Build Lists of Data with Angular

This is the 2nd blog post in our series on Angular. The first part entitled **Get Started with Angular** shows you how to add Angular to a project. You should read that blog first if you are not familiar with adding Angular to a project, don't know what a module or a controller is, or want to understand basic data binding.

In this blog post you will build a couple of different types of lists of data using the **ng-repeat** directive. Building unordered lists and HTML tables is very straight-forward in Angular. As with anything in Angular, you need to have a variable defined on the $scope variable that contains the data you wish to display. Use the ng-repeat directive and data binding to iterate over the data and create bulleted list items or <tr> and <td> elements.

Create an Unordered List

Start with the project from the first blog post entitled **Get Started with Angular**. Open the project, then locate and open the \Scripts\productController.js file. Add a new variable to the $scope variable in the controller function as shown in the code highlighted in bold below.

```
function PTCController($scope) {
  var vm = $scope;

  // Create a list of categories
  vm.categories = [
    { CategoryName: 'Videos' },
    { CategoryName: 'Books' },
    { CategoryName: 'Articles' }
  ];
}
```

This code creates an array of object literals. Each object is a category object with a single property called CategoryName. This CategoryName property is set to a string such as 'Videos', 'Books', or 'Articles. These string values are to be displayed in an unordered list on our HTML page.

Locate the AngularSample01.html page in the root of the project. Copy and paste this HTML page back into the root of the project. Rename this newly copied file to **AngularSample02**.html. Replace everything within the <div class="container"> with the following HTML.

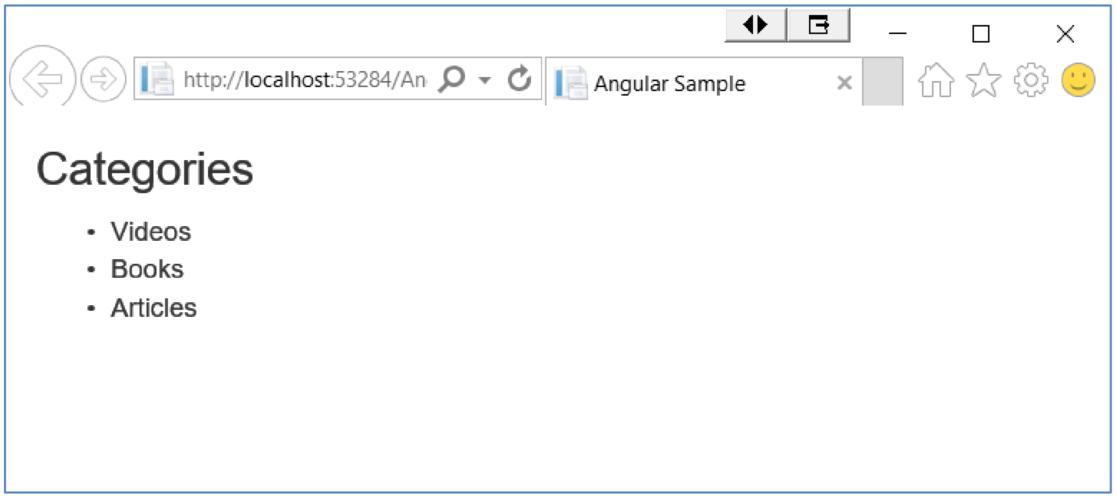
```
<h3>Categories</h3>
<div class="row">
  <div class="col-xs-12">
    <ul>
      <li ng-repeat="category in categories">
        {{category.CategoryName}}
      </li>
    </ul>
  </div>
</div>
```

The **ng-repeat** (also written as **ngRepeat**) directive contains two variable names and the word ‘in’. The first variable name ‘category’ is a name that you are using for a local variable within this loop. Think of this like the variable name you create in a foreach statement in C#. The second variable name, ‘categories’ refers to the categories variable you created in the controller on the $scope variable. The ‘in’ word simply separates the local variable from the variable on the $scope. Again, this is just like a foreach statement in C#.

The ng-repeat directive instructs Angular to iterate over the collection of objects in the categories variable and place each object into the variable named category. You are also instructing Angular to create a new <li> for each object in the collection. Angular then looks for any data binding tokens between the <li> and the </li> tags. If it finds one, it evaluates the expression. In this case the expression says to look for a property name called CategoryName on the object just retrieved from the categories collection. If it can get the value from the CategoryName property it displays that value between the <li> and the </li> elements.

At this point you should be able to run the page. Go ahead and run the page, and if you did everything correctly, you should see a page that looks similar to Figure 1.

Copyright © 2016 by PDSA, Inc.
All rights reserved worldwide. Reproduction is strictly prohibited.



Categories

- Videos
- Books
- Articles

Figure 1: An unordered list of categories

Create an HTML Table

Let's now use the ng-repeat directive to build an HTML table of products. Create an object literal array of product objects to build that table. Add a new variable named 'products' to your $scope variable in your controller function, as shown in the sample below.

```
function PTCController($scope) {
  var vm = $scope;

  vm.products = [
    { ProductName: 'Video 1', Price: 10 },
    { ProductName: 'Video 2', Price: 10 },
    { ProductName: 'Book 1', Price: 20 },
    { ProductName: 'Article 1', Price: 5 },
    { ProductName: 'Article 2', Price: 6 },
  ];
}
```

The above code creates an array of object literals where each one is a product object. These product objects contain two properties: ProductName and Price. Some default data has been filled in to give us something to display as shown in Figure 2.

Copyright © 2016 by PDSA, Inc.
All rights reserved. Reproduction is strictly prohibited.

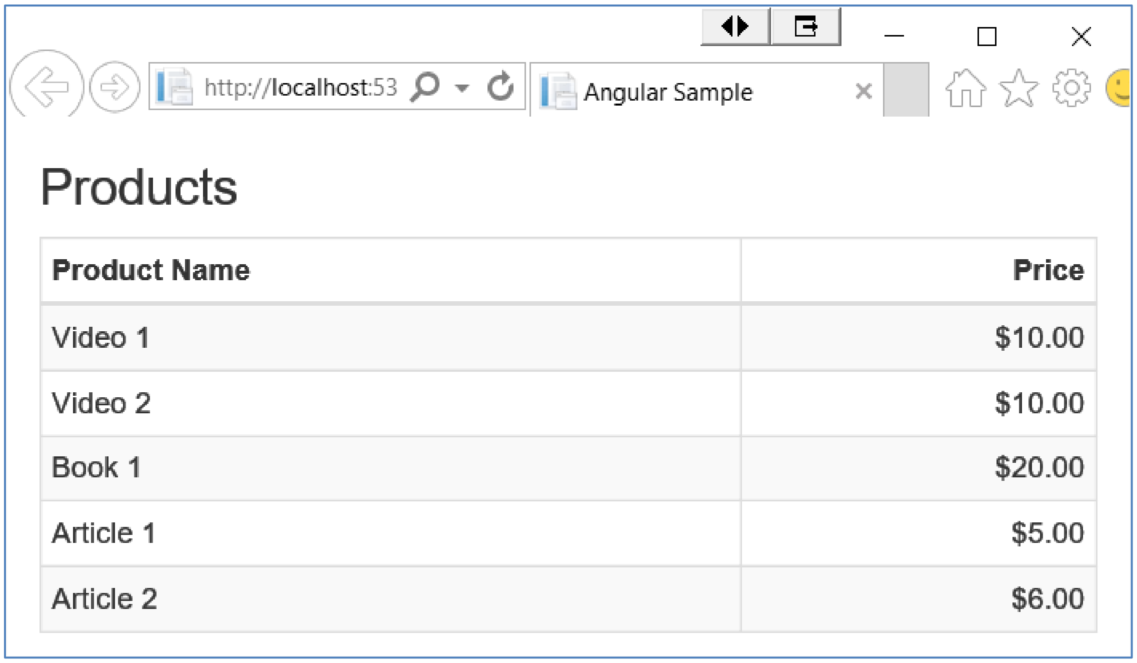
Locate the AngularSample02.html page in the root of the project. Copy and paste this HTML page back into the root of the project. Rename this newly copied file to **AngularSample03**.html. Replace everything within the <div class="container"> with the following HTML.

```
<h3>Products</h3>
<table class="table table-bordered table-striped
              table-condensed">
  <thead>
    <tr>
      <th>Product Name</th>
      <th class="text-right">Price</th>
    </tr>
  </thead>
  <tbody>
    <tr ng-repeat="product in products">
      <td>{{product.ProductName}}</td>
      <td class="text-right">
         {{product.Price | currency: $ }}
      </td>
    </tr>
  </tbody>
</table>
```

In the code above you use the ng-repeat directive to build a list of <tr> elements for each product object in the products collection. For each product object build two <td> elements. The first <td> element contains the ProductName value. The second <td> element displays the price; however, there is something a little extra in the data binding token.

This something extra is called a 'filter' and is used to format the data coming from the data binding expression. You place a vertical pipe after the property being bound, then add the filter named 'currency', followed by a colon and a US dollar sign symbol ($). This instructs Angular that it should take the price value, treat it as a decimal value, and format it according to United States currency formatting rules. There are other filters you can use in data binding: filter, number, date, json, lowercase, uppercase, limitTo, orderBy. You can learn more about these filters at https://docs.angularjs.org/api/ng/filter.

Copyright © 2016 by PDSA, Inc.
All rights reserved worldwide. Reproduction is strictly prohibited.



Products

Product Name	Price
Video 1	$10.00
Video 2	$10.00
Book 1	$20.00
Article 1	$5.00
Article 2	$6.00

Figure 2: A table created from Angular data binding

Summary

In this blog post you learned to use the ng-repeat directive to create a list of items. You first learned to build an unordered list using an array of category objects. Next, you built an HTML table of product objects and even saw how to format a currency value using a filter.

Sample Code

You can download the code for this sample at www.pdsa.com/downloads. Choose the category "PDSA Blog", then locate the sample **PDSA Blog Sample: Build Lists of Data with Angular**.

Copyright © 2016 by PDSA, Inc.
All rights reserved. Reproduction is strictly prohibited.