FORGE UI

Apps

* components - controller
* config -
  + route,
  + const variables - enum
  + services – communicate with API using $http service (ajax)
* data – json data (temp, sample data)
* views – name should match with component, except \_ (underscore), treat as sharing, no

**index.html** – contain all the js lib reference, if you do minify js, make sure that “app-const.js”, “forge-app.js” and “app-route.js” are first (in this order) everything else are dependency injection, so really doesn’t matter.

whenever you add a new js file, \*\*\* you must add to the index.html page.

AngularJs loading sequence <https://docs.angularjs.org/guide/module>

* **app-const.js** – currently only used const\_auth.undefined and const\_request (this was just an idea, it’s not part of angularjs)
* **forge-app.js** – Modules can list other modules as their dependencies.
* **app-route.js** - A module is a collection of configuration and run blocks which get applied to the application during the bootstrap process. In its simplest form the module consists of a collection of two kinds of blocks:
  + **Configuration** blocks - get executed during the provider registrations and configuration phase. Only providers and constants can be injected into configuration blocks. This is to prevent accidental instantiation of services before they have been fully configured.
    - using **ui.router** <https://ui-router.github.io/ng1/tutorial/hellogalaxy>
  + **Run** blocks - get executed after the injector is created and are used to kickstart the application. Only instances and constants can be injected into run blocks. This is to prevent further system configuration during application run time. <https://docs.angularjs.org/api/ngRoute/service/$route>
    - $stateChangeStart – before go to another page
    - $stateChangeSuccess – after successful load from page
  + **Constant –** Register a constant service with the [$injector](https://docs.angularjs.org/api/auto/service/$injector), such as a string, a number, an array, an object or a function. Like the [value](https://docs.angularjs.org/api/auto/service/$provide#value), it is not possible to inject other services into a constant. But unlike [value](https://docs.angularjs.org/api/auto/service/$provide#value), a constant can be injected into a module configuration function (see [angular.Module](https://docs.angularjs.org/api/ng/type/angular.Module#config)) and it cannot be overridden by an AngularJS [decorator](https://docs.angularjs.org/api/auto/service/$provide#decorator).
* **factory-service-authentication.js** - Register a service factory, which will be called to return the service instance. This is short for registering a service where its provider consists of only a $get property, which is the given service factory function. You should use [$provide.factory(getFn)](https://docs.angularjs.org/api/auto/service/$provide#factory) if you do not need to configure your service in a provider. <https://docs.angularjs.org/api/auto/service/$provide>
  + **authentication**: fn – login taking username and password
    - **$cookies.get(‘profile’)** – user information and token
    - **$rootScope.profile –** currently use in app-route
    - **sessionStorage -** <https://www.w3schools.com/html/html5_webstorage.asp>
      * \_contentManagement – contain management information
  + **contentManagement:** fn – get content management api
* **sign-in-component.js** – sign in page, which calls fa

**VIEWS –** contain all html file

* **\_dashboard.html** – has no component, it’s a layout page
  + ui-view=”left-menu”, point to left-menu-component
  + heirarchy-component currently static page
  + ui-view=”content-manager”, point to content-management-component