Learn Angular

BY MAKING A QUIZ APP FROM SCRATCH

HUNGRYTURTLECODE.COM

Part 7

Check Out The Whole Course Index

Angular Services – Serving (Some) Of Your Needs

Fully armed with the knowledge on how to hide and show HTML elements, we can now go ahead and create the quiz controller that we will show at the same time that we hide the list controller – when the user clicks the “start quiz” button. Then we can learn about Angular Services and how they solve some problems that may arise.

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

The git repo can be found here.

Pick Your Poison – Text Or Video

As always, you can read this article or you can watch the video below to get the same information.

Click the image to go to the video on youtube or go to https://www.youtube.com/watch?

v=HrbkZO5Mt0g&index=7&list=PLqr0oBkln1FBmOjK24_B4y_VAA8736wPq

The next part can be found here

Below our list controller markup in the HTML we can create new controller div for our quiz. Just like we did with our list controller we will use the “controller as” syntax.

```
<div ng-controller="quizCtrl as quiz">
    <!-- Rest of the markup goes here -->
</div>
```

Of course, we will add this controller into another JavaScript file in our controller directory. So we will need to reference that script in the HTML. So are application scripts at the bottom of our HTML looks like this now.

```
<script src="js/app.js"></script>
<script src="js/controllers/list.js"></script>
<script src="js/controllers/quiz.js"></script>
```

Let's Create Another Angular Controller

To start off the controller, we will use the immediately invoked function expression (IIFE)that we have seen in earlier parts. The rest of the setup will be familiar as well. The only change is the name of the controller that we create.

```
(function(){

    angular
        .module("turtleFacts")
        .controller("quizCtrl", QuizController);

    function QuizController(){

        var vm = this;

    }

})();
```

How Do We Show the Quiz Controller With the Quiz Is Active?

At this point, you may or may not have realised that we are running into a problem. Hiding the list controller was simple, the quizActive flag is on the list controller, so we can manipulate it and then use it for the ng-hide.

However, we want to show the quiz controller at the same time. So ideally, we would use the same quizActive property to do that right? quizActive flicks to true and the ng-hide on the list controller hide the list controller and we use ng-show on the quizController to show that at the same time.

Unfortunately, Angular does not allow controllers to communicate like that. Controllers are totally separate. So we do not have access to the quizActive flag on the list controller when we are inside the quiz controller. So what shall we do?

Introducing Angular Services

This is where we can use a useful tool in the Angular Toolbox – services. There are many types of services available in Angular. Too many to cover in this tutorial series. So for now, we will be focusing on one type of service called Factories.

Factories in Angular will allow us to share some data between the controllers – which is exactly what we want.

A common use case for services like factories in angular would be for example, an application with multiple controllers, all of which need to pull some data from an API. In that case, it would be possible to write the API calls into every controller separately. But this is repeating code and breaking a key rule of code.

Instead, that API logic could be put into a service and all controllers can be given access to that service. Note that a controller wouldn't automatically have access, you would have to explicitly allow it.

Now, all the controllers can make an API call without having to repeat code. Perfect. This also serves to separate out concerns and ensure that all bits of code are only doing one thing. Added bonus.

Because controllers can share code via a service, it is possible to use the service to contain properties that all the controllers will need access to. This is our use case.

Get Building Your Factory!

We will create a new directory in our project called factories, which is, of course, where we will store any factories we create. We can then create a new script called quizmetrics.js which will be a factory that holds all data relevant to the general operation of the quiz app that all controllers may need access to.

Again, we use an IIFE to start things off. Then call our turtleFacts module. But this time we call a new method – .factory. The factory method is much like the controller method in that it accepts two arguments – the name of the factory and the function that defines the factory.'

```
(function(){

    angular
        .module("turtleFacts")
        .factory("quizMetrics", QuizMetrics);

    function QuizMetrics(){


    }

})();
```

A Brief Aside About How Factories Are Built
A factory does some logic then it returns something. It could be an object, or just a single property or a function or anything. Whatever it returns, is what the controllers that use the function will have access to.

So for our purposes, we will build an object that contains many properties that we will need in our controllers. Then this object is what we will return from the factory and ultimately will be

what is shared between the controllers.

We will start creating the object inside the factory by adding the quizActive property that is currently in our list controller. We can then remove it from the list controller once that is done.

```
function QuizMetrics(){

    var quizObj = {
        quizActive = false
    };

    return quizObj;

}
```

Don’t forget to add the script that references the factory to the end of the HTML. I placed the code in js/factories/quizMetrics.js, so that is the source in my HTML script tag.

On To Part 8

Now that we have the factory roughly built, we can start refactoring the list controller to remove the quizActive property and then inject the factory into the controller so we have access to the property again.

I will see you over there in part 8.

Adrian