Exercise (Instructions): Angular HTTP Client

Objectives and Outcomes

In this exercise you will learn to use the Angular HTTP client to make requests for data to a server and obtain and process the response. At the end of this exercise you will be able to:

- Use Angular HTTP client to obtain data from a server
- Process the HTTP response from the server to retrieve the data and use it in your application.

Exercise Resources

images.zip

Updating the Json-Server Configuration

- You will now update the json-server to enable it to serve up resources from its public folder. Go to the json-server folder and create a folder in their named public
- Download the images.zip file provided above and unzip it and move the images folder to the public folder that you created above
- Start the json-server by typing the following at the prompt. This will introduce a delay of 2 seconds before the server sends the reply to a request:

```
1  json-server --watch db.json -d 2000
```

Configuring the Base Server URL

- You need to import the HttpModule in app.module.ts by adding the following to the file:

```
import { HttpModule } from '@angular/http';

. . .

@NgModule({

  . . .
  imports: [
    . . .

    HttpModule
  ],

  . . .
```

- Create a new file named baseurl.ts in the shared folder and update its contents as follows:

```
export const baseURL = 'http://localhost:3000/';
```

- Open AppModule, import baseURL and update the AppModule's providers property of the @NgModule decorator as follows:

```
. . .
import { baseURL } from './shared/baseurl';

. . .

providers: [
  . . .
  {provide: 'BaseURL', useValue: baseURL}
  ]
```

Updating the Dish Service

- Create a new service named ProcessHTTPMsg in the services folder
- Import the service into AppModule and include it in the providers property of the @NgModule decorator.
- Open process-httpmsg.service.ts and update its contents as follows:

```
. . .

import { Observable } from 'rxjs/Observable';
import { Http, Response } from '@angular/http';

. . .


  public extractData(res: Response) {
    let body = res.json();
    console.log(body);
    return body || { };
  }

  . . .
```

- Open dish.service.ts file and update its contents as follows:

```
. . .
import { Http, Response } from '@angular/http';
import { baseURL } from '../shared/baseurl';
import { ProcessHTTPMsgService } from './process-httpmsg.service';

. . .

import 'rxjs/add/operator/catch';

. . .

  constructor(private http: Http,
              private processHTTPMsgService: ProcessHTTPMsgService) { }

. . .


  getDishes(): Observable<Dish[]> {
    return this.http.get(baseURL + 'dishes')
                    .map(res => { return this.processHTTPMsgService.extractData
                      (res); });
  }

  getDish(id: number): Observable<Dish> {
    return  this.http.get(baseURL + 'dishes/'+ id)
                    .map(res => { return this.processHTTPMsgService.extractData
                      (res); });
  }

  getFeaturedDish(): Observable<Dish> {
    return this.http.get(baseURL + 'dishes?featured=true')
                    .map(res => { return this.processHTTPMsgService.extractData
                      (res)[0]; });
  }

  getDishIds(): Observable<number[]> {
    return this.getDishes()
      .map(dishes => { return dishes.map(dish => dish.id) });
  }

  . . .

```

Updating Menu Component

- Open menu.component.ts and update it as follows:

```
import { Component, OnInit, Inject } from '@angular/core';

. . .

  constructor(private dishService: DishService,
    @Inject('BaseURL') private BaseURL) { }

. . .
```

- Also delete the DISHES import and the selectedDish variable and the onSelectDish() method.
- Open menu.component.html and update it as follows:

```
. . .
        <img height="200px" src="{{BaseURL + dish.image}}" alt={{dish.name}}>

. . .
```

- Similarly update dishdetail.component.html, dishdetail.component.ts, home.component.html and home.component.ts
- Save the changes and do a Git commit with the message "HTTP Part 1".

Conclusions

In this exercise you learnt to use the Angular HTTP client to obtain data from a server and process the HTTP response from the server.

Mark as completed