



To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

Our best price of the year. **Get 20% off new memberships** for a limited time.

User Authentication and Authorization in angular 16 with JWT

4 min read · Apr 13, 2024



Faruk taiwo

Follow



Listen



Share



In this article, we delve into the intricacies of user authentication and authorization in Angular 16, focusing specifically on the utilization of JSON Web Tokens (JWT). JWT

has become a popular choice for implementing authentication and authorization due to its simplicity. To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

Lets dive right into it then.

1. Authentication Service

The very first service we want to create is an authentication service to handle login, logout, and also token management.

```
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

export class AuthService {
  constructor(private http: HttpClient) {}

  isLoggedIn: boolean = false;

  login(userDetails: { username: string; password: string }): Observable<boolean> {
    return this.http.post<any>('http://examples/api/login', userDetails)
      .pipe(
        map(response => {
          localStorage.setItem('JWT_Token', response.token);
          this.isLoggedIn = true;
          return true;
        }),
        catchError(error => {
          console.log(error);
          this.isLoggedIn = false;
          return of(false);
        })
      );
  }

  logout(): void {
    localStorage.removeItem('JWT_Token');
    this.isLoggedIn = false;
  }

  isAuthenticated(): boolean {
    return this.isLoggedIn;
  }
}
```

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

In this service, the login method sends a POST request to the login API endpoint with user credentials, Upon successfully vetting the credentials a success response containing the JWT token is sent back, we then store the JWT in our local storage and update the authentication status. The logout method clears the JWT token from local

storage and resets the authentication status. The `isAuthenticated` method returns whether the user is authenticated. To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

2. Authorization Guard Service

This service ensures secure navigation by restricting access to routes based on user authentication status.

```
import { CanActivateFn, Router } from '@angular/router';
import { AuthService } from '../auth.service';
import { inject } from '@angular/core';

export const AuthGuardService : CanActivateFn () => {

  let isAuthenticated = inject(AuthService).isAuthenticated()
  let router = inject(Router)

  if (isAuthenticated) {
    return true;
  } else {
    router.navigate(['/Login']);
    return false;
  }
}
```

By implementing the `CanActivateFn`, it intercepts route activation attempts. Initially, we define a Function `AuthGuardService` of type `CanActivateFn`. Next, we assign an instance of the `isAuthenticated` method from our Authorization service to the variable `isAuthenticated`, and the Router to the variable `router`. Subsequently, we establish a condition: if `isAuthenticated` (user is authenticated) it returns `true` navigation proceeds. otherwise, it returns `false`, denying user access to the protected route and redirecting them to the 'login' page using the router service.

3. Interceptor Service

Get Faruk taiwo's stories in your inbox

Join Medium for free to get updates from this writer.

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

This Service (`JwtInterceptor`) intercepts outgoing HTTP requests and adds the JWT to the Authorization header before sending them to the server.

```
import { HttpEvent, HttpHandler, HttpInterceptor, HttpRequest } from '@angular/co

export class JwtInterceptor implements HttpInterceptor {

  intercept(req: HttpRequest<any>, next: HttpHandler) {
    const token = localStorage.getItem('JWT_Token');
    if (token) {
      const authReq = req.clone({
        setHeaders: {
          Authorization: `Bearer ${token}`
        }
      });
      return next.handle(authReq);
    } else {
      return next.handle(req);
    }
  }
}
```

We begin by importing essential modules from Angular's HTTP package. Subsequently, we define a class named `JWTInterceptor`, which implements the `HttpInterceptor` interface. Within this class, we define the `intercept` method, accepting two parameters: `req` (the HTTP request) and `next` (the HTTP handler for the next interceptor in the chain).

Retrieving the JWT (JSON Web Token) from the local storage, which was previously stored by our `AuthService`, is the initial step. We validate the presence of the token, and if it exists, we create a modified copy of the outgoing request (`authReq`) with an added Authorization header containing the JWT token. This modified request, now including the token, is then forwarded to the server.

Consequent resources. I To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy. ted unaltered.

Regardless, we ensure the request proceeds to the server by invoking `next.handle(request)`. This interceptor plays a crucial role in augmenting outgoing requests with the necessary authorization credentials, facilitating secure communication with the server.

4. Add our Authorization guard service to our `Approuting.module.ts`

```
import { Routes } from '@angular/router';
import { HomeComponent } from './home.component';
import { ProductsComponent } from './products.component';
import { AuthGuardService } from './auth-guard.service';

const routes: Routes = [
  { path: 'home', component: HomeComponent, canActivate: [AuthGuardService] },
  { path: 'products', component: ProductsComponent, canActivate: [AuthGuardService] },
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule {}
```

Here, we define the routes for our application using the `Routes` array. Each route is an object with properties defining the path and the component to be displayed when that path is accessed. The `canActivate` property is used to guard routes, ensuring that certain conditions must be met before allowing access. In this case, `AuthGuardService` which we earlier created is then imported so that it implements the `CanActivate` interface to provide route guarding functionality.

5. Provide interceptor in the `app.module.ts`

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

```

import { HttpClientModule, HTTP_INTERCEPTORS } from '@angular/common/http';
import {
  To make Medium work, we log user data. By using Medium, you agree to
  our Privacy Policy, including cookie policy.
import { JwtInterceptor } from './jwt.interceptor';

@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule, HttpClientModule, AppRoutingModule],
  providers: [
    AuthGuardService,
    { provide: HTTP_INTERCEPTORS, useClass: JwtInterceptor, multi: true }
  ],
  bootstrap: [AppComponent]
})
export class AppModule {}

```

In this Module, we are providing the `HeadersInterceptor` class as an interceptor using the `HTTP_INTERCEPTORS` token. The `multi: true` option ensures that the interceptor is appended to the existing array of interceptors rather than replacing them.

Conclusion

The `AuthService` manages user login/logout and token storage, while the `AuthGuardService` ensures secure navigation by restricting access to authenticated users. The `JwtInterceptor` intercepts outgoing HTTP requests to include the JWT token in the Authorization header. By integrating these components into the application's routing and module configurations, a robust authentication and authorization system can be established, enhancing security and user experience.

Happy coding!

Well, if you find this quite educative do follow me on [linkedin](#)

Typescript

JavaScript

Angular





Written by,

29 followers · 25 following

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

Follow

Responses (2)



Write a response

What are your thoughts?



daksh patel
Sep 25, 2024



Hey, i tried this approach and found the error as below:

ERROR Error: Uncaught (in promise): TypeError: guard is not a function

TypeError: guard is not a function

so, i removed the authGaurd from providers (from app-routing), now its working fine. (because of new version of angular).

Overall good article, Thank you



2



1 reply

[Reply](#)



marcordero
Sep 1, 2024



Hi. I'm trying this from Angular 18 and the value from isLoggedIn gets lost at the guard. I had to use an observable to overcome this problem, but aside from that, this is really great. Thanks.




1 reply

[Reply](#)

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

More from

```
{
  "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "refreshToken": "dGhpcyBpc3MgdGVzdCB5ZWZyZXNoIHRva2Vu...",
  "expiresIn": 3600,
  "tokenType": "Bearer",
  "userId": "123456789",
  "issuedAt": "2024-07-24T12:00:00Z",
  "expiresAt": "2024-07-24T13:00:00Z"
}
```

 Faruk taiwo

Integrating refresh Tokens in Angular: Ensuring Secure and Seamless Authentication

In today's world of web development, ensuring a seamless and secure user experience is paramount. One critical aspect of this is managing...

Jul 29, 2024

See all from Faruk taiwo

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

Recommended from Medium



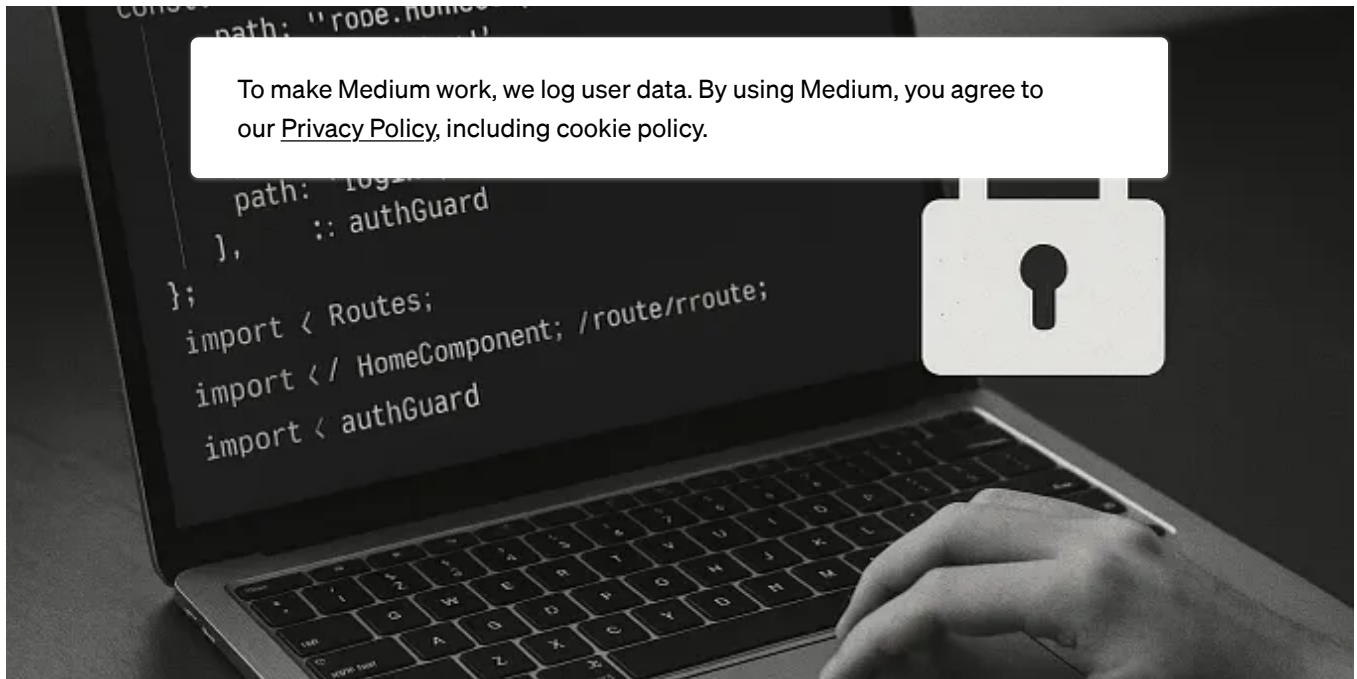
Stanislav Babenko


Why HttpOnly Cookies Don't Show Up on Page Reload in Angular (But Work When Routing)

A reader recently asked a great question after reading our last article on securing session cookies in Angular apps:

Jun 12  53  1





 Coding master

5 Angular Auth Guard Hacks That Instantly Level Up Your App Security

If your app trusts the client, an attacker will own the app. This sentence must stop you from scrolling. Security starts at the router...

★ Oct 17 🖱 1



Ramit Raj

Mastering Managem

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

Angular has long been a powerhouse in the world of web development, offering a robust framework for building scalable, maintainable, and...

Aug 20



Angular Interview Questions



In Level Up Coding by Pawan Kumawat

Top Angular Interview Questions for Mid-Senior Developers (2025)

Ace your next Angular interview with this handpicked collection of advanced, real-world questions covering RxJS, Dependency Injection...



Nov 10



1



To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

```
$ app.routes
import { Routes } from '@angular/router';
import { MsalGuard } from './auth/auth.guard';
import { HomeComponent } from './home/home.component';
import { SecureComponent } from './secure/secure.component';

export const routes: Routes = [
  {path: 'auth', component: MsalRedirectComponent},
  {path: '', component: HomeComponent},
  {path: 'secure', component: SecureComponent, canActivate: [MsalGuard]},
];
```



Maulik Patel

Microsoft Entra AD SSO with Angular + .NET core API

First Create App Registration for Web and API.



Jul 22



2





Ayush Maurya

Route-Sco

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

By default, inl

HttpClient service across the entire...

le by the



Oct 12



56



See more recommendations