## **Overview**

The Angular Seed's common directory contains a wide variety of services, directives, and views that were mostly developed by the OrderCloud.io team to simplify interaction with the data model. Some others facilitate features within OrderCloud.io Components like infinite scrolling.

## **Address**

There are two tools in the address directory. The first is an address form directive. This directive allows you to easily place a basic form anywhere it may be needed.

The address form directive accepts two attributes

* address - This is the object upon which the form inputs will be attached to which you may then store or use as desired.
* isbilling - Setting this to true will make the First Name and Last Name field a requirement.

This is the basic structure of the directive which can be placed anywhere in your HTML

<ordercloud-address-form isbilling="" address=""></ordercloud-address-form>

The second tool in this directory is an address filter. This will allow you to display the address object captured from your address form and display it to your user. It defaults to displaying only the streets but accepts an argument to display the full address as well. Shown below are both forms:

* ng-bind="yourAddressObject | address" - This will display just the street information for your address
* ng-bind="yourAddressObject | address:'full'" - This will display the full address

## **Auto-ID**

The Auto-ID directive adds a checkbox to an input field. If checked, this will auto-generate an ID. This gives users the flexibility to create their own ID or generate a random one.

To use this simply add ordercloud-auto-id as an attribute on an input field

## **Buyer-Select**

The buyer-select directive uses the OrderCloud.io SDK to change the active buyer organization

OrderCloud.BuyerID.Set(buyerid);

By default this directive is placed on the Base components left view (app/src/base/templates/base.left.tpl.html). It gives admin users the ability to quickly switch between buyer companies in their organization. Each buyer specific component will display data relevant to the active buyer.

This is the basic structure of the directive which can be placed anywhere in your HTML

<ordercloud-select-buyer><ordercloud-select-buyer>

## **Current-Order**

* CurrentOrder.Get() - returns the current order object
* CurrentOrder.GetID() - returns the ID of the current order
* CurrentOrder.Set(OrderID) - stores the order with the provided ID in [$localForage](https://github.com/localForage/localForage)
* CurrentOrder.Remove() - returns all of the line items on the current order
* CurrentOrder.GetLineItems() - returns all of the line items on the current order

## **Files**

This directive allows you to upload a file to a model quickly and easily. There are three attributes that this directive accepts:

* model - The model (any OrderCloud.io resource with XP) you would like your uploaded file saved on
* keyname - The key under the given model's XP where your file object will be stored.
* extension - The file types that you want to accept such as .jpg, .png, .csv, etc.
* invalid-extensions - File extensions that will be recognized as invalid on the front-end.

This is the basic structure of the directive which can be placed anywhere in your HTML:

<ordercloud-file-upload  
 model="view.model"  
 keyname="productImage"  
 label="Product Image"  
 extensions=".jpg, .png"  
 invalid-extension=".csv"><ordercloud-file-upload>

## **Helper-Factories**

There are two factories in this directory: Assignments and Paging

The assignments helpers are meant to simplify creating assignment UI's when users are selecting and/or deselecting items for assignment changes. Let's start with the Assignments factory which has six methods on it:

* getAssigned(assignmentsArray, ID\_name - returns all values in a given array with a given ID\_name.
* getSelected(listArray) - returns the ID's from an array of assignable objects where the value selected is set to true.
* getUnselected(listArray) - returns the ID's from an array of assignable objects where the value selected is set to false.
* getToAssign(listArray, assignmentsArray, ID\_name) - returns values that have not yet been selected or assigned.
* getToDelete(listArray, assignmentsArray, ID\_name) - returns values that are both not selected *and* assigned.
* saveAssignments(listArray, assignmentsArray, SaveFunc, DeleteFunc, ID\_Name) - This will save assignments by applying a delete function to the values returned from getToDelete and a Save function to the values returned from getToAssign.

The paging function facilitates infinite scroll capabilities when making assignment choices. It has two methods:

* setSelected(listArray, assignmentsArray, ID\_name)- This checks a list array to see if the item has been assigned, if it has it sets the select value to true.
* paging(listObject, ServiceName, AssignmentObjects, AssignmentFunc) -Given a listObject and a Service name this will generate a page of items. If provided an assignment Object and an assignment function then this will generate a page of those assignments.

## **Search**

This element directive was built to simplify simple queries when using ordercloud-infinite-scroll. It accepts the following three attributes:

* placeholder - The text that will be displayed in the search box before anything is typed
* servicename - The name of the OrderCloud.io service
* controlleras - The name of the controlleras in the current scope

This is the basic structure of the directive which can be placed anywhere in your HTML:

<ordercloud-search  
placeholder="Search Products"  
servicename="Products"  
controlleras="products"  
></ordercloud-search>

## **InfiniteScroll**

This attribute directive is used by most list views to allow for infinite scrolling. It works along side ordercloud-search to accurately page through search results. You can configure infinite scroll using the four attributes below:

* servicename - The name of the OrderCloud.io service
* controlleras - The name of the controlleras in the current scope
* idname - the ID name for the service you are using. This is an optional parameter if you would like to scroll through assignments.
* threshhold - the number in pixels from the bottom of the scroll container when pagination will occur. If left out this will default to 0 which will begin pagination when you get to the bottom of the scrollheight. Increasing this number will increase how long until pagination takes effect.

This directive must be placed on the scrollable element which encompasses the relevant repeated (ng-repeat) element:

<div  
placeholder="Search Products"  
servicename="Products"  
controlleras="products"  
<div ng-repeat="item in controllerName.model.Items"></div>  
<div>

## **LineItem Helpers Factory**

The LineItemHelpers factory has seven methods to allow you to easily interact with line items:

* SpecConvert() - Given a spec object from a line item, this will copy the spec options onto a new spec object.
* RemoveItem(Order,LineItem) - Returns a line item from an order
* UpdateQuantity(Order,LineItem) - This will update the quantity on a line item
* GetProductInfo(LineItems) - Returns all of the product info (excluding the product ID) from a line item
* CustomShipping(Order,LineItem) - Executing this function will open a modal that allows you to enter a custom shipping address for a line item
* UpdateShipping(Order, lineItem, AddressID) - updates the shipping address on a line item
* ListAll(OrderID) - returns all of the line items on an order

## **Media**

The $ocMedia service is used to evaluate whether a given media query is true or false given the current device's screen/window size. The media query will be re-evaluated on resize allowing you to register a watch.

The service also has pre-programmed support for media queries that match the following layout breakpoints:

|  |  |
| --- | --- |
| **Breakpoint** | **Screen Sizes** |
| sm | max-width:600px |
| gt-sm | min-width:600px |
| md | min-width:600px and max-width:960px |
| gt-md | min-width:960px and max-width:600px |
| lg | min-width:960px and max-width:1200px |
| gt-lg | min-width:1200px |

To use the service simply inject $ocMedia into your controller and pass in a media query. The function will return a boolean which you can then use to act upon. Here is an example using one of our pre-programmed options:

var isGreaterThanSmall = $ocMedia('gt-sm');

If the current user's window screen is greater than 600px then this will return true.

Here is an example using a custom option:

var isCustomBreakpoint = $ocMedia('min-width:1234px');

If the current user's window screen is greater than 1234px then this will return true.

## **TokenRefresh**

This factory has three handy tools that allow you to interact with the refresh tokens used for authorization:

* SetToken(token) - This will set the token in your cookies using [$localForage](https://github.com/localForage/localForage)
* GetToken() - This will get the token stored in your cookies
* Refresh(token) - This will allow you to refresh your token.