CONTENTS

PAGE

ABSTRACT

Working on a template we found using ASP.NET Core MVC, Model-View-Controller (MVC) design model. When we created the project, we started our EntityFramework Project by first installing this tamplate. We would add controller and then we created wiev. We took the code from the html file in the template file and added it to the view section, that is, to the cshtml file. Then we created the table classes and we created the Context class. We created DbSet<> of lists in Context class. We created the database lists for the controller and added the connectionStrings part to the Web.config. Then we created the migrations and made an update database. We have created tables in the database section. Finally, we added the data in our tables to our web page with the foreach method in the cshtlm section.

Keywords:

ASP.NET Core MVC; EntityFramework;update Database;create table.

1. INTRODUCTION

Our topic is to find an HTML template and add it to ASP.NET MVC and pulling any part of this template from the database. Our aim is to create the project, load the template we found in it, create the controller part, add the view, add the code in html to cshtml, create the classes, add the connectionStrings part in the web.config and create a table in the database part. Then it is to pull data from there to tamplate. Thanks to ASP.NET MVC. By using Net Framework languages and MVC Pattern, it is possible to develop fast working, testable and reusable web applications.

2. CODES AND OUTPUT

2.1. DefaultController

We created the DefaultController and added the view. then we created the lists in this class. Since we could not return 3 lists separately, we created a CombinedViewModel class that could view all of them, and we created the get, set methods of the lites here.

```
using hw_1.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace hw_1.Controllers
{
    public class DefaultController : Controller
    {
        // GET: Default
        Context db = new Context();
        public ActionResult Index()
        {

            List<Starters> Starterslist = db.Starter.ToList();
            List<MainCourses> MainCourseslist = db.MainCourse.ToList();
            List<Desserts> Dessertslist = db.Dessert.ToList();

            // Combine the lists into a single model object
            var combinedModel = new CombinedViewModel
            {
                Starter = Starterslist,
                MainCourse = MainCourseslist,
                Dessert = Dessertslist
            };

            return View(combinedModel);
        }

    }
}
public class CombinedViewModel
{
    public List<Starters> Starter { get; set; }
    public List<MainCourses> MainCourse { get; set; }
    public List<Desserts> Dessert { get; set; }

}
```

Figure1: DefaultController Code

2.2. index.cshtml

First, we added @using hw_1.Models and @model CombinedViewModel to the cshtml part. Then we created the @foreach methods.

```
@using hw_1.Models
@model CombinedViewModel
@{
    Layout = null;
}
```

Figure2: index.cshtml Code-1

```
</div>
<h1><span class="menu-icon">___MENU </span></h1>
<table border=3px solid; style="background-color: white;">
    <tr style="height:50px">
        <th width="500px" style="font-size: 20px;"><b>Name</b></th>
        <th width="500px" style="font-size: 20px;"><b>Price </b></th>
        <h2>Starters</h2>
        @foreach (var item in Model.Starter)
        {

            <table border=3px solid; style="background-color: lightgray;">
                <tr style="height:50px">
                    <th width="500px" style="font-size: 20px;">@item.StarterName</th>
                    <th width="500px" style="font-size: 20px;">@item.Price</th>

                </tr>
            </table>

        }


        <div id="Main" class="tm-tab-content">
            <h2>Main Courses </h2>

            @foreach (var item in Model.MainCourse)
            {
                <table border=3px solid; style="background-color: lightgray;">
                    <tr style="height:50px">
                        <th width="500px" style="font-size: 18px;">@item.MainName</th>
                        <th width="500px" style="font-size: 18px;">@item.Price</th>

                    </tr>
                </table>

            }


        </div>
```

Figure3: index.cshtml Code-2

```
        <div id="Dessert" class="tm-tab-content">
            <h2>Desserts</h2>

            @foreach (var item in Model.Dessert)
            {
                <table border=3px solid; style="background-color: lightgray;">
                    <tr style="height:50px">
                        <th width="500px" style="font-size: 20px;">@item.DessertName</th>
                        <th width="500px" style="font-size: 20px;">@item.Price</th>

                    </tr>
                </table>

            }
        </tr>
    </table>

</div>
```

Figure4: index.cshtml Code-3

2.3. Starters Class

We created StarterId, StarterName, Price.

```
namespace hw_1.Models
{
    public class Starters
    {
        [Key]

        public int StarterId { get; set; }
        public string StarterName { get; set;}
        public string Price{ get; set;}
    }
}
```

Figure5: Starters Class Code

2.4. MainCourses Class

We created MainId, MainName, Price.

```
    public class MainCourses
    {
        [Key]

        public int MainId { get; set; }
        public string MainName { get; set; }
        public string Price { get; set; }
    }
}
```

Figure6: MainCourses Class Code

2.5. Desserts Class

We created DessertId, DessertName, Price.

```
public class Desserts
{
    [Key]

    public int DessertId { get; set; }
    public string DessertName { get; set; }
    public string Price { get; set; }
}
```

Figure7: Desserts Class Code

2.6. Context:DbContext Class

We created the DbSet<> of the Starters, MainCourses and Desserts classes.

```
public class Context:DbContext
{
    public DbSet<Starters> Starter { get; set; }

    public DbSet<MainCourses> MainCourse { get; set; }

    public DbSet<Desserts> Dessert { get; set; }

}
```

Figure8: Context:DbContext Class Code

2.7. connectionStrings

The project is the part where we create the database.

```
<connectionStrings>
    <add name="Context"
        connectionString="Data Source=DESKTOP-SHPA6VN\MSSQLSERVER01; Initial Catalog=DbRestaurant;
        Integrated Security=True" providerName="System.Data.SqlClient"/>
</connectionStrings>
```

Figure9: connectionStrings Code

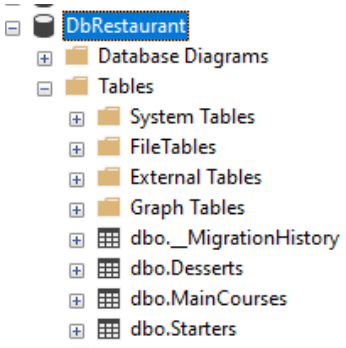

Figure10: Created DbRestaurant database

2.8. Output

Figure11: Output-1

Figure12: Output-2

3. SOURCE

- https://www.w3schools.com/w3css/defaulT.asp