# Overview

```

src/

|- app/

| |- app.constants.json

| |- app.controller.js

| |- app.module.js

| |- app.run.js

| |- app.spec.js

```

The `src/app` directory contains all code specific to this application. Apart

from `app.\*.js` and its accompanying tests (discussed below), this directory is

filled with subdirectories corresponding to high-level sections of the

application, often corresponding to top-level routes. Each directory can have as

many subdirectories as it needs, and the build system will understand what to

do. For example, a top-level route might be `productManagement`, which would be a directory

within the `src/app` directory that conceptually corresponds to the top-level

route `/products`, though this is in no way enforced. `productManagement` may then have

subdirectories for `inventory`, `pricing`, `product`, etc. The `product` submodule may

then define a route of `/products/:id`, ad infinitum.

## `app.constants.json`

This small, yet powerful file host a JSON object of key value pairs to be used

throughout the application. From this object, the build process will generate

a file containing AngularJS constants [here](https://github.com/ordercloud-api/angular-seller/blob/development/gulp.config.js#L73), which are used to [connect to your OrderCloud organizations](LINK TO CONNECTING TO YOUR SELLER ORGANIZATION GUIDE).

## `app.controller.js`

This is the application's main controller. `AppCtrl` is a good place for logic

not specific to the template or route, such as menu logic or page title wiring. This controller also allows for numerous Angular services such as `$state` and `$ocMedia` and service methods such as `ocIsTouchDevice` and `stateLoading` to be globally available throughout the application's various templates. `AppCtrl` is not declared in a state provider like the application's component controllers. Instead, it is declared directly in `index.html` with `ng-controller="AppCtrl as application"`.

```js

angular.module('orderCloud')

.controller('AppCtrl', AppController)

;

function AppController($state, $ocMedia, LoginService, appname, ocStateLoading, ocIsTouchDevice, ocRoles) {

var vm = this;

vm.name = appname;

vm.$state = $state;

vm.$ocMedia = $ocMedia;

vm.isTouchDevice = ocIsTouchDevice;

vm.stateLoading = ocStateLoading.Watch;

vm.logout = LoginService.Logout;

vm.userIsAuthorized = ocRoles.UserIsAuthorized;

}

```

## `app.module.js`

This is our main app file. It kickstarts the whole process by

requiring all the modules that we need.

By default, the OrderCloud AngularJS Seed includes a few useful modules written

by the AngularJS and Angular-UI teams. We also include the `orderCloud.sdk` module for connecting to the OrderCloud API. Lastly, some helpful third party modules are included as well, such as `toastr` and `angular-busy`.

All components within the application are tied directly to the `orderCloud` module, so they do not need to be included here.

```js

angular.module('orderCloud', [

'ngSanitize',

'ngAnimate',

'ngMessages',

'ngTouch',

'ui.tree',

'ui.router',

'ui.select',

'ui.bootstrap',

'ui.select',

'LocalForageModule',

'toastr',

'angular-busy',

'jcs-autoValidate',

'treeControl',

'hl.sticky',

'angularPayments',

'ordercloud-angular-sdk'

]

);

```

## `app.run.js`

Use the main applications run method to execute any code after services

have been instantiated. By default, we initialize `ocStateLoading`, validation error messages (using `angular-auto-validate`), and validation styling.

```js

angular.module('orderCloud')

.run(AppRun)

;

function AppRun(ocStateLoading, ocRefreshToken, defaultErrorMessageResolver, ocErrorMessages, validator) {

ocStateLoading.Init();

defaultErrorMessageResolver.getErrorMessages().then(function (errorMessages) {

angular.extend(errorMessages, ocErrorMessages);

});

validator.setValidElementStyling(false);

}

```

## `app.spec.js`

One of the design philosophies of `angular-seller` is that tests should exist

alongside the code they test and that the build system should be smart enough to

know the difference and react accordingly. As such, the unit test for `app.\*.js`

is `app.spec.js`.