CS 572 Modern Web Applications

Najeeb Najeeb, PhD (najeeb@miu.edu)

V1.1.0

JavaScript Full Stack Development

- MongoDB
  - NoSQL database (document store)
  - Stores JSON documents
- Express
  - JavaScript web framework
  - On top of Node
- Angular
  - JavaScript UI framework
  - Single Page Applications
- Node
  - JavaScript server-side platform
  - Single threaded, fast and scalable

Roadmap and Outcomes

- Node.js: write asynchronous (non-blocking) code. Understand node platform to start a project.
- Express: setup express and get requests and send back responses. REST API.
- MongoDB: what NoSQL DB looks like. Full API interacting with DB.
- AngularJS: Investigate AngularJS and architect it. A single page application.
- MEAN application: Learn by example. We will create a MEAN Games application.

Integrating MEAN

Setup

- Check endpoints working properly using REST browser plugin.
- Create angular-app folder in the application public folder.
- Add public/angular-app/app.js file (empty for now). This is angular app.
- Install AngularJS using npm (or any other way)
  - npm i angular angular-route
- Add the angular files as dependencies to project
  - <script src="node_modules/angular/angular.js"></script>
    <script src="node_modules/angular-route/angular-route.js"></script>
- Include the angular application
  - <script src="angular-app/app.js"></script>
- Enable our node application to reach Angular (add app.use)
  - app.use("/node_modules", express.static(path.join(__dirname, "node_modules")));

MEAN

Title

Get List

Get One

Rating

Get the home page from Angular

Update index.html

```
…
<html ng-app="meanGames">
…
<body>
<div ng-view></div>
…
<script src="angular-app/game-list/game-list-
controller.js"></script>
</body>
}
```

MEAN

Title

Get List

Get One

Rating

Update angular-app/app.js

```
angular.module("meanGames", ["ngRoute"]).config(config);
function config($routeProvider) {
   $routeProvider.when("/", {
      templateURL: "angular-app/game-list/games.html",
      controller:" GamesController",
      controllerAs: "vm"
   });
}
```

Add the controller angular-app/game-list/game-list-controller.js

```
angular.module("meanGames", ["ngRoute"])
.controller("GamesController", GamesController);
function GamesController() {
 const vm= this;
 vm.title= "Mean Games App";
}
```

Add the template angular-app/game-list/gmaes.html

```
<H1>{{vm.title}}</H1>
```

MEAN

Title

Get List

Get One

Rating

Get the list of games from API

Update controller to make the request, public/angular-app/game-list/game-list-controller.js

```
function GamesController($http) {
  const vm= this;
  vm.title= "Mean Games App";
  $http.get("/api/games").then(function(response) {
    vm.games= response.data;
  }
}
```

Update the template angular-app/game-list/games.html

```
<H1>{{vm.title}}</H1>
<ul>
<li ng-repeat="game in vm.games">{{game.title}}</li>
</ul>
```

MEAN

Title

Get List

GetOne

Rating

Date routing to display a game

Update public/angular-app/app.js

```
…
function config($routeProvider, $locationProvier) {
$locationProvier.hashPrefix("");
…
.when("/game/:id", {
templateUrl: "angular-app/game-display/game.html",
controller: "GameController",
controllerAs: "vm"
});
}
```

Add controller to html page public/index.html

```
…
<script src="angular-app/game-data-factory/game-data-factory.js"></script>
<script src="angular-app/game-display/game-display-controller.js"></script>
```

MEAN

Title

Get List

GetOne

Rating

Create the data factory that calls the endpoints, and it used in our app.

Create public/game-data-factory/game-data-factory.js

```
angular.module("meanGames").factory("GameDataFactory", GameDataFactory);
function GameDataFactory($http) {
  return {
    getAllGames: getAllGames,
    getOneGame: getOneGame
  };
  function getAllGames() {
    return $http.get("/api/games").then(complete).catch(failed);
  }
  function getOneGame(id) {
    return $http.get("/api/games/"+id).then(complete).catch(failed);
  }
  function complete(response){
    console.log(response.data);
    return response.data;
  }
  function failed(error) {
    return error.status.statusText;
  }
}
```

Update game-list-controller.js to use the factory

```
function GamesController(GameDataFactory) {
 const vm= this;
 vm.title= "Mean Games App";
 GameDataFactory.getAllGames().then(function(response) {
  vm.games= response;
 });
}
```

MEAN

Title

Get List

GetOne

Rating

Get data about one game, add controller and template

Add controller public/angular-app/game-display/game-display-controller.js

```
angular.module("meanGames").controller("GameController", GameController);
function GameController(GameDataFactory, $routeParams) {
    const vm= this;
    const id= $routeParams.id;
    GameDataFactory.getOneGame(id).then(function(response) {
        vm.game= response;
    });
}
```

Add the template angular-app/game-display/game.html

```
<H1>Information about game: <p>{{vm.game.title}}</p></H1>
<p>
    Price: {{vm.game.price | currency}}<BR/>
    Minimum Players: {{vm.game.minPlayers}}<BR/>
    Maximum Players: {{vm.game.maxPlayers}}<BR/>
    Minimum Age: {{vm.game.minAge}}</BR>
    Publisher: {{vm.game.publisher.name}}
</p>
```

MEAN

Title

Get List

GetOne

Rating

Selecting a game from the list

Update public/angular-app/game-list/games.html

```
…
<li ng-repeat="game in vm.games"><a ng-
href="#/game/{{game._id}}">{{game.title}}</a></li>
}
```

Display Ratings

- What is the best way to display ratings?
- Number :(
- Images :/
- Stars :)
- Custom directive

Custom Directives

MEAN

Title

Get List

GetOne

Rating

Update template public/game-display/game-display-controller.js

…

vm.rating= response.rate;

…

Update template public/game-display/game.html

<H1>Information about game: {{vm.game.title}} - {{vm.rating}} </H1>

…

We would prefer to see stars according to this number

MEAN

Title

Get List

GetOne

Rating

Update template public/game-display/game.html

```
<H1>Information about game: <p>{{vm.game.title}} <game-rating
stars={{vm.rating}}></game-rating> </p></H1>
```

Add to html file index.html

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css">
...
<script src="angular-app/game-rating/game-rating-directive.js></script>
```

Update controller to send an array instead of a number game-display-controller.js

```
…
    vm.rating= _getStarRating(response.rate);
  });
}
function _getStarRating(stars) {
    return new Array(rate);
}
...
```

MEAN

Title

Get List

GetOne

Rating

Create directive public/angular-app/game-rating/game-rating-directive.js

```
angular.module("meanGames").directive("gameRating".
GameRating);
function GameRating() {
  return {
    restrict: "E",
    templateUrl: "angular-app/game-rating/rating.html",
    bindToController: true,
    controller: "GameController",
    controllerAs: "vm",
    scope: {
      stars: "@"
    }
  }
}
```

Create template public/angular-app/game-rating/rating.html

```
<span ng-repeat="star  in vm.rating track by $index"
class="glyphicon glyphicon-star"></span>
```

MEAN

Title

Get List

GetOne

Rating

Use component instead public/angular-app/game-rating/game-rating-directive.js

```
angular.module("meanGames").component("gameRating"
, {
    bindings: {
      stars: "*"
    },
    templateUrl: "angular-app/game-rating/rating.html",
    controller: "GameController",
    controllerAs: "vm",
});
```

Form Validation

Forms

Field Checking

Pattern Check

Check on submit

Add Game

We will use JSBin for this part

WE can use HTML 5 form validation attributes (required, email number, url) alos AngularJS form validation ng-minlength, ng-maxlength, ng-pattern

```
<form name="myForm">
<input type="text" name="name" required ng-minlength="3"
ng-maxlength="10" ng-model="name"></input>
</form>
<p>{{myForm.$pristine}}</p>
<p>{{myForm.$dirty}}</p>
<p>{{myForm.name.$pristine}}</p>
<p>{{myForm.name.$dirty}}</p>
<p>{{myForm.name.$valid}}</p>
<p>{{myForm.name.$invalid}}</p>
```

Forms

Field Checking

Pattern Check

Check on submit

Add Game

```
<form name="myForm">
<input type="text" name="name" required ng-minlength="3" ng-maxlength="10" ng-model="name"></input>
<span ng-show="myForm.name.$dirty && myForm.name.$invalid">
This feild requires 3-10 characters.
</span>
</form>
```

Forms

Field Checking

Pattern Check

Check on submit

Add Game

```
<form name="myForm">
<input type="text" name="name" required ng-pattern="/^[0-9]{2,3}$/" ng-model="name"></input>
<span ng-show="myForm.name.$dirty && myForm.name.$invalid">
This feild requires 2 or 3 digits.
</span>
</form>
```

Forms

Field Checking

Pattern Check

Check on submit

Add Game

```
<script>
 angular.module("myApp", []).controller("MyController", MyController);
 function MyController() {
  const vm= this;
  vm.message= "hello";
  vm.isSubmitted= false;
  vm.add= function() {
   if (vm.myForm.$valid) {
    console.log("Add to database...");
   } else {
    vm.isSubmitted= true;
   }
  }
 }
</script>
...
 <form name="vm.myForm" ng-submit="vm.add()">
 Please enter age greater than 9: <input type="text" name="name"
required ng-pattern="/^[0-9]{2,3}$/" ng-model="name"></input>
 <span ng-show="vm.myForm.name.$dirty &&
vm.myForm.name.$invalid && vm.isSubmitted">
   This feild requires 2 or 3 digits.
 </span>
 <button type="submit">Add data</button>
</form>
```

Forms

Field Checking

Pattern Check

Check on submit

Add Game

Add Game form to public/angular-app/game-list/games.html

```
<form name="vm.gameForm" ng-submit="vm.addGame()" >
    To add a new game please fill in all the fields below:<BR/>
    Title: <input type="text" name="title" required ng-model="vm.newGameTitle"
style="color:black"/><BR/>
    Price: <input type="text" name="price" required ng-
model="vm.newGamePrice" style="color:black"/><BR/>
    Year of Publication: <input type="text" name="year" required ng-
model="vm.newGameYear" style="color:black"/><BR/>
    Rating: <input type="text" name="rating" required ng-
model="vm.newGameRating" style="color:black"/><BR/>
    Minimum Number of Players: <input type="text" name="minPlayers" required
ng-model="vm.newGameMinPlayers" style="color:black"/>
    <span ng-show="vm.gameForm.minPlayers.$dirty &&
vm.gameForm.minPlayers.$invalid && vm.gameForm.isSubmitted"
style="color:black">Minimum players must be at least 1.</span>
    <BR/>
    Maximum Number of Players: <input type="text" name="maxPlayers" required
ng-model="vm.newGameMaxPlayers" style="color:black"/><BR/>
    Minimum Recommended Player Age: <input type="text" name="minAge"
required ng-model="vm.newGameMinAge" style="color:black"/><BR/>
    Designer name: <input type="text" name="designer" required ng-
model="vm.newGameDesigner" style="color:black"/><BR/>
    <button type="submit" class="btn-success">Add Game</button><BR/>
</form>
```

Forms

Field Checking

Pattern Check

Check on submit

Add Game

Add controller functionality for submitting. Update public/angular-app/game-list/game-list-controller.js

```
function GamesController(GameDataFactory) {
    const vm= this;
    vm.title= "Mean Games App";
    vm.isSubmitted= false;
    GameDataFactory.getAllGames().then(function(response) {
        // console.log(response);
        vm.games=  response;
    });
    vm.addGame= function(){
        const postData= {
            title: vm.newGameTitle,
            price: vm.newGamePrice,
            rate: vm.newGameRating,
            year: vm.newGameYear,
            rating: vm.newGameRating,
            minPlayers: vm.newGameMinPlayers,
            maxPlayers: vm.newGameMaxPlayers,
            minAge: vm.newGameMinAge,
            designers: vm.newGameDesigner,
        };
        if (vm.gameForm.$valid) {
            GameDataFactory.postGame(postData).then(function(response){
                console.log("Game saved");
                //
            }).catch(function(error) {
                console.log(error);
            });
        } else {
            vm.isSubmitted= true;
        }
    };
}
```

Forms

Field Checking

Pattern Check

Check on submit

Add Game

Update the Factory public/angular-app/game-data-factory/game-data-factory.js

```
function GameDataFactory($http) {
    return {
        getAllGames: getAllGames,
        getOneGame: getOneGame,
        postGame: postGame
    };

...
    function postGame(game) {
        return $http.post("/api/games/",
game).then(complete).catch(failed);
    }
```

Forms

Field Checking

Pattern Check

Check on submit

Add Game

Enable JSON processing. Update app05,js

```
...
app.use(express.urlencoded({extended  : false}));
app.use(express.json());
...
```