Angular Application – Movie DB (part 1)

Create new application

Using the CLI create new angular app typing in console “ng new movie-finder”. That will create a new angular app. To make sure everything is up and tuning, navigate in console to the project “src” directory and there type “ng serve –open” → that will start the app on localhost: 4200 and will open it in browser for you.

Create components and add style

Type in the console “ng g c movies” (that is short command for ng generate component movies). Make sure that the new component is added in “app.module”. Go to https://bootswatch.com/ and search for “Slate” (if you want you could choose different). Click on it and copy this link from URL https://bootswatch.com/4/slate/bootstrap.min.css. After that go to index.html and post this in the head part <link rel="stylesheet" href="https://bootswatch.com/4/slate/bootstrap.min.css">. After that go to app.component.html delete everything and type <app-movies></app-movies>. Save the whole project and you should see this on the browser:

Go to http://getbootstrap.com/docs/4.1/examples/starter-template/ and click right button on the mouse and select “View page source”. Copy the code inside the <nav>.

After that use the resources to achieve the following:

Sign up

Now go to moviedb and create an account. Go to your account in settings and navigate to API. From there you will see your details for API Key, API Read Access Token (v4 auth) and Example API Request. We will use them in the following steps.

Create a service and connect to the API

We will create a service in terminal which will be responsible for fetching the data from the API.

From the terminal in “src/app” type “mkdir service”, after that “cd service” and then type “ng g s movies”. That will create a service.

Reminder: Do not forget to import the service and register it as a provider in app.module. In order to make requests to the API we should add to our project HttpClientModule. After that we should import HttpClient in our service and inject it through the constructor.

In the service make constant which will hold our API key from MovieDb.

const apiKey = '{your api key}’;

After that go to out movies component and inject as a dependency (through the constructor) the movies service we just created.

Ok, now let’s make our first call to the API. Go to https://www.themoviedb.org/documentation/api/discover and see which API call is responsible for getting the most popular films and copy it.

Let’s go to the service and create a function which will be responsible for calling the API for the most popular movies.

Create 3 variable fields

1. path = 'https://api.themoviedb.org/3/';

2. popular = 'discover/movie?sort\_by=popularity.desc';

3. authentication = '&api\_key=';

The function should look like this:

Where the apiKey is our const which holds the key from our MovieDb profile. Notice that after calling the server we RETURN the result, because we are about to call this function from another place (in our case movies component) and there we will subscribe to the result.

Go to our movie component and in ngOnInit call our service with the new function and subscribe to it:

Check the browser console to see how the information is structured:

The same logic we will apply in order to get the movies in theaters.

List the results in the view

Let’s go to our movies component and get rid of the html. Instead of this add this html code:

<div class="panel panel-default">

<div class="panel-heading">

<h3 class="panel-title">Popular Movies</h3>

</div>

<div class="panel-body">

<div class="row">

<div \*ngFor="let movie of popular.results;let i=index" class="col-md-2">

<div \*ngIf="i < 6">

<img \*ngIf="movie.poster\_path" class="thumbnail" src="http://image.tmdb.org/t/p/w500/{{movie.poster\_path}}">

<h4>{{movie.title}}</h4>

<p>{{movie.release\_date}}</p>

<p><a class="btn btn-default" href="#">View Details</a></p>

</div>

</div>

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading">

<h3 class="panel-title">In Theaters</h3>

</div>

<div class="panel-body">

<div class="row">

<div \*ngFor="let movie of theaters.results;let i=index" class="col-md-2">

<div \*ngIf="i < 6">

<img \*ngIf="movie.poster\_path" class="thumbnail" src="http://image.tmdb.org/t/p/w500/{{movie.poster\_path}}">

<h4>{{movie.title}}</h4>

<p>{{movie.release\_date}}</p>

<p><a class="btn btn-default" href="#">View Details</a></p>

</div>

</div>

</div>

</div>

</div>

Notice how we use \*ngIf for our variables (instead of popular.results, use your created previous variable in which you have saved the results).

In movies.component.css (or .scss) file set the rule that the img should be with width of 100%

Your end result should look like this:

In the next part we will add routing to our website to create an actual Single-Page Application.

Angular Application – Movie DB (part 2)

We will continue working on our previous project.

Implement search logic

Go to app.component.html and copy the jumbotron section. Remove it from there and paste it in movies.component.html. Make changes to see the component look like this:

The whole html:

<div class="jumbotron">

<div class="container">

<h1>Find a movie</h1>

<p>Search for a movie using the form below:</p>

<form action="">

<input type="text" class="form-control">

<button >Search</button>

</form>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading">

<h3 class="panel-title">Popular Movies</h3>

</div>

<div class="panel-body">

<div class="row">

<div \*ngFor="let movie of popular.results;let i=index" class="col-md-2">

<div \*ngIf="i < 6">

<img \*ngIf="movie.poster\_path" class="thumbnail" src="http://image.tmdb.org/t/p/w500/{{movie.poster\_path}}">

<h4>{{movie.title}}</h4>

<p>{{movie.release\_date}}</p>

<p><a class="btn btn-default" href="#">View Details</a></p>

</div>

</div>

</div>

</div>

</div>

<div class="panel panel-default">

<div class="panel-heading">

<h3 class="panel-title">In Theaters</h3>

</div>

<div class="panel-body">

<div class="row">

<div \*ngFor="let movie of theaters.results;let i=index" class="col-md-2">

<div \*ngIf="i < 6">

<img \*ngIf="movie.poster\_path" class="thumbnail" src="http://image.tmdb.org/t/p/w500/{{movie.poster\_path}}">

<h4>{{movie.title}}</h4>

<p>{{movie.release\_date}}</p>

<p><a class="btn btn-default" href="#">View Details</a></p>

</div>

</div>

</div>

</div>

</div>

List more categories

Using your knowledge from previous exercise list two more section. One for popular kids movies and one for the best drama movies.

Your project should look like this:

and for drama:

Prepare for navigation

When we click on “view details” button we should be redirected to another page and be able to see the details for the selected movie.

To achieve that we will need navigation. Go to the directory where the app.module is and create new file app.routing.ts

and import

Note that you should import your MoviesComponent and create a new component called MovieComponent which will be responsible to list the details for a single movie.

After that export a varriable of type ModuleWithProviders and assigned to be: RouterModule.forRoot({here place the array you just created})

Go to app module and add to imports the routing varriable we have just exported form the app.routing file.

Set up the movie component

We should import in our component the service and ActivatedRoute from “@angular/router” and inject them through the constructor:

Go to our service and create new function called getMovie(id) which will take a parameter (id of the movie) and return the call from the API for the movie.

Return to the movie component and in the OnInit method subscribe to the router’s parameters and take the id. Below call the service with the new function and give the id as an argument

Now we should implement the link. Go to movies.component.html and replace the href links in “view details” buttons with routerLink=”/movie/{{movie.id}}”

Do not forget to replace all in app.component.html in container with <router-outlet></router-outlet>

When we click on some of the “view details” buttons the page should like this:

In order to finish this part of application we should add HTML template for our MovieComponent

<div \\ensure that there is a movie in order to render this div>

<div class="panel panel-default">

<div class="panel-heading">

<h3 class="panel-title">{{place movie title here}}</h3>

</div>

<div class="panel-body">

<div class="row">

<div class="col-md-5">

<img class="thumbnail" src="http://image.tmdb.org/t/p/w500/{{place poster path here}}">

</div>

<div class="col-md-7">

<ul class="list-group">

<li class="list-group-item">Genres: <span \*ngFor="let genre of movie.genres">{{list all genres for this movie}}</span></li>

<li class="list-group-item">Release Date: {{show the release date of the movie}}</li>

</ul>

<br>

<a \*ngIf="movie.homepage" href="{{place movie's home here}}" target="\_blank" class="btn btn-default">Visit Movie Website</a>

</div>

</div>

</div>

</div>

</div>

\* Working search form (optional)

We make our UI for our search form but it does not include any functionality. So we are going to use forms here.

In app.module import FormsModule from '@angular/forms' and add FormsModule to the imports

Our HTML should look like this:

<form #form="ngForm" (submit)="search(form.value)">

<input type="text" class="form-control" name="search" ngModel>

<button type="submit" class="btn btn-primary">Search</button>

</form>

Next steps:

1. Go to our service and create a function findAMovie(name)

2. After that go to moves.component.ts and create a function which will take the string from user input and will call the new service function. Subscribe to it and list the results in the view.

3. Render the results only if the user has clicked search button.

The results should look like this

**API Key (v3 auth)**

**df1873e8be545f002c4b9d369a35c525**

**Example API Request**

https://api.themoviedb.org/3/movie/550?api\_key=df1873e8be545f002c4b9d369a35c525

**API Read Access Token (v4 auth)**

eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkZjE4NzNlOGJlNTQ1ZjAwMmM0YjlkMzY5YTM1YzUyNSIsInN1YiI6IjVlMzFkNDhkNGNhNjc2MDAxNDRjNjBjOSIsInNjb3BlcyI6WyJhcGlfcmVhZCJdLCJ2ZXJzaW9uIjoxfQ.ZrIKQJAhZ9yzKkAhmhlsCIx7TEhGwcQ3Z5jozjI9SCM