# Lab: ASP.NET MVC with DB

his document defines several walkthroughs for creating ASP.NET MVC-based apps, from setting up the framework to implementing the fully functional applications.

## Application with SoftUni Database

Create a Web application, which **displays and modifies** the data from table Towns in the **SoftUni Database**. The app should look like this:

Graphical user interface, text, application

Description automatically generated

We are going to **use** the existing SoftUni database. **Start** your Microsoft SQL Management Studio.

### Create a New ASP.NET MVC Project

Open **Visual Studio** and create a C# web project using the **ASP.NET Web Application (.NET Core)** template.

Graphical user interface, text, application, email

Description automatically generated

In the "**Create a new ASP.NET Core Web Application"** window, choose **[Model-View-Controller (MVC)]**.

Graphical user interface, application, Teams

Description automatically generated

Graphical user interface, text, application

Description automatically generated

### Install Entity Framework Core

Go to [Tools] 🡪 [NuGet Package Manager] 🡪 [Package Manager Console] and **run** the following commands individually:

|  |
| --- |
| Install-Package Microsoft.EntityFrameworkCore.Tools |
| Install-Package Microsoft.EntityFrameworkCore.SqlServer.Design |
| Install-Package Microsoft.EntityFrameworkCore.SqlServer |

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

### Create Model and Database Context

In the **Package Manager Console run** the following command:

|  |
| --- |
| Scaffold-DbContext -Connection "Server=(localdb)\MSSQLLocalDB;Database=SoftUni;Integrated Security=True;" -Provider Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models |

Graphical user interface, text, application

Description automatically generated

### Scaffold Controller with Views

**Create an ASP.NET Core MVC Controller** for handling data and performing basic CRUD operations.

Right-click on the **Controllers** folder in Solution Explorer and select [Add] 🡪 [Controller]:

Graphical user interface, application

Description automatically generated

Select **[MVC Controller with Views using Entity Framework]**.

Graphical user interface, application

Description automatically generated

* Select Town as **Model class**
* SoftUniContext as the **Data Context class**
* Tick the [Generate Views]option
* In the "**Select a Layout Page**", browse through Views 🡪 Shared 🡪 \_Layout.cshtml
* Give a good name for the controller class, e.g. TownsController

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Now, you will have generated a new **controller** class, with a few **views** in the code:

Graphical user interface, text, application, chat or text message

Description automatically generated

**Register the DB Context Class in the App Configuration**

**Register** your configuration class at the ConfigureServices()method in Startup.cs, by adding:

|  |
| --- |
| services.AddDbContext<SoftUniContext>(); |

Graphical user interface, text, application, email

Description automatically generated

### Display the “Towns” Page in the Menu

To **display the “**Towns**” page** in the app, you need to add it to the layout view:

In Solution Explorer go to: Views 🡪 Shared 🡪 \_Layout.cshtml.

In the **<ul class="navbar-nav flex-grow-1">** **add new** <li> **element**:

|  |
| --- |
| <li class="nav-item">  <a class="nav-link text-dark" asp-area="" asp-controller="Towns" asp-action="">Towns</a>  </li> |

This is how it looks like:

Graphical user interface, text, application

Description automatically generated

### Run and Test the App

Now **build** and **run** the application using **[Ctrl] + [F5]**.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

## TODO List

Create a “**TODO List**” Web application, which keeps track of a person’s **tasks** inside a **database**. The application should support **creating** tasks and **deleting** tasks.

Graphical user interface, text, application, email

Description automatically generated

### Create the Database

Create **database** named **"**ToDoDB**"** in **Microsoft SQL Management Studio**. It will have a ToDoTasks **table** with **2 columns**:

* Id– a unique **integer**, with which to differentiate tasks from one another.
* Title – the **title** of the task, stored as a **string**.

Table

Description automatically generated

### Insert Some Sample Data

Create 3 tasks in the table:

|  |
| --- |
| **Tasks** |
| Water the flowers. |
| Buy cat food. |
| Make the project for Wednesday. |

Graphical user interface, text, application, email

Description automatically generated

### Create a New Project

Create new project and name it TODOTasksList.

Open **Visual Studio** and create a C# web project using the **ASP.NET Web Application (.NET Core)** template.

In the Create a new ASP.NET Core Web Application window, choose Model-View-Controller (MVC).

**Integrating Entity Framework Core:**

Go to [Tools]🡪 [NuGet Package Manager]🡪 [Package Manager Console] and **run** the following commands individually:

|  |
| --- |
| Install-Package Microsoft.EntityFrameworkCore.Tools |
| Install-Package Microsoft.EntityFrameworkCore.SqlServer.Design |
| Install-Package Microsoft.EntityFrameworkCore.SqlServer |

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

### Create Model and Database Context

In the **Package Manager Console** run the following command:

|  |
| --- |
| Scaffold-DbContext -Connection "Server=(localdb)\MSSQLLocalDB;Database=ToDoDB;Integrated Security=True;" -Provider Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models |

Graphical user interface, text, application

Description automatically generated

### Scaffold Controller with Views

**Create an ASP.NET Core MVC Controller** for handling data and performing basic CRUD operations.

Right-click on the **Controllers** **folder** in Solution Explorer and select [**Add**] > [**Controller**]

Select **MVC Controller with Views using Entity Framework**.

* Select ToDoTasks as **Model class**
* ToDoDBContext as the **Data Context class**
* Tick the [Generate Views]option
* In the "**Select a Layout Page**", browse through Views 🡪 Shared 🡪 \_Layout.cshtml

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

### Register the DbContext Class in the App Config

**Register** your configuration class at the **ConfigureServices()** method in **Startup.cs**, by adding:

|  |
| --- |
| services.AddDbContext<ToDoDBContext>(); |

**Graphical user interface, text, application, email

Description automatically generated**

### Display the “Tasks” Page

To **display the page** in the app, you need to add it to the title page.

In **Solution Explorer** go to: Views 🡪 Shared 🡪 \_Layout.cshtml.

In the <ul class="navbar-nav flex-grow-1"> add **new** <li> **element**:

|  |
| --- |
| <li class="nav-item">  <a class="nav-link text-dark" asp-area="" asp-controller="ToDoTasks" asp-action="Index">ToDo</a>  </li> |

Text

Description automatically generated

### Change the App Details

Change the **title** and **link**. In \_Layout.cshtmlchange the text of the **first** <a> **element** to TODO List and the controller to lead to ToDoTasks. **Delete** the<li class="nav-item"> that gives the[Privacy] **button** in the navigation bar.



Change the index page. In **Solution Explorer** go to: Views 🡪 Home 🡪 Index.cshtml.Change the **paragraph text** to: "You are about to experience my TODO List." and make the **last two words link** which leads to TODO Page.

Logo, company name

Description automatically generated with medium confidence

Change the TODO Page. In **Solution Explorer** go to: Views 🡪 ToDoTasks🡪 Index.cshtml**.** Change the heading to "To Do Tasks":

Graphical user interface, text, application

Description automatically generated

### Run and Test the App

Now **build** and **run** the application using **[Ctrl] + [F5]**.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated