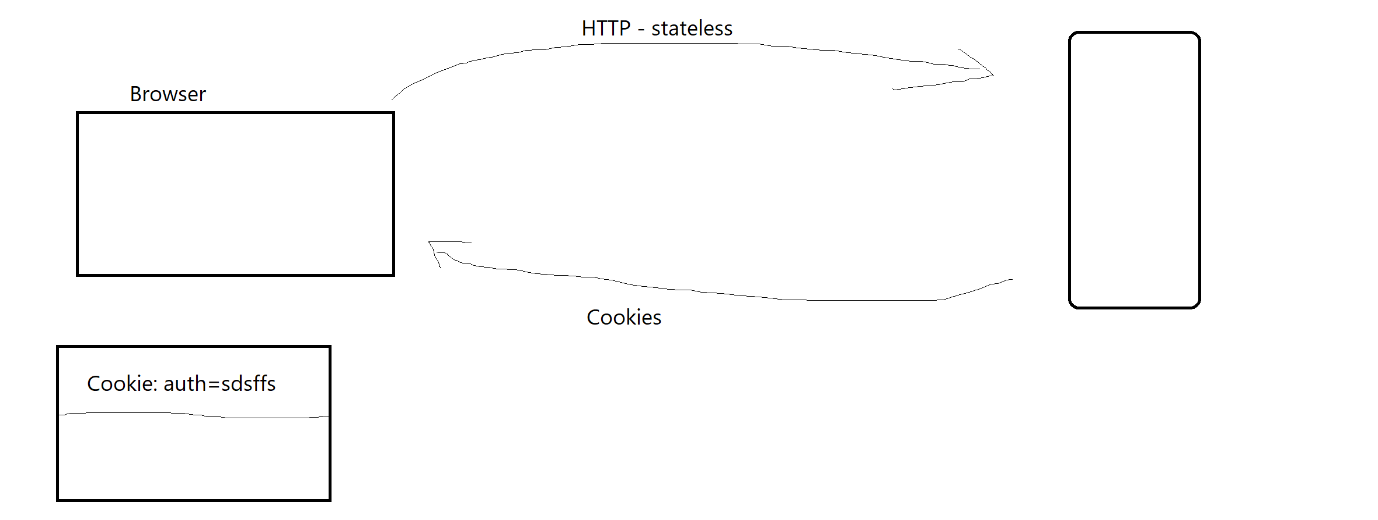
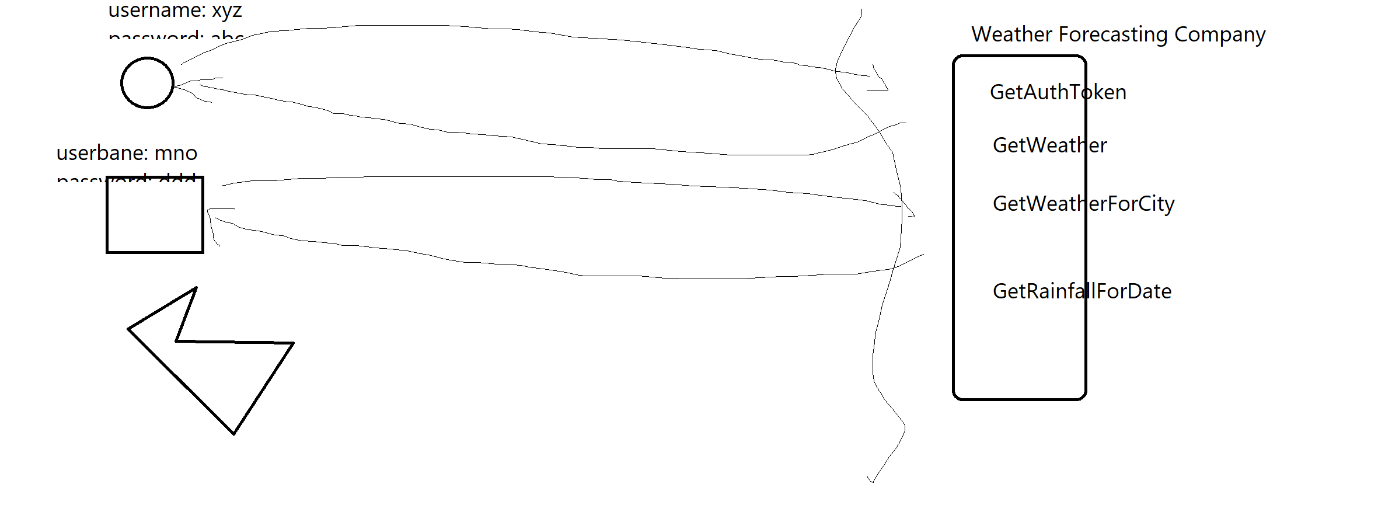
# Authentication with ASP.NET Web Site



# Authentication with ASP.NET Web API



1. Client ask API Server for Token, send parameters username and password
2. API server sends back a Token and its expiry date/time   
   (Token: sdkfl;k143rwurfwe52dfsfsdhyt242rjekfsfjskt242rwebfhwuirwefaj32u432rwfdfiysef, ExpiryDateTime: 5000)
3. Client then includes this token in the API request header for every subsequent request
4. Client should use the same token till it is expired. No need to do step 1 again and again.

## Steps to add authentication in webapi project

1. NUGET Package for Token Management: **Microsoft.IdentityModel.Tokens,   
   System.IdentityModel.Tokens.Jwt**
2. Add the following in appsettings and write the necessary code to inject this value

"JwtSecuritySettings": {

"SecurityKey": "54d6504255f2effe17f74a8b8170e7a8ece0fc79",

"TokenValiditySeconds": 300

}

1. Add Authentication Scheme in program.cs

var securityKey = builder.Configuration.GetValue<string>("JwtSecuritySettings:SecurityKey");

builder.Services.AddAuthentication(options =>

{

options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

})

.AddJwtBearer(options =>

{

options.SaveToken = true;

options.RequireHttpsMetadata = false;

options.TokenValidationParameters = new TokenValidationParameters()

{

ValidateIssuer = false,

ValidateAudience = false,

IssuerSigningKey =

new SymmetricSecurityKey(Encoding.ASCII.GetBytes(@"54d6504255f2effe17f74a8b8170e7a8ece0fc79"))

};

});

1. Enable Authentication and Authorization capabilities in Program.cs

app.UseAuthentication();

app.UseAuthorization();

1. Generate the token as follows

public AuthResponse GetAuthToken(string username, string password)

{

if (username != "abcdef" || password != "mno")

throw new AuthFailedException("Invalid UserName and/or Password");

// Get token and expiryDatetime

var expiryDateTime = DateTime.Now.AddSeconds(\_jwtSecuritySettings.TokenValiditySeconds);

var tokenKey = Encoding.ASCII.GetBytes(\_jwtSecuritySettings.SecurityKey);

var securityTokenDescriptor = new SecurityTokenDescriptor

{

Subject = new ClaimsIdentity(new List<Claim>

{

new("userName", username),

new("expiresIn", expiryDateTime.ToString()),

}),

Expires = expiryDateTime,

SigningCredentials = new SigningCredentials(

new SymmetricSecurityKey(tokenKey),

SecurityAlgorithms.HmacSha256Signature)

};

var jwtSecurityTokenHandler = new JwtSecurityTokenHandler();

var securityToken = jwtSecurityTokenHandler.CreateToken(securityTokenDescriptor);

var token = jwtSecurityTokenHandler.WriteToken(securityToken);

return new AuthResponse {Token = token, ExpiryDateTime = expiryDateTime};

//return new AuthResponse {Token = "asdghfsgsdlhsgshgs", ExpiryDateTime = DateTime.Now.AddMinutes(5)};

}

1. Create an API endpoint /api/AuthToken which accepts username/password and returns Token and ExpiryDatetime.
   1. Specify [AllowAnonymous]
   2. Use the code in step 5 to return the token and ExpiryDateTime from this endpoint
   3. Ensure this endpoint is POST as we don’t want password in url (as in case of GET)
2. For all other endpoints which need to be secured
   1. Specify [Authorize]
3. Test this API from postman as follows
   1. Invoke the POST for /api/AuthToken with username/password in JSON format
   2. Copy the Token from response
   3. Invoke the GET/POST for any other authorized endpoint
      1. Add Authorization header with the token taken from point 8.b

Graphical user interface, text, application

Description automatically generated

* + 1. You can resuse the same token for subsequent api calls till the ExpiryDateTime has passed