# Lesson 4 – Developing data-driven applications

# Setup DB

1. Open the window **SQL Server Object Explorer** by clicking **View** > **SQL Server Object Explorer**. The following window will appear

Graphical user interface, text, application, email

Description automatically generated

1. Expand **(localdb)\ProjectsModels**. Do **not** expand MSSQLLocalDB.
2. If you cannot find it, click on the icon **Add SQL Server…**  Expand **Local** and select **ProjectModels**. Click on the [Connect] button.
3. Right-click **Databases** and click **Add New Database**. The **Create Database** window will appear.

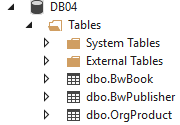
Graphical user interface, text, application

Description automatically generated

1. In the **Database Name** enter **DB04**
2. In the **Database Location:** , select the folder **Databases** in your C236 folder which you have created in Lesson 01.
3. Right-click on **DB04**, click **New Query…** The **SQLQuery1.sql** window will appear
4. Copy the SQL statements given in **DBSetup.txt** and paste inside SQLQuery1.sql
5. Click the **green** execute button at the top left corner of the SQLQuery1.sql window to execute the SQL statements.



1. Right-click on **DB04**, click **Refresh** . Expand the **Tables**. You will see the three tables that have been created: **BwBook, BwPublisher** and **OrgProduct.**

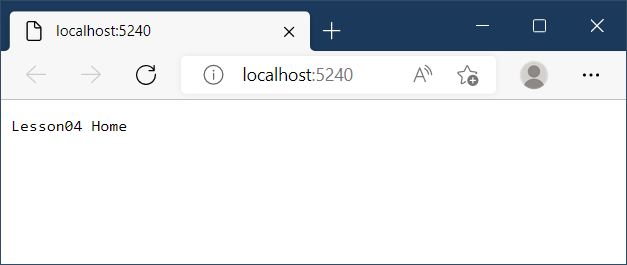


1. Close the **SQLQuery1.sql** window. You do not need to save any changes.
2. You can right-click on any of the tables **BwBook**, **BwPublisher** and **OrgProduct** and click **View Data**. Complete the following table.

|  |  |  |
| --- | --- | --- |
| **Database Table** | **Number of Columns** | **Number of Rows (Records)** |
| OrgProduct | 5 | 7 |
| BwBook | 7 | 28 |
| BwPublisher | 3 | 5 |

# Setup Application

1. In the Solution Explorer, right-click on the Project **Lesson04** > Set as Startup Project.
2. Right-click **Lesson04** and select **Build**
3. Run the project.



1. From the **Models** folder inthe lesson materials, **copy all files** into the **Models** folder in VS2022 Lesson04. Do not drag in the whole folder, or you will create Lesson04/Models/Models, which will not work.
2. From the **Views** folder in the lesson materials, **copy all the sub-folders and contents** into the **Views** folder in VS2022 Lesson04. Do not drag in the whole folder.
3. From the **Controllers** folder in the lesson materials, copy **all the files** into the **Controllers** folder in VS2022 Lesson04.
4. Drag the **images folder** from the lesson materials **onto the** **wwwroot** folder in VS2022 Lesson 04.
5. Drag the **utils folder** from the lesson materials **onto** **Lesson04** in VS2022 Lesson 04.
6. Drag the **libman.json file** from the lesson materials **onto** **Lesson04** in VS2022.. Right-click **libman.json > Restore Client-Side Libraries**. This step creates **wwwroot/lib** folder structure containing the necessary bootstrap libraries.
7. Modify the **Program.cs** file by adding the following line to the very top of the file, then save the file.

global using Lesson04.Models;

global using System.Data;

global using RP.SOI.DotNet.Utils;

1. Double click Lesson04 to open the Lesson04.csproj file. Amend the file as shown.

|  |
| --- |
| <Project Sdk="Microsoft.NET.Sdk.Web">  <PropertyGroup>  <TargetFramework>net6.0</TargetFramework>  <Nullable>enable</Nullable>  <ImplicitUsings>enable</ImplicitUsings>  </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.

|  |
| --- |
| {  "Logging": {  "LogLevel": {  "Default": "Information",  "Microsoft.AspNetCore": "Warning"  }  },  "ConnectionStrings": {  "DefaultConnection": "Data Source=(localdb)\\ProjectModels;Initial Catalog=DB04;Integrated Security=True",  "ProductionConnection": "Server=tcp:sqlsoirp.database.windows.net,1433;Database=dbsoirp;UserID=<userid>;Password=<pw>;Encrypt=True;TrustServerCertificate=False;Connection Timeout=45;"  }  } |

1. Right-click the project **Lesson04** and select **Build**. Ensure there are no errors.
2. Run the project with debugging.

Graphical user interface, text, application

Description automatically generated

CONTINUE ONLY IF YOUR PROJECT IS ERROR-FREE.

# Apprentice Activity – Create HTML Tables

1. Open **DemoController.cs** and examine the actions **Country**, **CountryBS** and **PolytechnicBS**. Fill in the following table with the full filenames of the corresponding views. (BS=BootStrap)

|  |  |
| --- | --- |
| **Actions** | **Views** |
| Demo/Country | Views/Demo/Country.cshtml |
| Demo/CountryBS | Views/Demo/CountryBS.cshtml |
| Demo/PolytechnicBS | Views/Demo/PolytechnicBS.cshtml |

1. **Create the three views** which you specified in the above table. Write html code for the three cshtml files you have created. Remember to **<link>** the Bootstrap library for the second and third views.
2. Run your project, add the controller (**demo**) and the following actions (**Country**, **CountryBS**, **PolytechnicBS**) to your Browser's URL to see the outputs.
3. Compare your outputs from with the following url's

<http://c236dotnet.azurewebsites.net/Lesson04/Demo/Country>

<http://c236dotnet.azurewebsites.net/Lesson04/Demo/CountryBS>

<http://c236dotnet.azurewebsites.net/Lesson04/Demo/PolytechnicBS>

# Apprentice Activity – Experiment with Razor

1. Double click on the file **launchSettings.json** in Lesson04/Properties folder. Add the highlighted line **after** the line the contains **IISExpress** (Line 19). Save the file.

**Note:** If you run your project from **Lesson04**, rather than from IIS, you will also need to change the **Lesson04** profile.

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

1. Run your project to see the output. You will notice that the controller **Demo** and the action **RazorLoopAndIf** have been added to your Browser's URL.

Table

Description automatically generated

1. Open **DemoController.cs** and examine the action **RazorLoopAndIf**. Answer the following questions

|  |  |  |
| --- | --- | --- |
| a. | What is the name of the List? | candList |
| b. | What is the class of each element in the List? | Candidate |
| c. | How many elements are stored in the List? | 6 |
| d. | For each element, how many properties are there? | 4 |
| e. | Who is the only female candidate? | Dewi Aisha |
| f. | Write down the name of the View that this action is calling? | RazorLoopAndIf.cshtml |
| g. | What is passed to the View as a model? | candList |

1. Open **RazorLoopAndIf.cshtml** and examine the code. Perform the following modifcations.
2. Do not display the gender icons, instead display "Man" or "Woman"
3. If the candidate is cleared, display "Cleared", otherwise display nothing.
4. Highlight candidate if the RegNo is odd.
5. After the above modifications, your output will look like the following image.

Table

Description automatically generated

# Apprentice Activity – Accessing a database

1. Run your project, add the controller (**demo**) and action (**DoSelect**) to your Browser's URL to see the output. Enter the following SQL statements and click the button [Submit].

SELECT \* FROM BwPublisher

SELECT \* FROM NoSuchTable

SELECT \* FROM BwBook

1. Open **DemoController.cs** and examine the action **DoSelectPost**. Answer the following questions

|  |  |  |
| --- | --- | --- |
| a. | How the user-entered SQL statement is read by the action? | HttpContext.Request.Form[“sql”].ToString().toIpper().Trim() |
| b. | What is the DBUtl method used to run the SQL statement? | DBUtil.Get\_Table(sql) |
| c. | What is the purpose of the DataTable dt object? | Store data rows that are returned after executing SQL statement |
| d. | What happens when there is an error in the SQL statement? | Database error message will be retrieved from the DBUtil.DB\_Message and stored in ViewData[“Message”]. Pass database error message from action to view. |
| e. | Does the action pass the data to the view using ViewData or Model? | Model  return View(“DoSelect”,dt) |

# Organic Fruits

1. The Organic Fruits' Products is a **sample application** provided to you so that you can understand how to access a database and how records can be displayed in a Web Application. Run your project and add the controller (**organic**) and action (**products**) to your Browser's URL to see the output.
2. The code for Organic Fruits are found in the following files

Controllers/OrganicController.cs

Views/Organic/Products.cshtml

# Solving the Problem

1. As a team, discuss and create SQL SELECT statements to answer Rachel's queries. Remember to Right-click the database in SQL Server Object Explorer then select **New Query...** to write your SQL. Once you have obtained the result you expect, cut and paste appropriately.

Q1: What are the non-English titles?

|  |
| --- |
| SELECT Isbn, Title, Lang  WHERE Lang != “ENGLISH”; |

Q2: Which titles are out of stock?

|  |
| --- |
| SELECT Isbn, Title, Qty  WHERE Qty=0; |

Q3: How many titles are there for each publisher?

|  |
| --- |
| SELECT P Publisher, COUNT(\*) AS Title  FROM BwBook BP  INNER JOIN BwBook BP ON BP.PubID = P.PubID  GROUP BY P.Publisher |

Q4: What are the titles that have more than one language?

|  |
| --- |
| SELECT Title  FROM BwBook  GROUP by Title  HAVING COUNT(Lang)>1 |

1. To solve Rachel's problem. You need to create the following files and folders.

* Controllers/**BookWormController.cs**
* Views/**BookWorm**/**Query.cshtml**

1. You can visit <http://c236dotnet.azurewebsites.net/Lesson04/BookWorm/Query> and perform a "View Source" to reuse the html code to help you with coding the **Query.cshtml**. Remember to **remove** the virtual directory "**Lesson04"** from your code. Also, you can reuse the code segment in **DoSelect.cshtml** to display the results of the SQL query using the **<table>** element.
2. The code for the **BookWormController.cs** is given below.

|  |
| --- |
| const string SELECT1 = // QUESTION ONE  @"SELECT … ";  const string SELECT2 = // QUESTION TWO  @"SELECT … ";  const string SELECT3 = // QUESTION THREE  @"SELECT … ";  const string SELECT4 = // QUESTION FOUR  @"SELECT … ";  public IActionResult Query()  {  return View();  }  public IActionResult Submit()  {  IFormCollection form = HttpContext.Request.Form;  string question = form["\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"].ToString();  string sql = "";  if (question.Equals("1"))  {  sql = SELECT1;  }  else if (question.Equals("2"))  {  sql = SELECT2;  }  else if (question.Equals("3"))  {  sql = SELECT3;  }  else if (question.Equals("4"))  {  sql = SELECT4;  }  DataTable dt = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;  return View(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_);  }  } |

*— End of Worksheet —*