C236 Lesson 09

Answer sheet

# 1. Lesson 09 - Security Part 2

You can find a demonstration of this lesson at the following link: [C236 Lesson 09](http://c236dotnet.azurewebsites.net/lesson09)

# 2. Contents

* [1. Lesson 09 - Security Part 2](#Xd876a27462e0d85b3b7cd478e9d225f1cb27452)
* [2. Contents](#Xcc3d74dea0716fe80c759877aded6871406b368)
* [3. Setup](#Xaf720dab6e1a6fe367187d817bf580bba6dcd9c)
  + [3.1. Setup DB](#X719230e80ea4b0c8214cc75c192ca94f93a892f)
  + [3.2. Setup Application](#X72fd94144bf905b6f7ca05de7b37df5a70a4f90)
* [4. Examine the Code](#X94d8629eaa20e71d33b68250388a252bebae461)
  + [4.1. SQL Injection](#Xec435c5af4f970823990a9a0f7a48d5f0e8b50a)
  + [4.2. Improving Security](#Xb164604351210b0457801cc8b1d8a3c6c7aac42)
  + [4.3. Data Validation](#X51f544cd1bfed1d4598e716ea93a10836775854)
* [5. Solve the Problem](#Xa1c6213ee4f22d2811190b92416d62868f55c03)
  + [5.1. Create New Users](#X370af2492d587af6c2e8d4df3a082d5aafd421c)
  + [5.2. Authorize Roles](#X8410856a7e0f454c222cc6fc7092957a5c04938)
  + [5.3. Test the Security and Safety of the Application](#Xb01e9016e9441af97dd17351882e992bd729f60)

# 3. Setup

## 3.1. Setup DB

1. Open the window SQL Server Object Explorer by clicking **View > SQL Server Object Explorer** and create DB09.
2. **Execute** the SQL statements given in DBSetup.txt against DB09.
3. Right-click on DB09, click **Refresh**. Expand the Tables. You will see the **five** tables: Fruit, Tenancy, TravelHighlight, TravelUser and Warehouse.

## 3.2. Setup Application

1. Set **Lesson09** as the **Startup Project**.
2. Build and **run** the project. You will see **Lesson09 Home** displayed. Stop running the project.
3. From the Models folder in the lesson materials, **copy all files** into the Models folder in Lesson09.
4. From the Views folder in the lesson materials, **copy all the sub-folders and contents** into theViews folder in Lesson09. **Overwrite** any existing files.
5. From the Controllers folder in the lesson materials, **copy all the files** into the Controllers folder.
6. **Drag** the images and photos folders from the lesson materials onto the wwwroot folder in Lesson09.
7. **Drag** the utils folder from the lesson materials onto Lesson09.
8. **Drag** the libman.json file from the lesson materials onto Lesson09. **Restore Client-Side Libraries** from libman.json.
9. Modify the Program.cs file by adding the required globals, authentication and authorization code.

|  |
| --- |
| global using Lesson09.Models; global using RP.SOI.DotNet.Utils; global using System.Data;  using Microsoft.AspNetCore.Authentication.Cookies;  var builder = WebApplication.CreateBuilder(args); builder.Services.AddControllersWithViews(); // authentication builder.Services .AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme) .AddCookie(options => {  options.LoginPath = "/Account/Login";  options.AccessDeniedPath = "/Account/Forbidden"; }); var app = builder.Build(); // Configure the HTTP request pipeline. if (!app.Environment.IsDevelopment()) {  app.UseExceptionHandler("/Home/Error"); } app.UseDefaultFiles(); app.UseStaticFiles(); app.UseAuthentication(); app.UseRouting(); app.UseAuthorization(); app.MapControllerRoute( name: "default", pattern: "{controller=Home}/{action=Index}/{id?}"); app.Run(); |

1. Double click Lesson09 to open the Lesson09.csproj file. Amend the file as shown.

|  |
| --- |
| <Project Sdk="Microsoft.NET.Sdk.Web">   <PropertyGroup>  <TargetFramework>net6.0</TargetFramework>  <Nullable>enable</Nullable>  <ImplicitUsings>enable</ImplicitUsings>  <TreatWarningsAsErrors>true</TreatWarningsAsErrors>  <RootNamespace>Lesson09</RootNamespace>  </PropertyGroup>   <ItemGroup>  <PackageReference Include="System.Data.SqlClient" Version="4.8.3" />  </ItemGroup>  </Project> |

1. Open Appsettings.Development.json (expand Appsettings.json to see the file). Amend the file as shown adding in the ConnectionStrings object.   
   **PDF Users please note:** You will need to type out the DefaultConnection line manually as it cannot be copied from a PDF document and directly pasted into Visual Studio without side-effects.

|  |
| --- |
| {  "Logging": {  "LogLevel": {  "Default": "Information",  "Microsoft.AspNetCore": "Warning"  }  },  "ConnectionStrings": {  "DefaultConnection": "Data Source=(localdb)\\ProjectModels;Initial Catalog=DB09;Integrated Security=True"  } } |

1. Save all files ***Ctrl-Shift-S***.
2. Right-click the project Lesson09 and select **Build**.
3. Add the launch URL to the **IIS Express** (and optionally to **Lesson09**) profiles in launchSettings.json so your project starts with the demonstration application.

|  |
| --- |
| ... "IIS Express": {  "launchUrl": "Injection/Demo",  "commandName": "IISExpress",  "launchBrowser": true,  "environmentVariables": {  "ASPNETCORE\_ENVIRONMENT": "Development"  } }, ... |

1. Run the application to ensure there are no errors.

ONLY CONTINUE WHEN YOUR PROJECT IS ERROR-FREE

# 4. Examine the Code

📌 Where necessary, refer to **c236-lesson09-ln.pdf** as you move through this worksheet.

## 4.1. SQL Injection

1. Run your project, add the controller **Injection** and action **Demo** to your browser's URL; or you can modify the line "launchUrl": "Injection/Demo", in your launchSettings.json file so that you need not add the controller and action over again.
2. Perform the following tasks.
   1. Add "Banana" using the button. [Unsafe]
   2. Add "Pear" using the button. [Safe A]
   3. Add "Pineapple" using the button. [Safe B]
   4. Were three fruits added to the database table?

Yes

1. Use the SQL Injection technique to delete all records in the Fruit table. ‘): DELETE FROM Fruit --
2. Use the SQL Injection technique to drop the Fruit table.
   1. Confirm the table was dropped.
      1. Copy the relevant portion of SQL code from the DBSetup.txt file and recreate the Fruit table and data.
3. Open InjectionController.cs and Views/Injection/Demo.cshtml and examine the code. Complete the following table. The first record has been in *italics* has been created for you.

|  |  |  |  |
| --- | --- | --- | --- |
| Buttons | [Unsafe] | [Safe A] | [Safe B] |
| Controller/Action | Injection/Insecure | Injection/Secure | Injection/Secure2 |
| Secure DB Code? | *No* | Yes | Yes |
|  |  |  |  |
| Reason | Escquote is not ran | EscQuote is ran from the execsql method | EscQuote is ran from the execsql method |

## 4.2. Improving Security

1. Run your project, add the controller **Account** and action **Login** to your browser's URL; or you can modify the line "launchUrl": "Account/Login", in your launchSettings.json file so that you need not add the controller and action over again.
   1. Log into John's account using his id and his password.
   2. Log out.
   3. Try to log into John's account again, but this time, armed with your knowledge of SQL injection, try to do so without his password.
2. Open AccountController.cs, perform **Task 1** so that the security vulnerability in **Account/Login** can be patched.
3. Open TravelController.cs, perform **Task 2** to change all insecure database access to secure database access.

## 4.3. Data Validation

1. Run your project, add the controller **Validation** and action **Demo** to your browser's URL; or you can modify the line "launchUrl": "Validation/Demo", in your launchSettings.json file so that you need not add the controller and action over again.
   1. Examine the error messages in detail.
   2. Change values. In particular examine what occurs when Date B occurs before Date A.
   3. Clear the form then try to resubmit. Note all the error messages and the reason in the table below.

|  |  |  |
| --- | --- | --- |
| Field | Error Message | Reason |
| DateFieldA | Please enter Date A. | Field is mandatory [Required] |
| DateFieldB | Please enter Date B. | dd/mm/yyyy is not a valid date |
| DateFieldB | Date B must be after Date A | Date B must be later than Date A [Required] |
| EmailField | Please enter email | Email is required |
| Teenager | Values from thirteen to nineteen only. | Age range only from 13-19 |
| ProductCode | Please enter Product Code | Product code is required |
| ProductCode | 6-12 characters only. | Product code must range only from 6-12 characters. |

1. Open the file ValidationController.cs, DemoData.cs and Views\Validation\Demo.cshtml. Examine the code, then answer the following questions.
   1. In ValidationController.cs, there are two methods (actions) with the name Demo(). Which one displays the initial default web page to accept user input? Which one handles the submission of the user input? public IActionResult Demo() for part 1 and part 2 is public IActionResult Demo(DemoData \_)
   2. In DemoData.cs:
      1. What is the data type of DateFieldA and DateFieldB? [DataType(DataType.Date)]
      2. Look at the [DataType] attributes. What is the difference between DataFieldA and DateFieldB? DateFieldB has [DateGreaterThan("DateFieldA", ErrorMessage = "Date B must be after Date A.")] indicating that B needs to be greater than A.
      3. For the ProductCode property, is the [Required] attribute really necessary, since there is already a [StringLength] attribute? Yes.
   3. In Views\Validation\Demo.cshtml:
      1. To which action does the form's data flow? Validation/Demo
      2. Which tag helpers are responsible for the form's data flow? asp-controller=”Validation” asp-action=”Demo”
      3. Which tag helper is responsible to generate the html elements to display a single error messages? asp-validation-for

# 5. Solve the Problem

## 5.1. Create New Users

1. Open the views Users.cshtml and CreateUser.cshtml in Views\Admin, then complete **Lesson09 Task 3, 4, 5**.

## 5.2. Authorize Roles

1. Complete **Lesson09 Task 6, 7**.

## 5.3. Test the Security and Safety of the Application

1. Run your project, add the controller **Travel** and action **MyTrips** to your Browser's URL. If your application is working correctly, you be redirected to the **/Account/Login** page. If this does not happen, there is something wrong about the authorization. Check you have the correct [Authorise] attributes specified.
2. Login as **peter**, update your Browser's URL to **Admin/CreateUser**. You will be redirected to **Account/Forbidden** page.
3. Login as **pauline**, update your Browser's URL to **Admin/Users**. You will be redirected to **Account/Forbidden** page.
4. Login as **benny** (admin). Test to see if you can navigate to all the administrator links in the navigation bar.
5. Test if you can log into any user's account using **SQL Injection** without supplying his/her password.

End of Worksheet