CS 572 Modern Web Applications

Najeeb Najeeb, PhD (najeeb@miu.edu)

V3.1.0

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.
- Angular: Investigate Angular and the architecture of an Angular application. Build a single-page application.
- MEAN application: Learn by example. We will create a MEAN Games application.

Angular SPA

1. Create Components to display the list of Games and display a single Game.
2. Create Routing to use the proper component.
3. Fetch the data from the Express Application.

Home

Home Page

Styles

Navigation

Footer

Routing

Add CSS & Images

bootstrap.min.css to assets/css

custom.css to assets/css

Use stylesheets in index.html

```
...
  <link rel="icon"  type="image/x-icon"
href="favicon.ico">
  <link href="assets/css/bootstrap.min.css"
rel="stylesheet"  />
  <link href="assets/css/custom.css"  rel="stylesheet"  />
</head>
...
```

Home Page
Styles
Navigation
Footer
Routing

Create the navigation Component

```
ng generate component navigation
```

Update navigation.component.html to display navigation option

```
<nav>
   <button>Home</button>
   <button>Games</button>
</nav>
```

Use the navigation component, update app.component.html

```
<app-navigation></app-navigation>
```

Home Page

Styles

Navigation

Footer

Routing

Create the footer Component

```
ng generate component footer
```

Update footer.component.html to display footer

```
<footer class="footer">
   <div class="container">
     <p class="test-muted text-center">
      <a href="https://compro.miu.edu" target="_blank">
        <img src="assets/images/compro-web-logo-442x112.webp" height="80"
alt="MIU Compro" />
      </a>
      <br />
      <small class="text black-50 text-center">&copy; 2022 Maharishi International
University. All Rights Reserved.
      </small>
     </p>
   </div>
</footer>
```

Add the image to the assets/iamges folder

Use the navigation component, update app.component.html

```
<app-navigation></app-navigation>
<app-footer></app-footer>
```

Home Page
Styles
Navigation
Footer
Routing

Create the footer Component

ng generate component home

Update home.component.html to display footer

<H1>Welcome to MEAN Games</H1>

Use the home component, update app.component.html

<app-navigation></app-navigation>

<app-home></app-home>

<app-footer></app-footer>

We wish to update the middle component based on a navigation option. How do we do that? We would like to have routing.

Home Page

Styles

Navigation

Footer

Routing

Add Routing to the module

```
...
import { RouterModule } from '@angular/router';
...
  imports: [
    BrowserModule,
    RouterModule.forRoot([
      {
        path: "",
        component: HomeComponent
      }
    ])
...
```

Use the router, update app.component.html

```
<app-navigation></app-navigation>

<router-outlet></router-outlet>

<app-footer></app-footer>
```

Home Page
Styles
Navigation
Footer
Routing

Update navigation.component.html to use routing

```
<nav>
   <button (click)="onHome()">Home</button>
   <button (click)="onGames()">Games</button>
</nav>
```

Add the navigation method to navigation.component.ts

```
 constructor(private _router:Router) { }
 onHome(): void {
  this._router.navigate(['']);
 }
 onGames(): void {
  this._router.navigate(['']);
 }
```

Games

Games Page

Component

Routing

Data

Create the games Component

```
ng generate component games
```

Update games.component.html to display a single game

```
<ul>
   <li *ngFor="let game of games">
      <a routerLink="/game/{{game._id}}">
        {{game.title}}
      </a>
   </li>
</ul>
```

Expose games from the component code games.component.ts

```
games: any[]= [{
   _id: "123",
   title: "Catan",
   price: 39.99
 }];
```

Games Page

Component

Routing

Data

Add Routing to the module

```
...
    {
      path: "games",
      component: GamesComponent
    }
  ])
...
```

Use the navigation, update navigation.component.ts

```
onGames(): void {
   this._router.navigate(['games']);
 }
```

Games Page

Component

Routing

Data

Use static routing, update navigation.component.html

```
<nav>
   <button routerLink="/">Home</button>
   <button routerLink="/games">Games</button>
</nav>
```

Games Page

Component

Routing

Data

Add a Game class in games.component.ts

```
class Game {
 #_id!: string;
 #title!: string;
 #year!: string;
 #rate!: number;
 #price!: number;
 #minPlayers!: number;
 #maxPlayers!: number;
 #minAge!: number;
 get _id() {return this.#_id;}
 get title() {return this.#title;}
 set title(title: string) {this.#title= title;}
 get year() {return this.#year;}
 get rate() {return this.#rate;}
 get price() {return this.#price;}
 set price(price: number) { this.#price= price;}
 get minPlayers() {return this.#minPlayers;}
 get maxPlayers() {return this.#maxPlayers;}
 get minAge() {return this.#minAge;}
 constructor(id: string, title: string, price: number) {
  this.#_id= id;
  this.#title= title;
  this.#price= price;
 }
}
```

Use the Game class

```
export class GamesComponent implements OnInit {
  games : Game[]= [];
...
```

Dependency Injection DI

Service

API Usage

Service

Component

CORS

Add a gamesData service

```
ng generate service gamesData
```

Update app.module.ts to use Http

```
Import { HttpClientModule } from '@angular/common/http';
...
imports: [
...
   HttpClientModule,
...
```

Update games-data.service.ts

```
export class GamesDataService {
   constructor(private http:HttpClient) { }
...
```

API Usage

Service

Component

CORS

Update games-data.service.ts

```
export class GamesDataService {
...
  public getGames(): Promise<Game[]> {
    const url: string= this.baseUrl + "games";
    return this.http.get(url).toPromise()
              .then(response => response as Game[])
              .catch(this._handleError);
  }
  private _handleError(err: any): Promise<any> {
    console.log("Service Error", err);
    return Promise.reject(err.message || err);
  }
...
```

API Usage

Service

Component

CORS

Update games-data.service.ts

```
export class GamesDataService {
...
  public getGames(): Observable<Game[]> {
    return this.http.get<Game[]>(this.baseUrl + '/games');
  }
...
```

API Usage

Service

Component

CORS

Update games.component.ts

```
export class GamesComponent implements OnInit {
  games: Game[]= [];
  constructor(private gamesService:GamesDataService) { }
  ngOnInit(): void {
    this.gamesService.getGames().subscribe(games  => {
      this.games= games;
    });
  }
}
```

API Usage

Service

Component

CORS

Update backend to support the Angular application

appxx.js

```
app.use("/api", function(req, res, next) {
    res.header('Access-Control-Allow-Origin',
'http://localhost:4200');
    res.header('Access-Control-Allow-Headers', 'Origin, X-
Requested-With, Content-Type, Accept');
    next();
});
```

Game

Game Page

Component

Service

Routing

Create the game Component

```
ng generate component game
```

Update game.component.html to display a single game

```
<H1>{{game.title}} - {{game.rate}}</H1>
<br/>
Year: {{game.year}}<br/>
Price: {{game.price | currency}}<br/>
Minimum Players: {{game.minPlayers}}<br/>
Maximum Players: {{game.maxPlayers}}<br/>
Minimum Age: {{game.minAge}}<br/>
```

Expose game from the component
code games.component.ts

```
game!: Game;
```

Game Page
Component
Service
Routing

Expose game from the component
code games.component.ts

```
 game!: Game;
 constructor(private route:ActivatedRoute, private
gameService:GamesDataService) {
  this.game= new Game("", "", 0);
  }
 ngOnInit(): void {
  const gameId= this.route.snapshot.params["gameId"];
  this.gameService.getGame(gameId).subscribe(game => {
    this.game= game;
  });
 }
```

Game Page
Component
Service
Routing

Expose game service, update code games-data.service.ts

```
public getGame(id: string):Observable<Game> {
  const url: string= this.baseUrl + "games/" + id;
  return this.http.get<Game>(url);
}
```

Game Page
Component
Service
Routing

Update the routing, update code app.module.ts

```
...
      {
       path: "game/:gameId",
       component: GameComponent
      },
      {
       path: "**",
       component: ErrorPageComponent
      }
...
```

Rate

Stars Rating

Usage

Component Template

Update game.component.html to display a single game

```
<H1>{{game.title}} - {{game.rate}}</H1>
...
```

Replace this with

```
<H1>{{game.title}} - <stars-rating [rating]=game.rate
></stars-rating> </H1>
...
```

Stars Rating

Usage

Component

Template

Create the stars-rating Component

```
ng generate component starsRating
```

Update stars-rating.component.ts

```
...
selector: 'app-stars-rating',
...
_rating: number= 0;
stars: number[]= [];
@Input()
set rating(rating: number) {
  this._rating= rating;
  this.stars= new Array<number>(rating);
};
```

Stars Rating

Usage

Component

Template

Create the stars-rating Component

```
ng generate component starsRating
```

Update stars-rating.component.ts

```
...
selector: 'app-stars-rating',
...
export class StarsRatingComponent implements OnInit, OnChanges
{
...
rating: number= 0;
stars: number[]= [];
ngOnChanges(changes: SimpleChanges): void {
  this.stars= new Array<number>(changes['rating'],currentValue);
};
```

Stars Rating

Usage

Component

Template

Update stars-rating.component.html

```
<span *ngFor="let star of stars">
   &#9733;
</span>
```

Main Points

- MEAN is the ultimate separation of concerns (SoC). Not only do we have each responsibility separated in code, but each one is handled by a different framework.