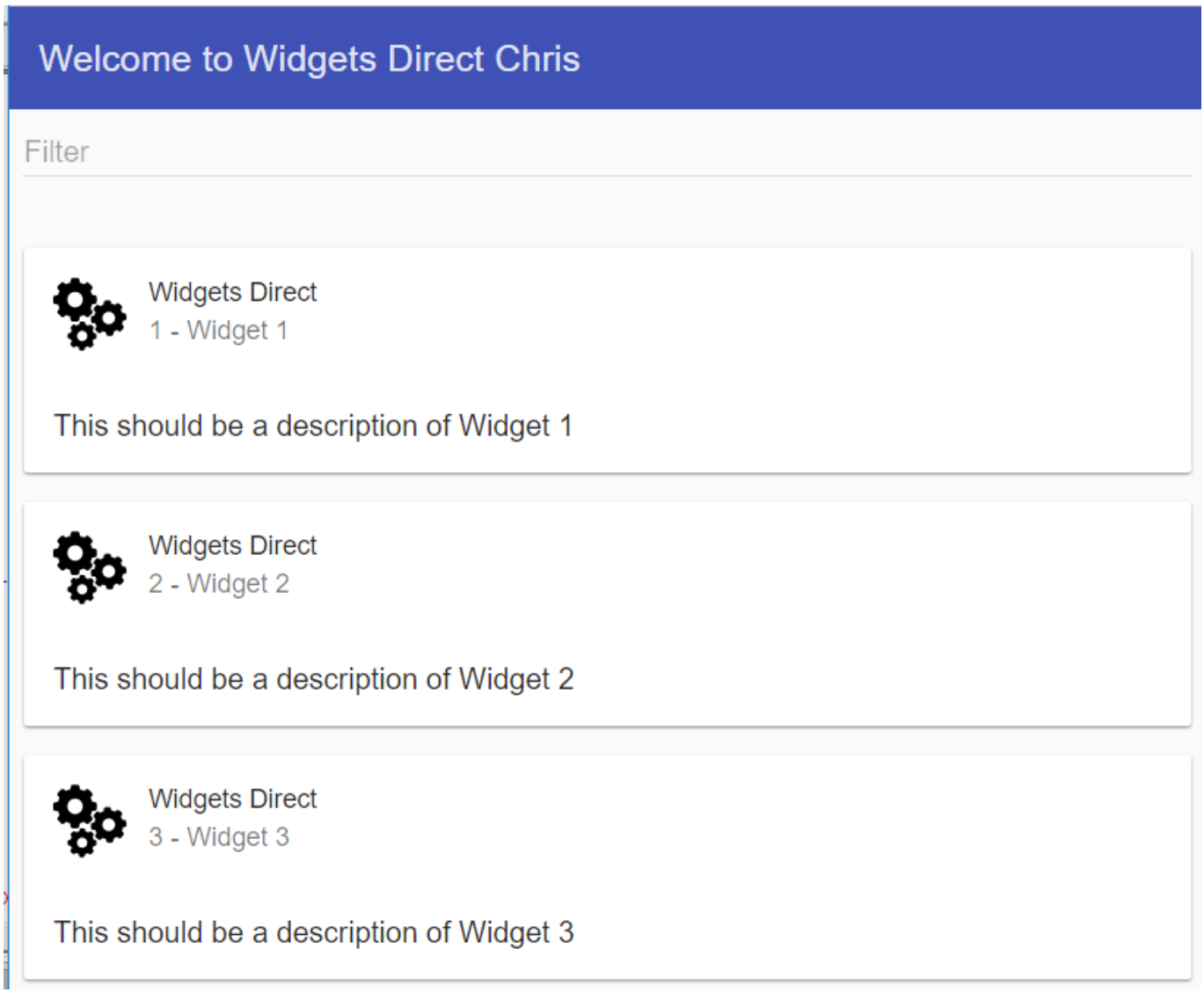
Article
Chris Stewart · Apr 21, 2017 2m read

Let's write an Angular 1.x app with a Caché REST backend - Part 5

At the end of our last lesson, we ended with our page displaying a nice (but garish) Angular Material Toolbar, and our Widget data displaying in a list of Material cards. Our page feels a bit static, and we already know that the large number of Widgets that we will be dealing with will not be especially usable on a static list. What can we do to help?

A filtering function would be very helpful, so let's add a text input to the top of the page. We will bind this to a variable in the $scope - widgetFilterText so that Angular has access to the value we type in. We will also wrap the whole page content in a div and set layout="column" to have the new text input and the existing cards arrange vertically on the screen

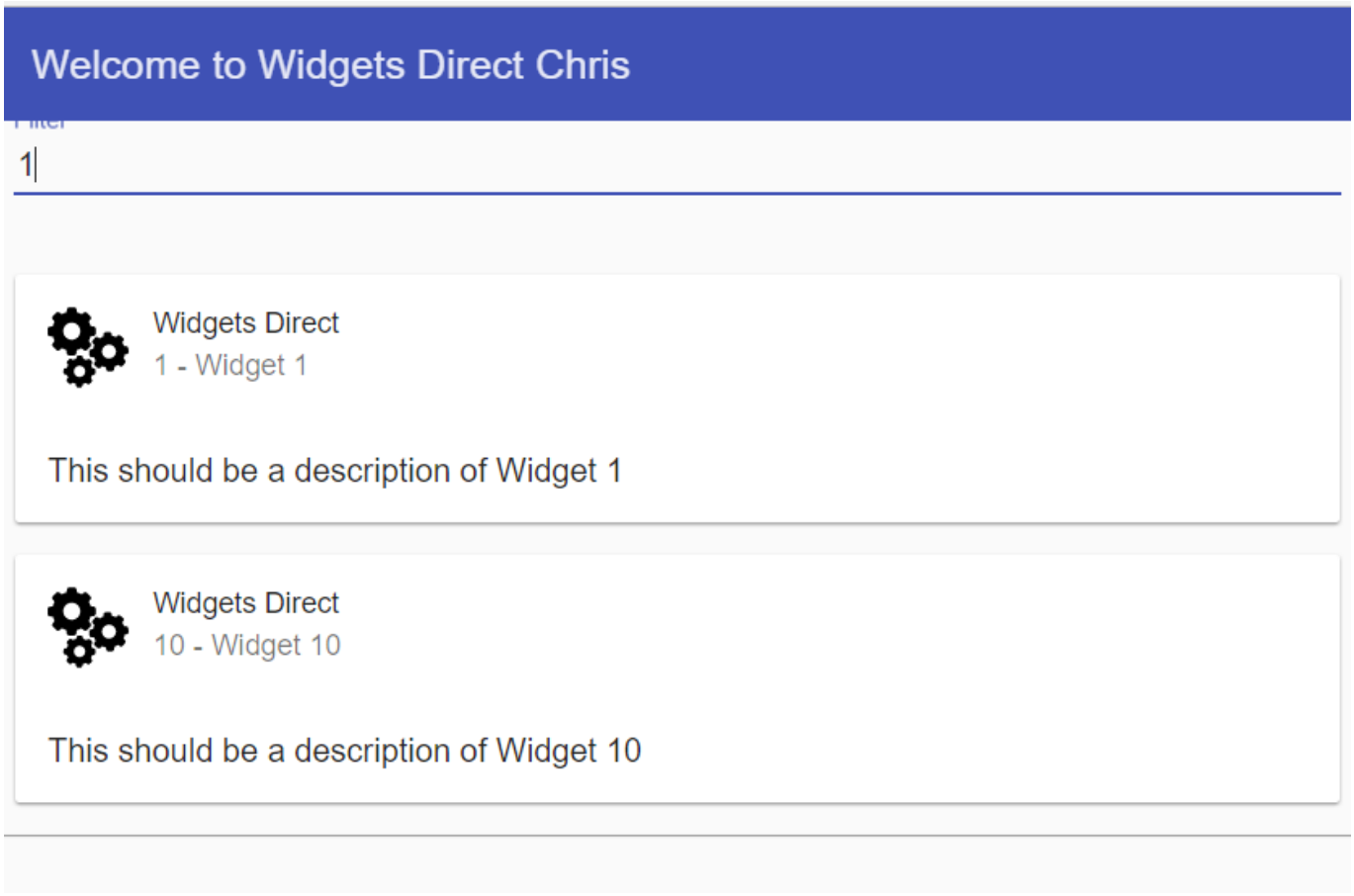
```
<!-- Page content starts here -->
<md-input-container style="margin:5px;" >
        <label>Filter</label>
        <input type="text" ng-model="widgetFilterText">
</md-input-container>

<!-- List of Widgets starts here -->
<md-card ng-repeat="widget in widgets">
 <md-card-header>
  <md-card-avatar>
   <img src="img/logo.svg"/>
  </md-card-avatar>
  <md-card-header-text>
```



We now have a text box, but typing in it does nothing. How can we hook it up to our list of widgets. Ng-repeat provides numerous filters to rearrange the data dynamically (all done on client side, so no need to request sorted data again from server). We will feed the variable being set by the text box into a filter, and this will restrict the data for us

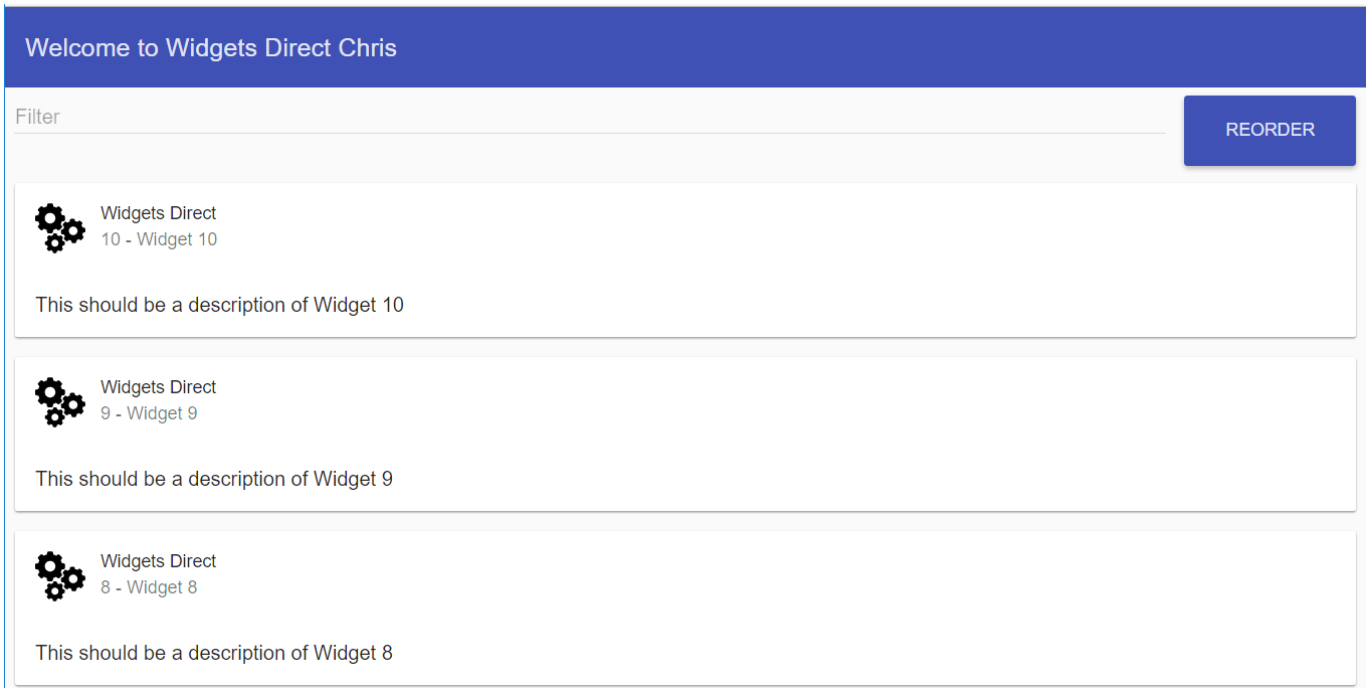
```
<!-- List of Widgets starts here -->
<md-card ng-repeat="widget in widgets | filter : widgetFilterText">
 <md-card-header>
  <md-card-avatar>
   <img src="img/logo.svg"/>
```

If we type in the Filter input now, we instantly get a filtered list. The filtering is applied to all fields of the list objects, so we can restrict on ID or Name.



We now have filtering, but being able to reorder the data may be useful too. We can use the orderBy filter to do this, and luckily for us, these filters stack so we can use the reordering in conjuction with our new text filter. We first add a variable to hold the sorting state, then add a button to the page which will toggle the boolean state of the variable (we will grab the button code from the Material demo page). Finally, we will add a switch onto the OrderBy filter to apply the sort order we want (field is ascending order on that field, -field is descending). Let's put it all together and see what we get

```
angular
  .module('WidgetsDirect', ['ngMaterial'])
   .controller('PageController', ['$q', '$scope', '$timeout', '$http', '$log', function($q, $scope, $timeout, $http, $log) {

    $scope.message = "";
    $scope.sortAsc = true;


    <div layout="row">
    <md-input-container style="margin:5px;" flex=85>
           <label>Filter</label>
           <input type="text" ng-model="widgetFilterText">
    </md-input-container>
    <md-button class="md-raised md-primary" ng-click="sortAsc=!sortAsc" flex>Reorder</md-button>
    </div>
    <!-- List of Widgets starts here -->
    <md-card ng-repeat="widget in widgets | filter : widgetFilterText |  orderBy: !sortAsc ? 'Id' : '-Id'">
```



SUCCESS! We can now both Filter and Sort (or both) our data instantly!

Recap

In this lesson we:

1. Add Material text inputs and buttons to our page
2. Learned about ng-repeat filters
3. Implemented a freetext filter on our data
4. Implemented a Reorder function on our data

In our next lesson we will:

- Hook our service up to a persistent class and start serving real data

This article is part of a multi-part series on using Angular on top of Caché REST services. The listing of the full series can be found at the Start Here page

#Angular #CSP #HTML #JavaScript #JSON #REST API #Frontend #Caché

Source URL: https://community.intersystems.com/post/lets-write-angular-1x-app-cach%C3%A9-rest-backend-part-5