
Weather Icons (http://erikflowers.github.io/weather-icons/)

Font

Open Sans (https://github.com/FontFaceKit/open-sans)

Demo images

uiFaces (http://uiFaces.com/)
Unsplash (http://unsplash.com)