## **Introduction**

Components are the key to a reusable approach when implementing AngularJS OrderCloud.io solutions. Angular offers the opportunity to create robust applications with many, small and self contained modules. Each module, or component, designed to be functional when included in the OrderCloud.io seed, encapsulates one responsibility enabled by the OrderCloud.io API.

### **Strategy**

Components are meant to be reusable across many applications. They can represent a single API resource, or a combination of several resources to create a common workflow. We know developers will want to take these components and make them their own, which is why we lean towards very basic UI, and validation that almost always mirrors that of the API. Developers can and should take these components and customize them to fit the needs of their own application.

### **Implementation**

To include a component in your application, simply drop the entire component folder into the src/app/ directory of the OrderCloud.io [Angular Seed](https://github.com/ordercloud-api/angular-seed). The seed has [logic to determine the presence of a component](https://github.com/ordercloud-api/angular-seed/blob/master/src/app/base/base.js#L59) and add a nav item to the base.left.tpl.html view.

Alternatively, you can create an [Accelerator](http://localhost:3000/docs/angularjs/accelerators) in the [Dev Center Dashboard](http://localhost:3000/docs/guides/getting-started/dashboard-and-api-console) as a starter application. Creating an Accelerator will configure your application and drop any desired components into your src/app/ directory for you. All you'll have to do is install the project dependencies and run the application!

### **Contributing**

OrderCloud.io's [Angular Component library](https://github.com/ordercloud-api/angular-components) is an open source project on [GitHub](https://github.com/). Contributions are welcome. Create a pull request for inclusion into the library. The project leaders will review all submissions and respond appropriately. We ask that every contributor follow the outstanding [Angular Style Guide](https://github.com/johnpapa/angular-styleguide/blob/master/a1/README.md), managed by [John Papa](https://github.com/johnpapa), to maintain consistent and clean code.

## **Component Structure**

Each component has roughly the [same directory and file structure](https://github.com/ordercloud-api/angular-components/tree/master/addresses). At the root we have the component's .js file and an .md file with some brief documentation.

There are three folders in each component:

* componentName
  + ./less
  + ./templates
  + ./tests

### **State Configuration**

The component library uses [ui-router](https://github.com/angular-ui/ui-router) (just like the Angular seed project) for defining the nested states & URLs relevant to each component. If you haven't already, familiarize yourself with the inner workings of ui-router.

We use resolves to load all of the relevant OrderCloud.io data before the controller of the state is initiated. Naming conventions are followed in order to make the code easier to understand for an outside developer; everywhere you see "Component" below is an example of component naming conventions.

function ComponentConfig( $stateProvider ) {  
 $stateProvider  
 .state( 'componentStateName', {  
 parent: 'base',  
 url: '/component-url',  
 templateUrl:'componentFolder/templates/componentView.tpl.html',  
 controller:'ComponentCtrl',  
 controllerAs: 'component',  
 data: {componentName: 'Component Display Name'},  
 resolve: {  
 ComponentResolve: function() {  
 var result = 'load before the controller inits'  
 return result;  
 }  
 }  
 })  
 .state('componentStateName.substate', {  
 //more than one state can be defined  
 //this is a nested state  
 });  
 }  
}

### **The Controller**

By way of [controllerAs](https://github.com/johnpapa/angular-styleguide/blob/master/a1/README.md#controlleras-view-syntax) construct, the "view model" or vm variable will be accessed in the HTML via the controllerAs value from the state configuration above. Below you can see we inject the resolved value and tie the pre-loaded data to the controller's view model.

function ComponentController(ComponentResolve) {  
 var vm = this;  
 vm.preloadedData = ComponentResolve;  
}

### **The Template**

Most components in the OrderCloud.io component library have more than one state involved. Below you see an example of a component's **root state** template (meaning that all of the other states in this component live under here).

<article id="COMPONENT\_componentName" ui-view class="container-fluid">  
 <div class="page-header">  
 <h3 ng-bind="component.preloadedData"></h3>  
 </div>  
</article>

Notice the ui-vew attribute on the component's root element. This is where all of the component sub-states will be loaded - replacing any child elements of the parent (the ".page-header" element in our case). However, the root element sticks around - making it the perfect CSS selector if you want to style an entire component in isolation.

## **Conclusion**

Now that you know the basics of how OrderCloud.io Components work in AngularJS, you're ready to begin developing your own custom workflows and save a considerable amount of time in the process.