CS6004NI
Application Development

Himalay Sunuwar

himalay@innovatetech.co

Working with DB Transaction

```csharp
1.   await using var transaction = await _dbContext.Database.BeginTransactionAsync();
2.
3.   try
4.   {
5.       // Perform database operations here...
6.
7.       await _dbContext.SaveChangesAsync();
8.       await transaction.CommitAsync();
9.   }
10.  catch (Exception ex)
11.  {
12.      await transaction.RollbackAsync();
13.      throw;
14.  }
```

Working with Cookies

```
1.   Response.Cookies.Append("mycookie", "myvalue", new CookieOptions
2.   {
3.       HttpOnly = true, // In the browser, document.cookie (JS) won't be able to read the cookie
4.       SameSite = SameSiteMode.None, // Cookie is sent on cross-site requests (api.example.com, app.example.com)
5.       Secure = true, // Cookie is sent only over HTTPS, it is required when SameSite is None
6.       Path = "/my/path", // Cookie is sent only on specific path prefix
7.       Path = "/", // Cookie is sent on all paths
8.       Domain = "api.example.com", // Cookie is for specific domain
9.       Domain = ".example.com", // Cookie is for cross subdomains
10.  });
11.  // Reading a cookie
12.  var myCookie = Request.Cookies["mycookie"];
13.  // Removing a cookie from browser
14.  if (Request.Cookies["mycookie"] != null)
15.  {
16.      var myCookie = Request.Cookies["mycookie"];
17.      myCookie.Expires = DateTime.Now.AddDays(-1); // Set the Expiry date to past date
18.      Response.Cookies.Add(myCookie); // Update the Cookie in Browser
19.  }
```

Profile Picture

1. **Blazor** Profile Page

```html
<InputFile max-file-size="1536000" accept=".png,.jpg,.jpeg" OnChange="@OnInputFileChange" single />
```

```csharp
1.    private async Task OnInputFileChange(InputFileChangeEventArgs e){
2.        try{
3.            var file = e.File;
4.            var filename = file.Name;
5.            var fileContent = new StreamContent(file.OpenReadStream(maxAllowedSize: 1024 * 1000));
6.            fileContent.Headers.ContentDisposition = new("form-data") { Name = "file", FileName = filename };
7.            fileContent.Headers.ContentType = new MediaTypeHeaderValue(file.ContentType);
8.
9.            using var content = new MultipartFormDataContent();
10.           content.Add(content: fileContent, name: "image");
11.
12.           var response = await AssetService.UploadAssetAsync(content);
13.           ProfilePicturePreview = response?.Url ?? string.Empty;
14.           _userRequest.ProfilePicture = filename;
15.       }catch (Exception ex){
16.           ErrorMessage = ex.Message;
17.       }
18.   }
```

Profile Picture

2. **Blazor** Service Method

```
1.   public async Task<AssetUploadResponse?> UploadAssetAsync(MultipartFormDataContent content){
2.       var response = await _httpClient.AuthPostAsync("/api/assets", content);
3.
4.       await CheckForErrorResponse(response);
5.
6.       var result = await response.Content.ReadFromJsonAsync<AssetUploadResponse>();
7.       return result;
8.   }
```

3. **API** Controller POST Action Method

```
1.   [Authorize]
2.   [HttpPost("/api/assets")]
3.   public async Task<ActionResult<AssetUploadResponse>> UploadFile(IFormFile? file){
4.       if (file == null || file.Length == 0)
5.           return BadRequest("No file selected");
6.
7.       var url = await _assetService.UploadAsync(file);
8.
9.       return Ok(new AssetUploadResponse { Url = url });
10.  }
```

Profile Picture

4. **API** Service Method

```
1.  public async Task<string> UploadAsync(IFormFile file){
2.      var fileName = file.FileName;
3.
4.      if (!IsAllowedFileType(Path.GetExtension(fileName)))
5.          throw new DomainException("Invalid file type");
6.
7.      if (file.Length > 1 * 1024 * 1024) // 1MB
8.          throw new DomainException("File size exceeds the limit");
9.
10.     return await _fileProvider.UploadFileAsync(file, fileName);
11. }
```

5. **API** Provider Method

```
1.  public async Task<string> UploadFileAsync(IFormFile file, string fileName){
2.      var filePath = Path.Combine(UploadDirectory, fileName);
3.      await using var fileStream = new FileStream(filePath, FileMode.Create, FileAccess.Write);
4.      await file.CopyToAsync(fileStream);
5.
6.      return $"https://localhost:5001/api/assets/{fileName}";
7.  }
```

Profile Picture

6. **API** Controller GET Action Method

```
1.   [HttpGet("/api/assets/{fileName}")]
2.   public IActionResult GetAsset(string fileName){
3.       var (file, contentType) = _assetService.GetFileAsStream(fileName);
4.       return File(file, contentType);
5.   }
```

7. **API** Provider Method

```
1.   public (FileStream file, string contentType) GetFileAsStream(string fileName){
2.       var filePath = Path.Combine(UploadDirectory, fileName);
3.       if (!File.Exists(filePath))
4.           throw new DomainException("File not found", 404);
5.
6.       var file = File.OpenRead(filePath);
7.       var contentTypeProvider = new FileExtensionContentTypeProvider();
8.       var contentType = contentTypeProvider.TryGetContentType(filePath, out var type) ? type :
     "application/octet-stream";
9.       return (file, contentType);
10.  }
```

Forgot Password

1. Prerequisites

    A.   Log-in into Gmail with your account

    B.   Navigate to https://security.google.com/settings/security/apppasswords

    C.   In select **Mail** and **Other (custom name)**, give it a name and press **Generate**

    D.   It will give your password

    E.   Update appsetting.json in API project

```
1.    ...// other settings
2.    "App": {
3.        "ApiBaseUrl": "https://localhost:5001/api",
4.        "WebAppBaseUrl": "https://localhost:3001"
5.      },
6.      "GmailCredentials": {
7.        "UserName": "himalay.sunuwar@islingtoncollege.edu.np",
8.        "Password": "passwordFromPreviousSteps"
9.    }
```

Forgot Password

2. **API** Email Provider

```
1.    ...// using statements
2.
3.    namespace BookReview.Api.Infrastructure.Email;
4.    public class GmailEmailProvider : IGmailEmailProvider {
5.        private readonly string _from;
6.        private readonly SmtpClient _client;
7.
8.        public GmailEmailProvider(IConfiguration configuration) {
9.            var userName = configuration.GetSection("GmailCredentials:UserName").Value!;
10.           var password = configuration.GetSection("GmailCredentials:Password").Value!;
11.
12.           _from = userName;
13.           _client = new SmtpClient("smtp.gmail.com", 587) {
14.               Credentials = new NetworkCredential(userName, password),
15.               UseDefaultCredentials = false,
16.               EnableSsl = true
17.           };
18.       }
```

Forgot Password

```
19.     public async Task SendEmailAsync(EmailMessage message) {
20.         var mailMessage = new MailMessage(_from, message.To, message.Subject, message.Body);
21.         foreach (var attachment in message.AttachmentPaths.Select(a => new Attachment(a))) {
22.             mailMessage.Attachments.Add(attachment);
23.         }
24.         await _client.SendMailAsync(mailMessage);
25.     }
26. }
```

3. **API** Email Service Method

```
1.  public async Task SendForgotPasswordEmailAsync(string name, string toEmail, string passwordResetToken) {
2.      var passwordRestUrl = $"{_webAppBaseUrl}/reset-password?token={passwordResetToken}";
3.      var message = new EmailMessage {
4.      Subject = "Password Reset Request",
5.      To = toEmail,
6.      Body = @$"Dear {name},
7.  To reset your password, please click on the following link:
8.  {passwordRestUrl}"
9.      };
10.
11.     await _emailProvider.SendEmailAsync(message);
12. }
```

Forgot Password

4. **API** Auth Service Methods

```
1.   public async Task ForgotPassword(string email) {
2.         var user = await _userManager.FindByEmailAsync(email);
3.         if (user != null) {
4.         var passwordResetToken = await _userManager.GeneratePasswordResetTokenAsync(user);
5.         var token = ToUrlSafeBase64(passwordResetToken);
6.         await _emailService.SendForgotPasswordEmailAsync(user.Name, email, token);
7.         }
8.   }
9.
10.  public async Task ResetPassword(string email, string token, string password) {
11.        var user = await _userManager.FindByEmailAsync(email);
12.        if (user != null) {
13.        var passwordResetToken = FromUrlSafeBase64(token);
14.        var result = await _userManager.ResetPasswordAsync(user, passwordResetToken, password);
15.
16.        ValidateIdentityResult(result);
17.        }
18.  }
```

Forgot Password

```
19.   private void ValidateIdentityResult(IdentityResult result) {
20.         if (result.Succeeded) return;
21.         var errors = result.Errors.Select(x => x.Description);
22.         throw new DomainException(string.Join('\n', errors));
23.   }
24.
25.   private static string ToUrlSafeBase64(string base64String) {
26.         return base64String.Replace('+', '-').Replace('/', '~').Replace('=', '_');
27.   }
28.
29.   private static string FromUrlSafeBase64(string urlSafeBase64String) {
30.         return urlSafeBase64String.Replace('-', '+').Replace('~', '/').Replace('_', '=');
31.   }
```

Forgot Password

Password Reset Request External ➤ Inbox × 🖨 ⧉

himalay.sunuwar@islingtoncollege.... 4:37 PM (6 hours ago) ☆ ↩ ⋮

to me ▾

Dear Seed Admin,

We received a request to reset your password. If you did not initiate this request, please ignore this message.

To reset your password, please click on the following link:

https://localhost:3001/reset-password?token=CfDJ8MYwSn275OBE
ngwozEubhANITMHJeV14WIUBVqimSw6HwIhVumz1Zp8hNrtW0xRA3QPCb3YB
9eQph3wWppHBDz52FM8UWMBTdU2FXDF4Eq0JColh7GBTPeb6o3qZVWXs915P
p2i1ZDYtJjl5gJpuYN/oKWlxtNR60/mG8oEAqNv4G1W765JjDgU2ZKKH0Vuh
B9fZ6pTk/YQZaCjki3n76UeJDrwlyk2WOEX5vkLuuHBA

This link will expire in 24 hours. If you need to reset your password after this time, please initiate a new request.

Thank you,
Book Review

↩ Reply   → Forward

User Registration

1. **API** Auth Service Methods

```csharp
1.   public async Task Register(string name, string email, string password) {
2.         var newUser = new AppUser { Name = name, UserName = email, Email = email };
3.         var result = await _userManager.CreateAsync(newUser, password);
4.         ValidateIdentityResult(result);
5.
6.         await _userManager.AddToRoleAsync(newUser, "User");
7.         var emailConfirmationToken = await _userManager.GenerateEmailConfirmationTokenAsync(newUser);
8.         var token = ToUrlSafeBase64(emailConfirmationToken);
9.         await _emailService.SendEmailConfirmationEmailAsync(name, newUser.Id, email, token);
10.  }
11.
12.  public async Task ConfirmEmail(string token, string userId) {
13.         var user = await _userManager.FindByIdAsync(userId);
14.         var emailConfirmationToken = FromUrlSafeBase64(token);
15.         var result = await _userManager.ConfirmEmailAsync(user, emailConfirmationToken);
16.         ValidateIdentityResult(result);
17.  }
```

JWT Token Authentication

1. **API** Prerequisites

A. Install the necessary NuGet packages:

```
dotnet add package Microsoft.AspNetCore.Authentication.JwtBearer --version 6.0.15
dotnet add package Microsoft.IdentityModel.Tokens --version 6.27.0
```

B. Add JWT settings to appsettings.json

```
1.    "Jwt": {
2.        "Key": "tZXiWWJeqXSwezvUFTDSwMkB$4xyPRpk$zeP^ytBU%FqUi&hVG@nDzMExTDDik%c",
3.        "Issuer": "book-review",
4.        "Audience": "book-review-app"
5.    }
```

C. Configure JWT authentication in the Program.cs

```
1.  var key = Encoding.UTF8.GetBytes(builder.Configuration["Jwt:Key"]);
2.  builder.Services.AddAuthentication(auth => {
3.      auth.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
4.      auth.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
5.  })
```

JWT Token Authentication

```
6.    .AddJwtBearer(options => {
7.        options.RequireHttpsMetadata = false;
8.        options.SaveToken = true;
9.        options.TokenValidationParameters = new TokenValidationParameters {
10.           ValidIssuer = builder.Configuration["Jwt:Issuer"],
11.           ValidAudience = builder.Configuration["Jwt:Audience"],
12.           IssuerSigningKey = new SymmetricSecurityKey(key),
13.           ValidateIssuerSigningKey = true,
14.           ValidateLifetime = false
15.       };
16.   });
17.   builder.Services.AddAuthorization();
```

JWT Token Authentication

2. **API** Token Service

```
1.   using System.IdentityModel.Tokens.Jwt;
2.   using System.Security.Claims;
3.   using System.Text;
4.   using Microsoft.IdentityModel.Tokens;
5.
6.   namespace BookReview.Api.Infrastructure.Identity;
7.   public class TokenService : ITokenService {
8.       private readonly string _key;
9.       private readonly string _issuer;
10.      private readonly string _audience;
11.
12.      public TokenService(IConfiguration configuration) {
13.          _key = configuration.GetSection("JWT:Key").Value!;
14.          _issuer = configuration.GetSection("JWT:Issuer").Value!;
15.          _audience = configuration.GetSection("JWT:Audience").Value!;
16.      }
```

JWT Token Authentication

```
17.        public string GenerateToken(AppUser user, string role) {
18.            var tokenHandler = new JwtSecurityTokenHandler();
19.            var key = Encoding.ASCII.GetBytes(_key);
20.            var tokenDescriptor = new SecurityTokenDescriptor {
21.                Subject = new ClaimsIdentity(new[] {
22.                    new Claim(ClaimTypes.NameIdentifier, user.Id),
23.                    new Claim(ClaimTypes.Email, user.Email),
24.                    new Claim(ClaimTypes.Role, role)
25.                }),
26.                Expires = DateTime.UtcNow.AddHours(12),
27.                Issuer = _issuer,
28.                Audience = _audience,
29.                SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key),
        SecurityAlgorithms.HmacSha256Signature)
30.            };
31.            var securityToken = tokenHandler.CreateToken(tokenDescriptor);
32.            return tokenHandler.WriteToken(securityToken);
33.        }
34.    }
```

JWT Token Authentication

3. **API** Auth Service Method

```csharp
1.  public async Task<string> TokenLoginAsync(string email, string password) {
2.      var user = await _userManager.FindByEmailAsync(email);
3.      if (user == null)
4.          throw new DomainException("Invalid email or password", 401);
5.
6.      var result = await _signInManager.CheckPasswordSignInAsync(user, password, false);
7.      if (!result.Succeeded)
8.          throw new DomainException("Invalid email or password", 401);
9.
10.     var roles = await _userManager.GetRolesAsync(user);
11.     var role = roles.FirstOrDefault();
12.     return _tokenService.GenerateToken(user, role!);
13. }
```

JWT Token Authentication

4. **API** Auth Controller Action Method

```
1.    [HttpPost("/api/auth/login")]
2.    public async Task<IActionResult> LoginAsync([FromBody] LoginRequest login) {
3.        var token = await _authService.TokenLoginAsync(login.Username, login.Password);
4.
5.        return Ok(new TokenLoginResponse {
6.            Token = token
7.        });
8.    }
```

5. **Blazor** Auth Service Methods

```
1.    public async Task LoginAsync(LoginRequest request) {
2.        var response = await _httpClient.AuthPostAsJsonAsync("/api/auth/login", request);
3.
4.        await CheckForErrorResponse(response);
5.
6.        var result = await response.Content.ReadFromJsonAsync<TokenLoginResponse>();
7.        await _jsRuntime.InvokeAsync<string>("localStorage.setItem", "token", result?.Token);
8.    }
```
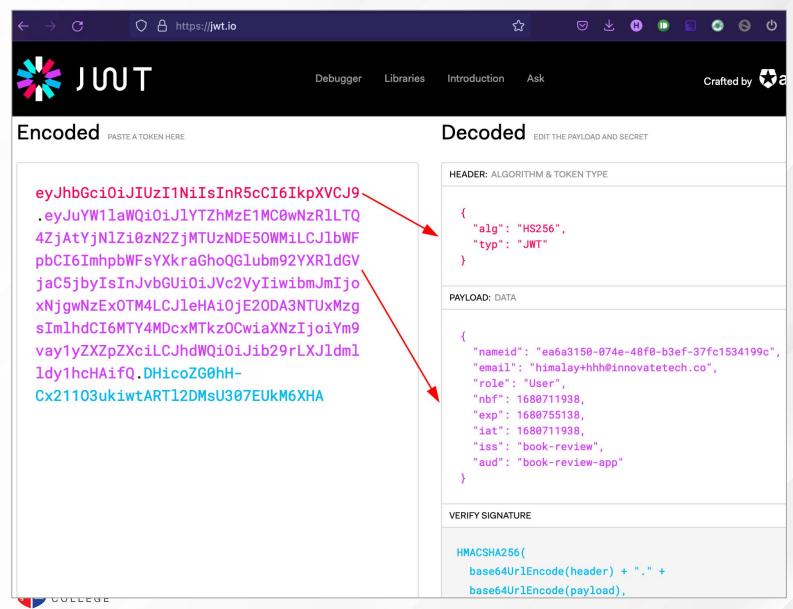
JWT Token Authentication

```
9.    public async Task<UserResponse?> GetProfileAsync() {
10.       var token = await _jsRuntime.InvokeAsync<string>("localStorage.getItem", "token");
11.       _httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", token);
12.       var response = await _httpClient.AuthGetAsync("/api/auth/profile");
13.
14.       await CheckForErrorResponse(response);
15.
16.       var result = await response.Content.ReadFromJsonAsync<UserResponse>();
17.       return result;
18.    }
19.
20.    public async Task LogoutAsync() {
21.       var token = await _jsRuntime.InvokeAsync<string>("localStorage.getItem", TokenKey);
22.       _httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", token);
23.       await _jsRuntime.InvokeVoidAsync("localStorage.removeItem", "token");
24.       var response = await _httpClient.AuthPostAsync("/api/auth/logout", null);
25.       await CheckForErrorResponse(response);
26.    }
```

JWT Token

Thank You