



Авторизация с помощью JWT-токенов в клиенте JavaScript

Последнее обновление: 10.01.2022



В прошлой статье был рассмотрен процесс конфигурации и генерации JWT-токенов. Теперь посмотрим, как мы можем применить JWT-токен для авторизации в приложении. Для этого определим в файле **Program.cs** следующий код:

```
1 using Microsoft.AspNetCore.Authentication.JwtBearer;
2 using Microsoft.AspNetCore.Authorization;
3 using Microsoft.IdentityModel.Tokens;
4 using System.IdentityModel.Tokens.Jwt;
5 using System.Security.Claims;
6 using System.Text;
7
8 // условная бд с пользователями
9 var people = new List<Person>
10 {
11     new Person("tom@gmail.com", "12345"),
12     new Person("bob@gmail.com", "55555")
13 };
14
15 var builder = WebApplication.CreateBuilder();
16
17 builder.Services.AddAuthorization();
18 builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
19     .AddJwtBearer(options =>
20     {
21         options.TokenValidationParameters = new TokenValidationParameters
22         {
23             ValidateIssuer = true,
24             ValidIssuer = AuthOptions.ISSUER,
25             ValidateAudience = true,
26             ValidAudience = AuthOptions.AUDIENCE,
27             ValidateLifetime = true,
28             IssuerSigningKey = AuthOptions.GetSymmetricSecurityKey(),
29             ValidateIssuerSigningKey = true
30         };
31     });
```

```
32 var app = builder.Build();
33
34 app.UseDefaultFiles();
35 app.UseStaticFiles();
36
37 app.UseAuthentication();
38 app.UseAuthorization();
39
40 app.MapPost("/login", (Person loginData) =>
41 {
42     // находим пользователя
43     Person? person = people.FirstOrDefault(p => p.Email == loginData.Email && p.Password == loginData.Password);
44     // если пользователь не найден, отправляем статусный код 401
45     if(person is null) return Results.Unauthorized();
46
47     var claims = new List<Claim> {new Claim(ClaimTypes.Name, person.Email) };
48     // создаем JWT-токен
49     var jwt = new JwtSecurityToken(
50         issuer: AuthOptions.ISSUER,
51         audience: AuthOptions.AUDIENCE,
52         claims: claims,
53         expires: DateTime.UtcNow.Add(TimeSpan.FromMinutes(2)),
54         signingCredentials: new SigningCredentials(AuthOptions.GetSymmetricSecurityKey(), SecurityAlgorithms.HmacSha256));
55     var encodedJwt = new JwtSecurityTokenHandler().WriteToken(jwt);
56
57     // формируем ответ
58     var response = new
59     {
60         access_token = encodedJwt,
61         username = person.Email
62     };
63
64     return Results.Json(response);
65 });
66 app.Map("/data", [Authorize] () => new { message= "Hello World!" });
67
68 app.Run();
69
70 public class AuthOptions
71 {
72     public const string ISSUER = "MyAuthServer"; // издатель токена
73     public const string AUDIENCE = "MyAuthClient"; // потребитель токена
74     const string KEY = "mysupersecret_secretkey!123"; // ключ для шифрации
75     public static SymmetricSecurityKey GetSymmetricSecurityKey() =>
76         new SymmetricSecurityKey(Encoding.UTF8.GetBytes(KEY));
77 }
78
79 record class Person(string Email, string Password);
```

Для представления пользователя в приложении здесь определен record-класс `Person`, который имеет два свойства: `email` и `password`. И для упрощения ситуации вместо базы данных все пользователи приложения хранятся в списке `people`. Условно говоря у нас есть два пользователя.

Для описания некоторых настроек генерации токена, как и в прошлой теме, в коде определен специальный класс **`AuthOptions`**, и также, как и в прошлой теме, с помощью метода **`AddJwtBearer()`** в приложение добавляется конфигурация токена.

В конечной точке `"/login"`, которая обрабатывает POST-запросы, получаем отправленные клиентом аутентификационные данные опять же для простоты в виде объекта `Person`:

```
1 app.MapPost("/login", (Person loginData) =>
```

Используя полученные данные, пытаемся найти в списке `people` пользователя:

```
1 Person? person = people.FirstOrDefault(p => p.Email == loginData.Email && p.Password == loginData.Password);
```

Если пользователь не найден, то есть переданы некорректные `email` и/или `password`, то отправляем статусный код 401, который говорит о том, что доступ запрещен:

```
1 if(person is null) return Results.Unauthorized();
```

Если пользователь найден, то создается список объектов `Claim` с одним `Claim`, который представляет `email` пользователя. Генерируем `jwt`-токен:

```
1 var claims = new List<Claim> {new Claim(ClaimTypes.Name, person.Email) };
2 var jwt = new JwtSecurityToken(
3     issuer: AuthOptions.ISSUER,
4     audience: AuthOptions.AUDIENCE,
5     claims: claims,
6     expires: DateTime.UtcNow.Add(TimeSpan.FromMinutes(2)), // действие токена истекает через 2 минуты
7     signingCredentials: new SigningCredentials(AuthOptions.GetSymmetricSecurityKey(), AuthOptions.ALGORITHM));
8 var encodedJwt = new JwtSecurityTokenHandler().WriteToken(jwt);
```

Далее формирует ответ клиенту. Он отправляется в виде объекта в формате `json`, который содержит два свойства: `access_token` - собственно токен и `username` - `email` аутентифицированного пользователя

```
var response = new { access_token = encodedJwt, username = person.Email }; return Results.Json(response);
```

Еще одна конечная точка - `"/data"` использует атрибут **`Authorize`**, поэтому для обращения к ней необходимо в запросе отправлять полученный `jwt`-токен.

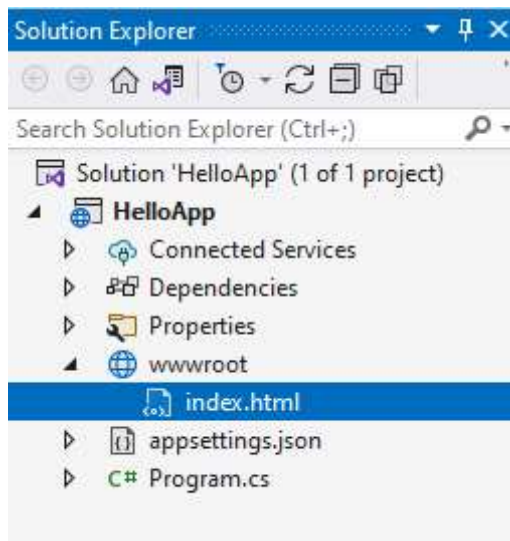
```
1 app.Map("/data", [Authorize] (HttpContext context) => $"Hello World!");
```

Создание клиента на javascript

Теперь определим клиент для тестирования авторизации с помощью токена. Итак, в коде приложения определено подключение статических файлов по умолчанию:

```
1 app.UseDefaultFiles();
2 app.UseStaticFiles();
```

В качестве веб-страницы по умолчанию добавим в проект для статических файлов папку `wwwroot`, а в нее - новый файл **index.html**:



В файле **index.html** определим следующий код:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="utf-8" />
5     <title>METANIT.COM</title>
6 </head>
7 <body>
8     <div id="userInfo" style="display:none;">
9         <p>Добро пожаловать <span id="userName"></span>!</p>
10        <input type="button" value="Выйти" id="logOut" />
11    </div>
12    <div id="loginForm">
13        <h3>Вход на сайт</h3>
14        <p>
15            <label>Введите email</label><br />
16            <input type="email" id="email" />
17        </p>
18        <p>
19            <label>Введите пароль</label><br />
20            <input type="password" id="password" />
21        </p>
22        <input type="submit" id="submitLogin" value="Логин" />
```

```
23     </div>
24     <p>
25         <input type="submit" id="getData" value="Получить данные" />
26     </p>
27     <script>
28         var tokenKey = "accessToken";
29         // при нажатии на кнопку отправки формы идет запрос к /login для получения
30         document.getElementById("submitLogin").addEventListener("click", async e => {
31             e.preventDefault();
32             // отправляет запрос и получаем ответ
33             const response = await fetch("/login", {
34                 method: "POST",
35                 headers: { "Accept": "application/json", "Content-Type": "application/json" },
36                 body: JSON.stringify({
37                     email: document.getElementById("email").value,
38                     password: document.getElementById("password").value
39                 })
40             });
41             // если запрос прошел нормально
42             if (response.ok === true) {
43                 // получаем данные
44                 const data = await response.json();
45                 // изменяем содержимое и видимость блоков на странице
46                 document.getElementById("userName").innerText = data.username;
47                 document.getElementById("userInfo").style.display = "block";
48                 document.getElementById("loginForm").style.display = "none";
49                 // сохраняем в хранилище sessionStorage токен доступа
50                 sessionStorage.setItem(tokenKey, data.access_token);
51             }
52             else // если произошла ошибка, получаем код статуса
53                 console.log("Status: ", response.status);
54         });
55
56         // кнопка для обращения по пути "/data" для получения данных
57         document.getElementById("getData").addEventListener("click", async e => {
58             e.preventDefault();
59             // получаем токен из sessionStorage
60             const token = sessionStorage.getItem(tokenKey);
61             // отправляем запрос к "/data"
62             const response = await fetch("/data", {
63                 method: "GET",
64                 headers: {
65                     "Accept": "application/json",
66                     "Authorization": "Bearer " + token // передача токена в заголовок
67                 }
68             });
69
70             if (response.ok === true) {
71                 const data = await response.json();
```

```

72         alert(data.message);
73     }
74     else
75         console.log("Status: ", response.status);
76 });
77
78 // условный выход - просто удаляем токен и меняем видимость блоков
79 document.getElementById("logOut").addEventListener("click", e => {
80
81     e.preventDefault();
82     document.getElementById("userName").innerText = "";
83     document.getElementById("userInfo").style.display = "none";
84     document.getElementById("loginForm").style.display = "block";
85     sessionStorage.removeItem(tokenKey);
86 });
87 </script>
88 </body>
89 </html>

```

Первый блок на странице выводит информацию о вошедшем пользователе и ссылку для выхода. Второй блок содержит форму для логина.

После нажатия кнопки на форме логина запрос будет отправляться методом POST на адрес `"/login"`. Конечная точка, которая отвечает за обработку POST-запросов по этому маршруту, если переданы корректные email и пароль, отправит в ответ токен.

Ответом сервера в случае удачной аутентификации будет примерно следующий объект:

```

1  {
2      access_token : "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy54bWw...
3                      cy8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9uYW11IjoicXdlcnR5IiwiaHR0cDov...
4                      WlJcm9zb2Z0LmNvbS93cy8yMDA4LzA2L2lkZW50aXR5L2NsYWltcy9yb2x1Ijoic...
5                      I6MTQ4MTYzOTMxMSwiZXhwIjoxNDgxNjM5MzcwLCJpc3MiOiJNeUF1dGhTZXJ2Z...
6                      odHRwOi8vbG9jYXRob3N0OjUxODg0LyJ9.dQJF6pALUZW3wGBANy_tCwk5_NR0TV...
7      username: "tom@gmail.com"
8  }

```

Параметр `access_token` как раз и будет представлять токен доступа. Также в объекте передается дополнительная информация о нике пользователя.

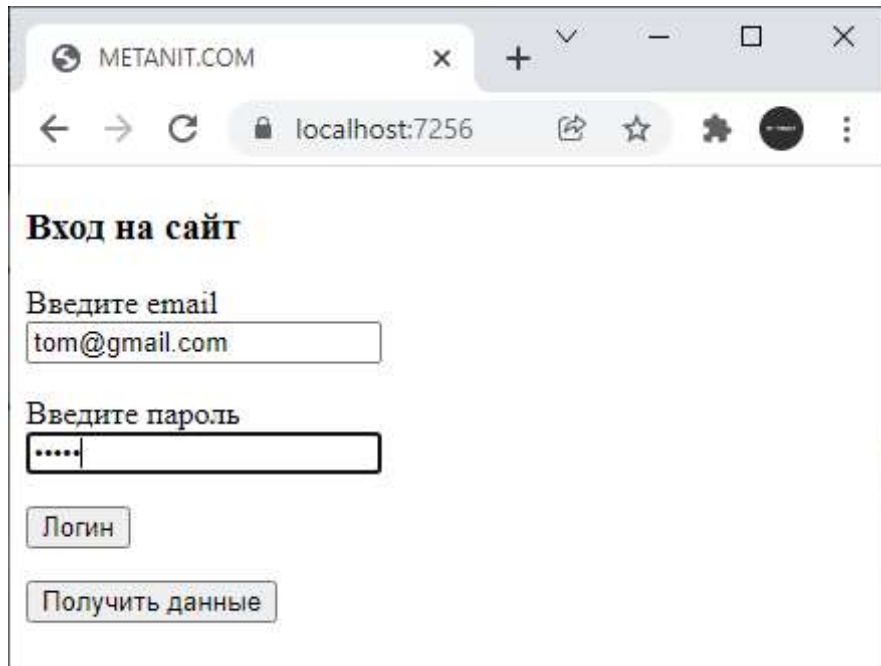
Для того, чтобы в коде js данный токен в дальнейшем был доступен, то он сохраняется в хранилище `sessionStorage`.

Дополнительная кнопка с `id="getData"` на странице предназначена для тестирования авторизации с помощью токена. По ее нажатию будет выполняться запрос по адресу

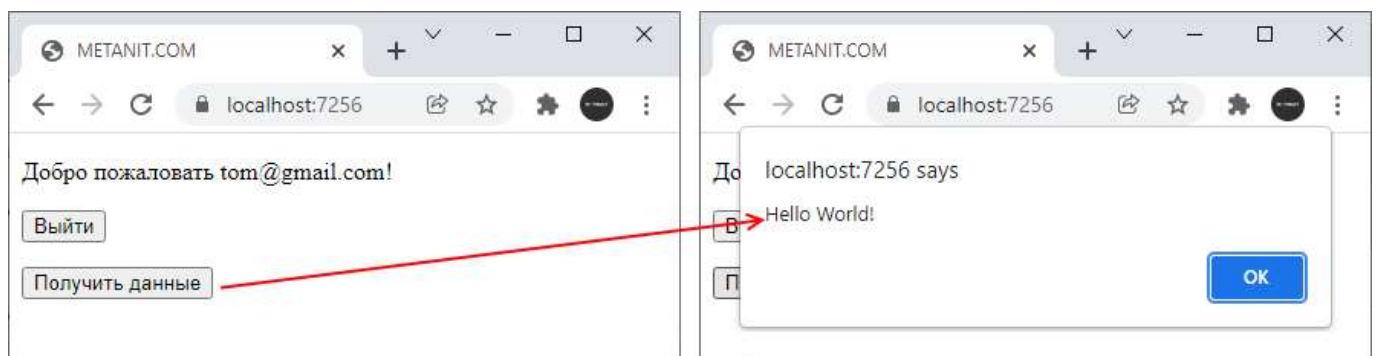
"/data", для доступа к которому необходимо быть аутентифицированным. Чтобы отправить токен в запросе, нам нужно настроить в запросе заголовок Authorization:

```
1 headers: {  
2     "Accept": "application/json",  
3     "Authorization": "Bearer " + token // передача токена в заголовке  
4 }
```

Запустим проект и введем данные одного из пользователя, который есть в списке people:



При вводе корректных данных сервер пришлет клиенту объект с jwt-токеном и логином пользователя. И после этого мы можем нажать на кнопку "Получить данные" и тем самым обратиться к ресурсу "/data", для доступа к которому требуется токен



В то же время если мы попробуем обратиться к этому же ресурсу без токена или с токеном с истекшим сроком, то получим ошибку 401 (Unauthorized).

[Назад](#) [Содержание](#) [Вперед](#)



ALSO ON METANIT.COM

Всплывающие окна

месяц назад • 4 комментариев

Всплывающие окна в .NET MAUI и C#, методы DisplayAlert, ...

ASP.NET и SignalR и C#

25 дней назад • 1 коммента...

Введение в SignalR Core. Первое приложение на SignalR в ASP.NET Core ...

Взаимодействие XAML и C#

2 месяца назад • 2 коммент...

Взаимодействие кода XAML и C# в .NET MAUI, определение логики ...

Созд

2 мес

Введе Multi-кросс

14 Комментариев

metanit.com

🔒 Политика конфиденциальности

Disqus


1 Войти ▾

📖 Favorite 3

🐦 Твитнуть

📌 Поделиться

👍 Лучшее ▾



Присоединиться к обсуждению...

ВОЙТИ С ПОМОЩЬЮ

ИЛИ ЧЕРЕЗ DISQUS ?

Имя

Long Landing • 3 месяца назад

Кто будет пытаться сделать такую авторизацию, обратите внимание на необходимые nuget пакеты:

- 1) Microsoft.AspNetCore.Authentication.JwtBearer
- 2) Microsoft.IdentityModel.Tokens
- 3) System.IdentityModel.Tokens.Jwt

Помощь сайту

YooMoney:
410011174743222

Перевод на карту
Номер карты:
4048415020898850

Номер карты:
4890494751804113

[Вконтакте](#) | [Телеграм](#) | [Twitter](#) | [Канал сайта на youtube](#) | [Помощь сайту](#)

Контакты для связи: metanit22@mail.ru

Copyright © metanit.com, 2012-2022. Все права защищены.