Learn Angular

BY MAKING A QUIZ APP FROM SCRATCH

HUNGRYTURTLECODE.COM

Part 2

Check Out The Whole Course Index

Using An Angular Controller To Add Content

In the last part we wrote our first bits of Angular code. One of those bits was the code that instantiates the controller for our list view. In this part we will take that Angular Controller and use it to dynamically insert data into our HTML. This gives us great control over the content that is on our page.

As the saying goes, there are many ways to skin a cat. What that means for us now is that there is more than one way that we can create properties on our controller and insert that data into our HTML.

In this article I will explain the two main methods of doing this and the pros and cons of both. I will of course also tell you which method I prefer and why.

If you want to see the app for yourself, check it out here.

The git repo can be found here.

Angular Controller Video Tutorial

As always, the more visually inclined can just watch this video and you will receive all the same information as you would from the article. If you prefer to read just scroll down past the video.

Click the image to go to the video on youtube or go to https://www.youtube.com/watch?v=mCDI3ZH3E58&index=2&list=PLqr0oBkln1FBmOjK24_B4y_VAA8736wPq

The next part can be found here

Method 1: $scope Service

The first method and the most commonly used method in beginners tutorials is using an Angular Service (more on what these are later in the course) call $scope. Although it is the most common, in my opinion it is not the best method. But I will explain it anyway.

We start off inside the list controller that we created in the previous tutorial and into the function we pass **$scope** . We can view $scope as simply an object that we are passing into our function. We can then attach properties onto that object and have access to those properties in our HTML.

So for example we could attach a property called “dummyData” onto $scope like this:

```
function ListController($scope){
    $scope.dummyData = "Hello  World";
}
```

Heading back into the HTML we could use that wonderful {{}} syntax to grab hold of that property like this:

```
<div ng-controller="listCtrl">
    {{dummyData}}
</div>
```

This will display the text “Hello World” out inside the div. Fantastic.

The Problems With $scope.

This approach is fine in smaller applications but as your applications grow just having bindings to a property like “dummyData” may get confusing. Especially if you have a property called dummyData inside more than one of your controllers, all of which have a different value.

This lack of explicit declaration of the data you are using can become a problem. I prefer to make everything explicit and immediately obvious what I am trying to do.

This is where the next method of doing this comes in.

Method 2: Controller As Syntax

The next method removes the $scope from our function and instead we bind our properties onto the "this" object inside our function.

To make this easier I usually set "this" equal to a variable at the start of the function and use that variable throughout. This saves some potential confusion later on.

```
function ListController(){
    var vm = this;

    vm.dummyData = "Hello World";
}
```

I used the variable name "vm" which simply stands for "view model". We are attaching these properties onto the view model – the view being our HTML.

We then attached that same dummyData property onto vm like we did with $scope earlier.

Our Code Is Broken!

If you now try to render the HTML using this controller code you will find the code no longer works. This is because we have to make a few changes in our HTML. This is where the name of the "Controller As" syntax will become apparent.

We need to make a few small changes. The first of which is changing **ng-controller="listCtrl"** to **ng-controller="listCtrl as list"** .
What we are doing here is creating an alias for our controller. Notice is is the name of the controller as before, then "as list".

So we are referring to our controller as list. In other words, when we use the name “list”, Angular will know we are referring to the listCtrl controller.

Now inside the {{}} we can refer to the exact controller which the dummyData property is on:

```
<div ng-controller="listCtrl as list">
    {{list.dummyData}}
</div>
```

Ahhhh! Yes! Everything is explicit now. No more potential confusion. When we see list.dummyData there is no doubt at all as to where the dummyData property is coming from. Just how we like it.

Which To Use?

Ultimately It does not really matter which of these methods you use. Just pick one and be consistent with it. Consistency is the key when you are coding, especially if you are working in teams.

However, if I was to recommend a method to use, I would definitely recommend going with the controller as syntax. It may be a bit more typing in the short term, but it will save you a lot of headaches in the future.

Onwards to Part 3

In part 3 we will be taking a look at another directive within Angular – ng-repeat. We will be using that to start building the markup for the list of turtles.

See you over there.

Adrian