JWT Token Authentication In Angular 14 And .NET Core 6 Web API

Jaydeep Patil Updated date Jul 02, 2022 7.3k 2 1

WebAPP.zip

Download Free .NET & JAVA Files API

We are going to discuss the JWT Authentication in Angular 14 step-by-step

If you want to learn the basics and details of JWT Token then check the following URL over there I explained the basics and details of JWT Authentication and Authorization.

JWT Token Authentication And Authorization

Also, I recommend you to read the following article before starting this article, I explained how to set up a backend server application using.NET Core 6

JWT Token Authentication Using The .NET Core 6 Web API

In this section, we are going to discuss the following things step-by-step

- Introduction
- Create Angular Application
- Add Bootstrap and Toaster Module
- Auth Guards
- Angular Service
- Create Angular Components

Let's start above things one by one

Introduction

- JSON Web Token is the open standard (RFC 7519) self-contained way that will be used to transmit the data securely over the different environments as a JSON Object.

- RFC (Request for Comment) is the shortened form of Remote Function Call and Formal Document from the Internet Engineering Task Force. RFC 7519 JSON Web Token (JWT) May 2015 Copyright Notice Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.
- JWT is the trusted way of authentication because it is digitally signed and secret using HMAC Algorithm or sometimes using a public/private key using RSA.
- Basically, HMAC stands for Hashed-based Message Authentication Code, it uses some great cryptographic hashing technique that provides us great security.
- Also, the JWT is part of great Authentication and Authorization Framework like OAuth and OpenID which will provide a great mechanism to transfer data securely.

Create Angular Application

Step 1

Create Angular Application using the following command

ng new WebAPP

Step 2

We use bootstrap in this application. So, use the following command to install bootstrap

npm install bootstrap

next, add the bootstrap script inside the angular.json file inside the scripts and styles section

```
"styles": [
            "src/styles.css",
            "./node_modules/bootstrap/dist/css/bootstrap.min.css"
          ],
          "scripts": [
            "./node_modules/bootstrap/dist/js/bootstrap.min.js"
          ]
```

Step 3

Install Toaster module for pop-up and notification

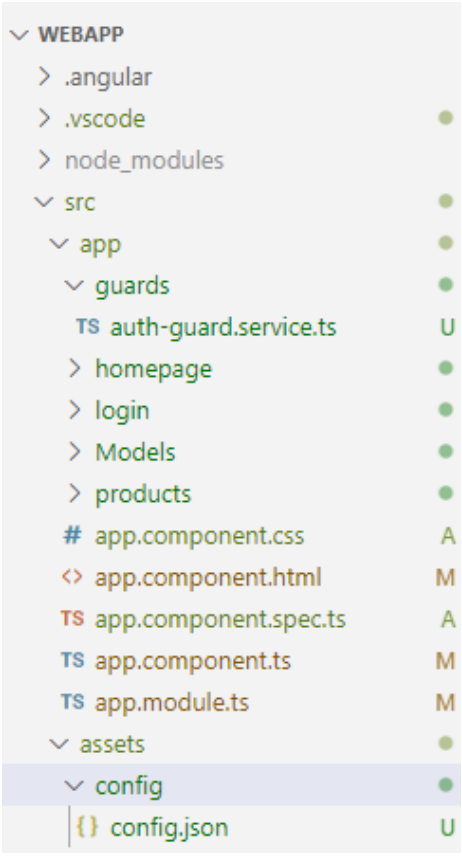
npm install ngx-toastr –save

Then, add the toaster in the styles section inside the angular.json file

```
"styles": [
            "src/styles.css",
            "node_modules/ngx-toastr/toastr.css",
            "./node_modules/bootstrap/dist/css/bootstrap.min.css"
          ],
          "scripts": [
            "./node_modules/bootstrap/dist/js/bootstrap.min.js"
          ]
```

Step 4

Application structure



Step 5

Create config folder inside assets and create config.json file inside that as shown below and put backend application API URL inside that

```
{
    "apiServer": {
      "url": "https://localhost:7299",
      "version": "v1"
    }
}
```

Step 6

Create Auth Guard inside the guard's folder

```
import { Injectable } from '@angular/core';
import { CanActivate, Router } from '@angular/router';
import { JwtHelperService } from '@auth0/angular-jwt';

@Injectable()
export class AuthGuard implements CanActivate {

```

```
  constructor(private jwtHelper: JwtHelperService, private router: Router
  }
  canActivate() {

    //get the jwt token which are present in the local storage
    const token = localStorage.getItem("jwt");

    //Check if the token is expired or not and if token is expired then r
    if (token && !this.jwtHelper.isTokenExpired(token)){
      return true;
    }
    this.router.navigate(["login"]);
    return false;
  }

}
```

So, here you can see we take the JWT token from the local storage and later on check if the token is expired or not, If the token is expired then it will redirect to login and return false.

Step 7

Open the App Component files and add the following code inside that

app.component.ts

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

  title = 'ProductWebAPP';
}
```

app.component.html

```
<router-outlet></router-outlet>
```

Step 8

Create a Model folder inside the app directory and create a Product class inside that

```
export class Products {
    productId: any;
```

```
    productName?: string;
    productCost?: number;
    productDescription?: string;
    productStock?: number;
  }
```

Step 9

Create Homepage component

homepage.component.ts

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { JwtHelperService } from '@auth0/angular-jwt';

@Component({
  selector: 'app-homepage',
  templateUrl: './homepage.component.html',
  styleUrls: ['./homepage.component.css']
})
export class HomepageComponent  {

  constructor(private jwtHelper: JwtHelperService, private router: Router
  }

  isUserAuthenticated() {
    const token = localStorage.getItem("jwt");
    if (token && !this.jwtHelper.isTokenExpired(token)) {
      return true;
    }
    else {
      return false;
    }
  }

  public logOut = () => {
    localStorage.removeItem("jwt");
  }

}
```

homepage.component.html

```
<nav class="navbar navbar-expand-lg navbar navbar-dark bg-dark">
  <a class="navbar-brand" href="#">Product Application</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" dat
```

```
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarText">
    <ul class="navbar-nav mr-auto">
      <li *ngIf="isUserAuthenticated()" class="nav-item">
        <a class="nav-link" routerLink="/product">Products</a>
      </li>
      <li *ngIf="isUserAuthenticated()" class="nav-item">
        <a class="nav-link" (click)="logOut()">Logout</a>
      </li>
    </ul>
  </div>
</nav>
<div *ngIf="!isUserAuthenticated()">
  <login></login>
</div>
<div *ngIf="isUserAuthenticated()" style="color:green;">
  <h2>
    Welcome to the Product Application
  </h2>
</div>
```

Step 10

Create Login Component

login.component.ts

```
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Component } from '@angular/core';
import { Router } from "@angular/router";
import { NgForm } from '@angular/forms';
import configurl from '../../assets/config/config.json';
import { JwtHelperService } from '@auth0/angular-jwt';
import { ToastrService } from 'ngx-toastr';

@Component({
  selector: 'login',
  templateUrl: './login.component.html'
})
export class LoginComponent {
  invalidLogin?: boolean;

  url = configurl.apiServer.url + '/api/authentication/';

  constructor(private router: Router, private http: HttpClient,private jw
```

```
    private toastr: ToastrService) { }

  public login = (form: NgForm) => {
    const credentials = JSON.stringify(form.value);
    this.http.post(this.url +"login", credentials, {
      headers: new HttpHeaders({
        "Content-Type": "application/json"
      })
    }).subscribe(response => {
      const token = (<any>response).token;
      localStorage.setItem("jwt", token);
      this.invalidLogin = false;
      this.toastr.success("Logged In successfully");
      this.router.navigate(["/product"]);
    }, err => {
      this.invalidLogin = true;
    });
  }

  isUserAuthenticated() {
    const token = localStorage.getItem("jwt");
    if (token && !this.jwtHelper.isTokenExpired(token)) {
      return true;
    }
    else {
      return false;
    }
  }

}
```

login.component.html

```
<form class="form-signin" #loginForm="ngForm" (ngSubmit)="login(loginForm
    <div class="container-fluid">
        <h2 class="form-signin-heading">Login</h2>
        <div *ngIf="invalidLogin" class="alert alert-danger">Invalid user
        <br/>
        <label for="username" class="sr-only">Email address</label>
        <input type="email" id="username" name="username" ngModel class="
        <br/>
        <label for="password" class="sr-only">Password</label>
        <input type="password" id="password" name="password" ngModel clas
        <br/>
        <button class="btn btn-lg btn-primary btn-block" type="submit">Lo

```

```
    </div>
</form>
```

Step 11

Create Products Component

products.component.ts

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, Validators } from '@angular/forms';
import { Observable } from 'rxjs';
import { ProductsService } from '../products/products.service';
import { Products } from '../Models/Products';
import { Router } from '@angular/router';
import { JwtHelperService } from '@auth0/angular-jwt';
import { ToastrService } from 'ngx-toastr';

@Component({
  selector: 'app-product',
  templateUrl: './products.component.html',
  styleUrls: ['./products.component.css']
})
export class ProductsComponent implements OnInit {

  ProductList?: Observable<Products[]>;
  ProductList1?: Observable<Products[]>;
  productForm: any;
  massage = "";
  prodCategory = "";
  productId = 0;
  constructor(private formbulider: FormBuilder,
     private productService: ProductsService,private router: Router,
     private jwtHelper : JwtHelperService,private toastr: ToastrService)

  ngOnInit() {
    this.prodCategory = "0";
    this.productForm = this.formbulider.group({
      productName: ['', [Validators.required]],
      productCost: ['', [Validators.required]],
      productDescription: ['', [Validators.required]],
      productStock: ['', [Validators.required]]
    });
    this.getProductList();
  }
  getProductList() {
```

```
38      this.ProductList1 = this.productService.getProductList();
39      this.ProductList = this.ProductList1;
40    }
41    PostProduct(product: Products) {
42      const product_Master = this.productForm.value;
43      this.productService.postProductData(product_Master).subscribe(
44        () => {
45          this.getProductList();
46          this.productForm.reset();
47          this.toastr.success('Data Saved Successfully');
48        }
49      );
50    }
51    ProductDetailsToEdit(id: string) {
52      this.productService.getProductDetailsById(id).subscribe(productResult
53        this.productId = productResult.productId;
54        this.productForm.controls['productName'].setValue(productResult.pro
55        this.productForm.controls['productCost'].setValue(productResult.pro
56        this.productForm.controls['productDescription'].setValue(productRes
57        this.productForm.controls['productStock'].setValue(productResult.pr
58      });
59    }
60    UpdateProduct(product: Products) {
61      product.productId = this.productId;
62      const product_Master = this.productForm.value;
63      this.productService.updateProduct(product_Master).subscribe(() => {
64        this.toastr.success('Data Updated Successfully');
65        this.productForm.reset();
66        this.getProductList();
67      });
68    }
69
70    DeleteProduct(id: number) {
71      if (confirm('Do you want to delete this product?')) {
72        this.productService.deleteProductById(id).subscribe(() => {
73          this.toastr.success('Data Deleted Successfully');
74          this.getProductList();
75        });
76      }
77    }
78
79    Clear(product: Products){
80      this.productForm.reset();
81    }
82
83    public logOut = () => {
```

```
    localStorage.removeItem("jwt");
    this.router.navigate(["/"]);
  }

  isUserAuthenticated() {
    const token = localStorage.getItem("jwt");
    if (token && !this.jwtHelper.isTokenExpired(token)) {
      return true;
    }
    else {
      return false;
    }
  }

}
```

products.component.html

```
<nav class="navbar navbar-expand-lg navbar navbar-dark bg-dark">
  <a class="navbar-brand" href="#">Product Application</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" dat
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarText">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" routerLink="/">Home</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" (click)="logOut()">Logout</a>
      </li>
      <li *ngIf="!isUserAuthenticated()" class="nav-item active">
        <a class="nav-link" routerLink="/product">Products</a>
      </li>
    </ul>
  </div>
</nav>

<form class="form-horizontal" [formGroup]="productForm">
    <h1 style="text-align: center;">Welcome to Angular 14 CRUD with .NET
    <div>
    <div class="form-group">
      <label class="control-label col-sm-2" for="pwd">Product Name:</labe
      <div class="col-sm-10">
        <input type="text" class="form-control" id="txtProductName" formC
          placeholder="Name of Product">
```

```
      </div>
    </div>
    <br />
    <div class="form-group">
      <label class="control-label col-sm-2" for="pwd">Product Description
      <div class="col-sm-10">
        <input type="text" class="form-control" id="txtProductDescription
          placeholder="Product Description">
      </div>
    </div>
    <br />
    <div class="form-group">
      <label class="control-label col-sm-2" for="pwd">Product Cost:</labe
      <div class="col-sm-10">
        <input type="text" class="form-control" id="txtProductCost" formC
      </div>
    </div>
    <br />
    <div class="form-group">
      <label class="control-label col-sm-2" for="pwd">Product Stock Avail
      <div class="col-sm-10">
        <input type="text" class="form-control" id="txtStock" formControl
      </div>
    </div>
    <br />
     <div class="form-group">
      <div class="container">
        <div class="row">
          <div class="col-sm">
            <button type="submit" class="btn btn-primary" (click)="PostPr
          </div>
          <div class="col-sm">
            <button type="submit" class="btn btn-primary" (click)="Update
          </div>
          <div class="col-sm">
            <button type="submit" class="btn btn-primary" (click)="Clear(
          </div>
        </div>
      </div>
    <br />
    </div>
    <div>
      <div class="alert alert-success" style="text-align: center;"><b>Pro
      <div class="table-responsive" style="text-align: center;">
        <table class="table table-striped">
          <thead>
```

```
          <tr>
            <th scope="col">#</th>
            <th scope="col">Product Name</th>
            <th scope="col">Description</th>
            <th scope="col">Cost</th>
            <th scope="col">Stock</th>
            <th scope="col">Action</th>
          </tr>
        </thead>
        <tbody>
          <tr *ngFor="let prd of ProductList | async; index as i">
            <th scope="row">{{ i + 1 }}</th>
            <td>{{prd.productName}}</td>
            <td>{{prd.productDescription}}</td>
            <td>{{prd.productCost}}</td>
            <td>{{prd.productStock}}</td>
            <td><button type="button" class="btn1" matTooltip="Click Edit
              |
              <button type="button" class="btn1" matTooltip="Click Delete
            </td>
          </tr>
        </tbody>
        </table>
      </div>
    </div>
    </div>
  </form>
```

Step 12

Next, Create a Product Service to send all our request and fetch the data from backend application

```
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { Products } from '../Models/Products';
import configurl from '../../assets/config/config.json'

@Injectable({
  providedIn: 'root'
})
export class ProductsService {

  url = configurl.apiServer.url + '/api/product/';
  constructor(private http: HttpClient) { }
```

```
  getProductList(): Observable<Products[]> {
    return this.http.get<Products[]>(this.url + 'ProductsList');
  }
  postProductData(productData: Products): Observable<Products> {
    const httpHeaders = { headers:new HttpHeaders({'Content-Type': 'appli
    return this.http.post<Products>(this.url + 'CreateProduct', productDa
  }
  updateProduct(product: Products): Observable<Products> {
    const httpHeaders = { headers:new HttpHeaders({'Content-Type': 'appli
    return this.http.post<Products>(this.url + 'UpdateProduct?id=' + prod
  }
  deleteProductById(id: number): Observable<number> {
    return this.http.post<number>(this.url + 'DeleteProduct?id=' + id, nu
  }
  getProductDetailsById(id: string): Observable<Products> {
    return this.http.get<Products>(this.url + 'ProductDetail?id=' + id);
  }
}
```

Step 13

Put the following code inside the app module

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppComponent } from './app.component';
import { ProductsComponent } from './products/products.component';

import {HttpClientModule} from '@angular/common/http';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { RouterModule, Routes } from '@angular/router';
import { JwtModule } from "@auth0/angular-jwt";
import { AuthGuard } from './guards/auth-guard.service';
import { HomepageComponent } from './homepage/homepage.component';
import { LoginComponent } from './login/login.component';
import { ToastrModule } from 'ngx-toastr';

//all components routes
const routes: Routes = [
  { path: '', component: HomepageComponent },
  { path: 'product', component: ProductsComponent, canActivate: [AuthGuar
  { path: 'login', component: LoginComponent },
];

//function is use to get jwt token from local storage
```

```
export function tokenGetter() {
  return localStorage.getItem("jwt");
}

@NgModule({
  declarations: [
    AppComponent,
    ProductsComponent,
    HomepageComponent,
    LoginComponent
  ],
  imports: [
    BrowserModule,
    HttpClientModule,
    FormsModule,
    ReactiveFormsModule,
    RouterModule.forRoot(routes),
    JwtModule.forRoot({
      config: {
        tokenGetter: tokenGetter,
        allowedDomains: ["localhost:7299"],
        disallowedRoutes: []
      }
  }),
  ToastrModule.forRoot()
  ],
  providers: [AuthGuard],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

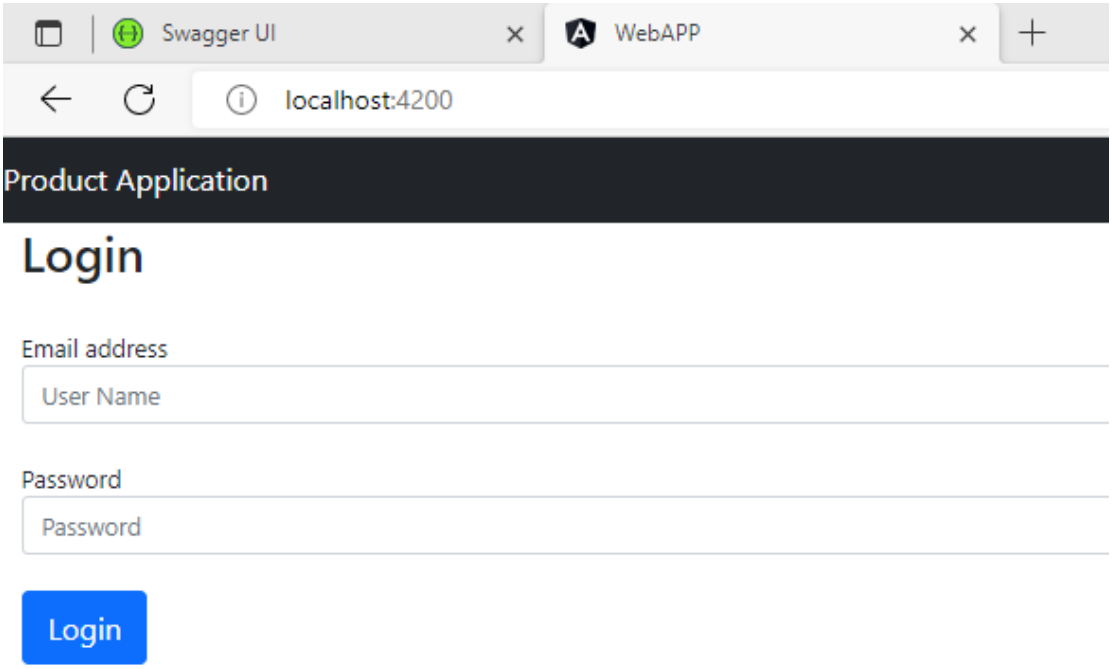
So, here you can see, first, we put some routes and create one method to get the JWT token from local storage and Also configure JWT module and Auth Guard inside that.

Step 14

Finally, run your application

npm start

You will see the login page when running the application



After login, you will see the product page

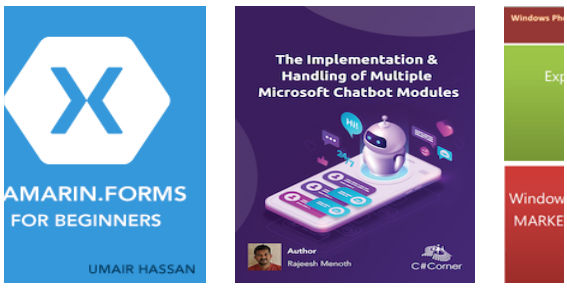
Conclusion

So, we discussed all JWT Authentication in Angular 14 step-by-step and how to store tokens in local storage and usage of it inside the product application.

.NET Angular C# Programming Software Development

> Next Recommended Reading
> Client-side Application For JWT Refresh Token In Angular 13

OUR BOOKS



Jaydeep Patil TOP 1000

C# Corner MVP | Fullstack Developer with 2.5+ yrs of experience in .NET Core API, Angular 8+, SQL Server, Docker, Kubernetics and Azure

https://www.linkedin.com/in/jaydeep-patil-b13a54134

612 **61.7k** **1**

1 **2**

Type your comment here and press Enter Key (Minimum 10 characters)

Follow Comments

Very good explanation....

Hamid Khan

324 **7.4k** **1.5m**

Jul 07, 2022

0 0 Reply

Is it safe to store jwt in the local storage?

Balamurugan Thanikachalam

2129 **16** **0**

Jul 04, 2022

1 0 Reply

FEATURED ARTICLES

Implementing Google reCAPTCHA v3 In Angular 14

Onion Architecture In ASP.NET Core 6 Web API

ASP.NET Core - Middleware

Working With ASP.NET 6 IDistributedCache Provider For NCache

Challenges With Microservices

TRENDING UP

01 Typescript Express Node Server

02 The A - Z Guide Of SQL Views

03 ASP.NET CORE - CRUD Using Dependency Injection

04 How to Migrate (P2V) Physical to a Virtual Data Center - Convergence VMware Virtualization Concepts

05 JWT Token Authentication In Angular 14 And .NET Core 6 Web API

06 Easy To Learn .NET 6.0 And Angular - Getting Started Admin LTE Design

07 Caching In Entity Framework Core Using NCache

08 Implement In-Memory Cache In The .NET Core API

09 Types Of Cloud Computing In VMware Virtualization Concepts

10 Getting Started With Angular Electron Application Development

Learn Angular 8 In 25 Days

CHALLENGE YOURSELF

Angular 13

GET CERTIFIED

Stratis Blockchain Developer