AngularJS Controllers

What's Hot?

Who's excited about what right now?

Review Yesterday

ngClassEven, ngClassOdd, ngForm, ngSubmit, inputs, ngChecked, ngSelected, ngInclude, ngList

Today

Controllers!

Learning objectives

- Students can demonstrate the use of basic controllers.
- Students can explain the purpose of controllers, and what does not belong in a controller.
- Students can create new controllers and templates, and correctly link them into an app.
- Students can explain how a well-structured Angular app fits the MVC and SPA models.

Controller Basics

Controllers provide **presentation logic** for a given view or component. Controllers interact with this view via the scope object.

```
myApp.controller('MyController', function($scope) {
      $scope.message = { hello: "Hello World"};
});
```

A few things to note:

- A controller is a function. Its dependencies are injected as parameters. Here, the only dependency is the $scope service.
- **Presentation logic** is subtly different from **business logic**. Presentation logic is the minimum amount of logic required to coerce the data into the format(s) required for presentation to the user. Any substantial calculations or API interactions are probably better-classified as business logic which belongs in services or perhaps even on your server.

- Recall that adding an object to $scope provides you with two-way data binding between the JavaScript (our controller) and the HTML (our view, or template).
- We are adding an object called 'message' to the $scope object. More on this later.

The simplest way to use a controller is via ng-controller:

```
<div ng-controller="MyController">
      <p ng-bind="message.hello"></p>
</div>
```

More often, however, we will bind a controller to a template via our **router** or via a **directive**. More on these topics later . . .

Adding Functions to Controllers

It is common to add functions as well as state to a given template. The functions are called when events are fired via built-in directives such as ng-click or ng-model which then interact with and change our internal state. Consider this example from the official AngularJS Docs:

```
var myApp = angular.module('myApp', []);
myApp.controller('DoubleController', DoubleController);
function DoubleController() {
   var dc = this;
   dc.num = 1;

   dc.double = double;

   function double(value) {
      return value * 2;
   }
}

<div ng-controller="DoubleController as dc">
  Two times <input ng-model="dc.num"> equals {{ double(dc.num) }}
</div>
```

- Do you see any problems with this example?
    - What if the user types a value that is coerced to a String rather than a Number?

An aside: the **digest**.

Reminder: use ng-bind instead of {{ }}. It eliminates the flash of the curly braces after the HTML has been rendered by the browser, but before AngularJS has bootstrapped and replaced them with their values.

Review

Controllers

Mini Project

Fork the ngBootSeed below, and add a contact page.

Resources

John Papa's AngularJS Style Guide
https://github.com/johnpapa/angular-styleguide#controllers
My Angular-Bootstrap Seed (starter app)
https://github.com/MountainlandWEB/ngBootSeed

Tomorrow

Modules