

AngularJS Basics

Estimated time for completion: 45 minutes

Overview:

In this lab you will use AngularJS to create a simple application to manage and edit a list of movies.

Goals:

* Create a simple application using AngularJS
* Create an AngularJS main module and controller
* Configure data binding and visibility in the view

Lab Notes:

Don’t forget to reference the AngularJS documentation at <http://docs.angularjs.org/api>.

Creating a movies list

In this part of the lab, you will create a basic AngularJS application and build a list of movies the user can scroll through. In the next part of the lab you will add an edit screen allowing users to select a movie and edit its details.

Criteria:

* The Index.html page is intended to be the main page for working with movies. This page is left without any AngularJS directives and with an empty main module. It is your job to implement these.

**Helpful links:**

* [The AngularJS documentation](http://docs.angularjs.org/api).

Steps:

1. Open up the App.js, located in Public/App, and create a new AngularJS module named “moviesApp”.
2. Add an AngularJS controller named “MoviesCtrl” to thi smodule. This controller requires the $scope service to provide data to the HTML view.
3. There is a global variable movies, defined in Movies.js, that needs to be added to the $scope so the table with movies can be displayed. Note that using global variables like the movies array is not a good practice and you will improve on this in another lab about using services and factories.
4. Open the Index.html in the Public folder and add the required AngularJS directives for the table of movies to be displayed.
   1. Use ngApp to load the moviesApp.
   2. Use ngController to load the MoviesCtrl.
   3. Use ngRepeat to create a row for each movie in the array of movies.
   4. Use {{}} or ngBind to display the movie Title, Director, Year and Rating for each movie.
5. Run and test the application.
   1. The Node.js application can be started using Node Server.js or using the provided start.bat.
   2. Once the server has been started you can start the browser and navigate to http://localhost:8080/.

.

Adding the movie editor

In this part of the lab, you will add markup and code to allow the user to edit movie details.

Criteria:

* Currently you can view a list of movies. However we also need the ability to edit them. In this part of the lab you will add an edit form to allow the user to select a movies, edit the details and then save those changes.

**Helpful links:**

* [The AngularJS documentation](http://docs.angularjs.org/api).

Steps:

1. Open the Index.html.
2. There is an edit form as an HTML comment. Uncomment this form so it is shown at the bottom of the page.
3. Add a click handler using the ngClick directive to each row in the table on call a function named select() on the $scope.
4. Open the App.js and add the select() function to $scope in the controller to store the selected movie in a property named selectedMovie.
5. Add ngModel data bindings directives in the Index.html to the input elements in the input form to allow the user to edit the movie details.
6. Add the array of directors and countries to the $scope. Both are available as global variables as with the movies array in step one.
7. Add ngOptions directives to the two select elements in the markup to populate the select input controls with directors and countries.
8. Run and test the application.

Hiding the movies table when editing

In this part of the lab, display or hide parts of the UI based on what the user is doing.

Criteria:

* Currently you will always see the movies table and edit form. In this section you will change that so that only one of the two is visible at the time.

**Helpful links:**

* [The AngularJS documentation](http://docs.angularjs.org/api).

Steps:

1. Open the Index.html.
2. Use the ngShow and ngHide directives to only show the movies table when there is no selectedMovie and only show the movies edit form when the selectedMovie points to a movie.
3. Add a save() function to the scope to set the $scope.selectedMovie to null when a user clicks on save
4. Add the ngClick directive the Save button to call the previous function.
5. Run and test the application.
   1. When the HTML page loads you should just see a list of movies.
   2. When you click on a movie you should just see an edit form for that movie.
   3. When you click save you should see the list of movies again.

Solutions:

The final solution for this lab is available in the ~/after directory.