ASP.NET MVC – wprowadzenie

1. Otwórz ***Microsoft Visual Studio***.
2. Wybierz **File->*New->Project->C# ->Web -> ASP.NET Core Web App (Razor Pages)***.
3. Aplikację nazwij***: Moja1***.
4. Zmodyfikuj kod widoku **Pages**-> ***Index.cshtml*** według wzoru (wypróbuj działanie wszystkich znanych znaczników HTML):

```
@page
@model IndexModel
@{
    ViewData["Title"] = "Home page";
}
<div>
    <b>To jest b</b>
    <br />
    <u>To jest u</u>
    <br />
    <i>To jest i</i>
    <br />
    <h1> To jest h1</h1>
    <h2> To jest h2</h2>
    <h3> To jest h3</h3>
    <ul>
        <li>1</li>
        <li>2</li>
        <li>3</li>
        <li>4</li>
    </ul>
    <ol>
        <li>1</li>
        <li>2</li>
        <li>3</li>
        <li>4</li>
    </ol>

</div>
```

Style I

1. Otwórz ***Microsoft Visual Studio***.
2. Wybierz **File->*New->Project->C# ->Web -> ASP.NET Core Web App (Razor Pages)***.
3. Aplikację nazwij***: Moja2***.
4. Zmodyfikuj sekcję **body** pliku **Pages**-> **Shared->_Layout.cshtml** według wzoru:

```
<body>
    <div class="StronaGlowna">
        <div class="Naglowek">
            <div class="Logo">
                Logo
            </div>
            <div class="Menu">
```

```
                Menu
            </div>
            <div class="Logowanie">
                Zaloguj
            </div>
        </div>
        <div class="Tresc">
            <div class="Lewy">
                Lewy obszar roboczy
            </div>
            <div class="Prawy">
                Prawy obszar roboczy
            </div>
            <div class="Srodkowy">
                @RenderBody()
            </div>
        </div>
    </div>


    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>

    @await RenderSectionAsync("Scripts", required: false)
</body>
```

5. Zmodyfikuj kod widoku **Pages->*Index.cshtml*** według wzoru:

```
@page
@model IndexModel
@{
    ViewData["Title"] = "Home page";
}
<div>
    Witam
</div>
```

6. W okienku ***Solution Explorer*** rozwiń folder ***wwwroot->css***, a następnie wyświetl kod pliku ***site.css.*** Uzupełnij ten plik według wzoru:

```
html {
  font-size: 14px;
}

@media (min-width: 768px) {
  html {
    font-size: 16px;
  }
}

.btn:focus, .btn:active:focus, .btn-link.nav-link:focus, .form-control:focus,
.form-check-input:focus {
  box-shadow: 0 0 0 0.1rem white, 0 0 0 0.25rem #258cfb;
}

html {
  position: relative;
  min-height: 100%;
}
```

```css
body {
  margin-bottom: 60px;
}
.StronaGlowna {
    width: 1000px;
    margin: auto;
}

.Logo {
    background-color: lightblue;
    font-size: 10px;
}

.Menu {
    background-color: azure;
    font-size: 10px;
}

.Logowanie {
    text-align: right;
    background-color: lightblue;
    padding: 5px 10px 5px 10px;
}

.Lewy {
    float: left;
    width: 200px;
    height: 400px;
}

.Prawy {
    float: right;
    width: 200px;
    height: 400px;
}

.Srodkowy {
    background-color: azure;
    padding: 50px 20px 50px 20px;
    margin: 0px 200px 0px 200px;
}
```

7. Sprawdź jak teraz wygląda Twoja strona główna.

MVC to wzorzec projektowy Model – Widok - Kontroler.

.NET Core – wzór

1. Otwórz ***Microsoft Visual Studio*** .
2. Wybierz **File->*New->Project->C#->Web->ASP.NET Core Web App (Model-View-Controller).***

3. Aplikację nazwij***: PierwszaWzor***.
4. Dokonaj kompilacji projektu.

ASP MVC Tworzenie Solution

Przygotowania

1. Otwórz główne okno Visual Studio i wybierz ***File->New Project-> Wyszukaj Blank Solution***.
2. Solution nazwij ***Firma***.
3. W oknie ***SolutionExplorer*** naciśnij prawym klawiszem myszki na solution ***Firma*** i wybierz ***Add->New Project->C#->WEB->ASP.NET Core Web App (Model-View-Controller)***. Projekt nazwij ***Firma.PortalWWW***.
4. W oknie ***SolutionExplorer*** naciśnij prawym klawiszem myszki na solution ***Firma*** i wybierz ***Add->New Project->C#->WEB->ASP.NET Core Web App (Model-View-Controller)***. Projekt nazwij ***Firma.Intranet***.
5. W oknie ***SolutionExplorer*** naciśnij prawym klawiszem myszki na cały projekt ***Firma.Intranet*** i wybierz ***Set as StartUp Project***.
6. Dokonaj kompilacji całej ***solution Firma***. Jako pierwszy uruchamia się projekt ***Firma.Intranet***.

ASP MVC: Sterowanie zawartością strony

Przygotowania

1. Pobierz od prowadzącego ćwiczenia solucję Firma.
2. Otwórz pobraną solucję. Solucja zawiera projekty ***Firma.Intranet*** i ***Firma.PortalWWW***.
3. Przeczytaj dokumentację **EntityFramework Core**
   https://docs.microsoft.com/pl-pl/ef/core/

Firma.Intranet – sterowanie zawartością

1. W oknie ***SolutionExplorer*** rozwiń projekt ***Firma.Intranet.***
2. W ***Solution Explore*** naciśnij prawym klawiszem myszy na projekt ***Firma.Intranet*** i zaznacz ***Set as StartUp Project***.
3. Naciśnij prawym klawiszem myszy na folder ***Models*** i wybierz ***Add->New Folder***. Katalog nazwij ***CMS***.
4. Naciśnij prawym klawiszem myszy na folder ***CMS*** i wybierz ***Add->Class***. Klasę nazwij ***Strona***. Utwórz kod tej klasy według wzoru:

```
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;

namespace Firma.Intranet.Models.CMS
{
    public class Strona
    {
        [Key]
        public int IdStrony { get; set; }
        [Required(ErrorMessage = "Wpisz tytuł odnośnika")]
        [MaxLength(10, ErrorMessage = "Tytuł powinien zawierać max 10 znaków")]
        [Display(Name = "Tytuł odnośnika")]
        public required string LinkTytul { get; set; }
        [Required(ErrorMessage = "Wpisz tytuł strony")]
        [MaxLength(30, ErrorMessage = "Tytuł powinien zawierać max 30 znaków")]
        [Display(Name = "Tytuł")]
        public required string Tytul { get; set; }
        [Display(Name = "Treść")]
        [Column(TypeName = "nvarchar(MAX)")]
        public required string Tresc { get; set; }
        [Display(Name = "Pozycja Wyswietlania")]
        [Required(ErrorMessage = "Wpisz pozycję wyświetlania")]
        public int Pozycja { get; set; }

    }
}
```

5. Naciśnij prawym klawiszem myszy na folder ***CMS*** i wybierz ***Add->Class***. Klasę nazwij

Aktualnosc. Utwórz kod tej klasy według wzoru:

```
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;

namespace Firma.Intranet.Models.CMS
{
    public class Aktualnosc
    {
        [Key]
        public int IdAktualnosci { get; set; }
        [Required(ErrorMessage = "Wpisz tytuł aktualności")]
        [MaxLength(20, ErrorMessage = "Tytuł powinien zawierać max 20 znaków")]
        [Display(Name = "Tytuł odnośnika")]
        public required string LinkTytul { get; set; }
        [Required(ErrorMessage = "Wpisz tytuł aktualności")]
        [MaxLength(30, ErrorMessage = "Tytuł aktualności powinien zawierać max 30
znaków")]
        [Display(Name = "Tytuł")]
        public required string Tytul { get; set; }
        [Display(Name = "Treść")]
        [Column(TypeName = "nvarchar(MAX)")]
        public required string Tresc { get; set; }
        [Display(Name = "Pozycja Wyswietlania")]
        [Required(ErrorMessage = "Wpisz pozycję wyświetlania")]
        public int Pozycja { get; set; }

    }
}
```

6. Naciśnij prawym klawiszem myszy na folder ***Models*** i wybierz ***Add->New Folder***. Katalog nazwij ***Sklep***.
7. Naciśnij prawym klawiszem myszy na folder ***Sklep*** i wybierz ***Add->Class***. Klasę nazwij ***Rodzaj***. Utwórz kod tej klasy według wzoru:

```
using System.ComponentModel.DataAnnotations;

namespace Firma.Intranet.Models.Sklep
{
    public class Rodzaj
    {
        [Key]
        public int IdRodzaju { get; set; }

        [Required(ErrorMessage = "Podaj nazwę rodzaju")]
        [MaxLength(30, ErrorMessage = "Nazwa kategori powinna mieć max 30
znaków")]
        public required string Nazwa { get; set; }

        public string Opis { get; set; } = string.Empty;
        public ICollection<Towar> Towar { get; } = new List<Towar>();

    }
}
```

8. Naciśnij prawym klawiszem myszy na folder ***Sklep*** i wybierz ***Add->Class***. Klasę

nazwij ***Towar***. Utwórz kod tej klasy według wzoru:

```
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;

namespace Firma.Intranet.Models.Sklep
{
    public class Towar
    {
        [Key]
        public int IdTowaru { get; set; }

        [Required(ErrorMessage = "Kod towaru jest wymagany")]
        public required string Kod { get; set; }

        [Required(ErrorMessage = "Nazwa towaru jest wymagana")]
        public required string Nazwa { get; set; }

        [Required(ErrorMessage = "Cena towaru jest wymagana")]
        [Column(TypeName = "money")]
        public decimal Cena { get; set; }

        [Required(ErrorMessage = "Zdjęcie towaru jest wymagane")]
        [Display(Name = "Wybierz zdjęcie")]
        public required string FotoURL { get; set; }

        public string Opis { get; set; } = string.Empty;

        [ForeignKey("Rodzaj")]
        public int IdRodzaju { get; set; }
        public Rodzaj? Rodzaj { get; set; }
    }
}
```

9. Dokonaj kompilacji projektu.
10. Naciśnij prawym klawiszem myszy na folder ***Controllers*** i wybierz ***Add->Controller***.

    Ustaw parametry tego kontrolera według wzoru:

    Template=

    MVC Controller with views, using Entity Framework

    Model Class =

    Aktualnosc

    Data Context Class=

    Dodajemy (przycisk +) **Firma.Intranet.Data.FirmaIntranetContext**

    Name=

    AktualnoscController

    Pozostałe opcje pozostaw bez zmian
11. Naciśnij prawym klawiszem myszy na folder ***Controllers*** i wybierz ***Add->Controller***.

    Ustaw parametry tego kontrolera według wzoru:

    Template=

MVC Controller with views, using Entity Framework

Model Class =

Strona

Data Context Class=

FirmaIntranetContext (Firma.Intranet.Data)

Name=

StronaController

Pozostałe opcje pozostaw bez zmian

12. Podobne operacje wykonaj dla tabeli **Rodzaj** i **Towar**.
13. Edytuj **Views->Shared->_Layout.cshtml** i zmień jego sekcję nawigacyjną według wzoru:

```
<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
    <ul class="navbar-nav flex-grow-1">
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-
action="Index">Home</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-
action="Privacy">Privacy</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-
controller="Aktualnoscs" asp-action="Index">Aktualność</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Rodzajs"
asp-action="Index">Rodzaje</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Towars"
asp-action="Index">Towary</a>
        </li>
    </ul>
   </div>
```

14. Dokonaj kompilacji projektu.
15. Uruchom **Menu Tools > Menedżer pakietów NuGet > Konsola Menedżera pakietów**
16. Ustaw projekt domyślny **Firma.Intranet**.
17. Uruchom `Add-Migration InitialCreate` do tworzenia szkieletu migracji, aby utworzyć początkowy zestaw tabel dla modelu.
18. Uruchom `Update-Database` zastosować nową migrację do bazy danych. Ponieważ baza danych nie istnieje, zostanie utworzony, przed zastosowaniem migracji.
19. Dokonaj kompilacji projektu.

Solution Firma – wspólne klasy do obsługi bazy danych

1. W oknie ***SolutionExplorer*** PPM na całą **Solution Firma** i wybierz ***Add->New Project*** ->**Class Library, C#**. Projekt nazwij **Firma.Data**.
2. PPM na **Firma.Data i Add->New Folder.** Folder nazwij **Data.**
3. PPM na **Data i Add->New Folder.** Folder nazwij **CMS.**
4. PPM na **CMS i Add->Class.** Klasę nazwij **Aktualnosc.** Utwórz kod tej klasy według wzoru:

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Firma.Data.CMS
{
    public class Aktualnosc
    {
        [Key]
        public int IdAktualnosci { get; set; }
        [Required(ErrorMessage = "Wpisz tytuł aktualności")]
        [MaxLength(20, ErrorMessage = "Tytuł powinien zawierać max 20 znaków")]
        [Display(Name = "Tytuł odnośnika")]
        public required string LinkTytul { get; set; }
        [Required(ErrorMessage = "Wpisz tytuł aktualności")]
        [MaxLength(30, ErrorMessage = "Tytuł aktualności powinien zawierać max 30
znaków")]
        [Display(Name = "Tytuł")]
        public required string Tytul { get; set; }
        [Display(Name = "Treść")]
        [Column(TypeName = "nvarchar(MAX)")]
        public required string Tresc { get; set; }
        [Display(Name = "Pozycja Wyswietlania")]
        [Required(ErrorMessage = "Wpisz pozycję wyświetlania")]
        public int Pozycja { get; set; }
    }
}
```

5. PPM na **CMS i Add->Class.** Klasę nazwij **Strona.** Utwórz kod tej klasy według wzoru:

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Firma.Data.CMS
{
    public class Strona
    {
        [Key]
        public int IdStrony { get; set; }

        [Required(ErrorMessage = "Wpisz tytuł odnośnika do strony")]
        [MaxLength(10, ErrorMessage = "Tytuł powinien zawierać max 10 znaków")]
        [Display(Name = "Tytuł Odnośnika")]
        public required string LinkTytul { get; set; }

        [Required(ErrorMessage = "Wpisz tytuł strony")]
        [MaxLength(30, ErrorMessage = "Tytuł strony powinien zawierać max 30
znaków")]
        [Display(Name = "Tytuł Strony")]
        public required string Tytul { get; set; }

        [Required(ErrorMessage = "Wpisz treść strony")]
        [Column(TypeName = "nvarchar(MAX)")]
        [Display(Name = "Treść")]
        public required string Tresc { get; set; }

        [Required(ErrorMessage = "Wpisz pozycję")]
        [Display(Name = "Pozycja wyświetlania")]
        public int Pozycja { get; set; }
    }
}
```

6. PPM na **Data i Add->New Folder.** Folder nazwij **Sklep.**
7. PPM na **Sklep i Add->Class.** Klasę nazwij **Rodzaj.** Utwórz kod tej klasy według wzoru:

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Firma.Data.Sklep
{
    public class Rodzaj
    {
        [Key]
        public int IdRodzaju { get; set; }

        [Required(ErrorMessage = "Podaj nazwę rodzaju")]
        [MaxLength(30, ErrorMessage = "Nazwa kategori powinna mieć max 30
znaków")]
        public required string Nazwa { get; set; }

        public string Opis { get; set; } = string.Empty;
        public ICollection<Towar> Towar { get; } = new List<Towar>();
    }
}
```

Uwaga polecam zapoznanie się ze stroną:

https://learn.microsoft.com/en-us/ef/core/modeling/relationships?source=recommendations&tabs=fluent-api%2Cfluent-api-simple-key%2Csimple-key

8. PPM na **Sklep i Add->Class.** Klasę nazwij **Towar.** Utwórz kod tej klasy według wzoru:

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Firma.Data.Sklep
{
    public class Towar
    {
        [Key]
        public int IdTowaru { get; set; }

        [Required(ErrorMessage = "Kod towaru jest wymagany")]
        public required string Kod { get; set; }

        [Required(ErrorMessage = "Nazwa towaru jest wymagana")]
        public required string Nazwa { get; set; }

        [Required(ErrorMessage = "Cena towaru jest wymagana")]
        [Column(TypeName = "money")]
        public decimal Cena { get; set; }

        [Required(ErrorMessage = "Zdjęcie towaru jest wymagane")]
        [Display(Name = "Wybierz zdjęcie")]
        public required string FotoURL { get; set; }

        public string Opis { get; set; } = string.Empty;

        [ForeignKey("Rodzaj")]
        public int IdRodzaju { get; set; }
        public Rodzaj? Rodzaj { get; set; }
    }
}
```

9. PPM na **Data i Add->Class.** Klasę nazwij **FirmaContext.** Utwórz kod tej klasy według wzoru:

```
using Firma.Data.CMS;
using Firma.Data.Sklep;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Firma.Data
{
```

```
    public class FirmaContext : DbContext
    {
        public FirmaContext(DbContextOptions<FirmaContext> options)
            : base(options)
        {
        }

        public DbSet<Rodzaj> Rodzaj { get; set; } = default!;
        public DbSet<Towar> Towar { get; set; } = default!;
        public DbSet<Aktualnosc> Aktualnosc { get; set; } = default!;
        public DbSet<Strona> Strona { get; set; } = default!;
    }
}
```

10. W klasie **FirmaContext** PPM **DbContext** i wybierz Quick Actions a nastepnie **Install package Microsoft.EntityFrameworkCore.dlln(Last)**
11. PPM na **Firma.Data** i wybierz **Build**.
12. W projekcie **Firma.Intranet** wykasuj zawartość folderów **Data**, **Migrations**, oraz z **Models** wykasuj foldery **CMS** oraz **Sklep** z całą zawartością.
13. PPM na **Dependencies** i **Add Project Reference,** a nstępnie **Solution->** zaznacz **Firma.Data.**
14. Wejdź do wszystkich widokow utworzonych dla **Aktualności**, **Stron**, **Rodzajów** i **Towarów** i zmień ich nagłówki według wozru:

    `@model Firma.Data.Data.CMS.Aktualnosc`

    `@model IEnumerable<Firma.Data.Data.CMS.Aktualnosc>`
15. Wejdź do wszystkich kontrolerów utworzonych dla **Aktualności**, **Stron**, **Rodzajów** i **Towarów** i popraw kod, tak żeby kontrolery uzywały klas z projektu **Firma.Data** (Quick Actions oraz zmiana nazwy klasy na `FirmaContext`).
16. Podobnie popraw plik **Program.cs**
17. W ***Solution Explore*** naciśnij prawym klawiszem myszy na projekt ***Firma.Intranet*** i zaznacz ***Set as StartUp Project***.
18. Dokonaj kompilacji projektu **Firma.Intranet.**

Uwaga w przypadku niezgodności Pakietów wykonaj następujące kroki:

1. Z menu **Tools** wybierz **NuGet Package Manager** -> **Package Manager Console.**
2. W **Package Manager Console** wybierz **Default project** (górna część okna): **Firma.Intranet**.
3. W konsoli wpisz `dotnet restore`
4. W **Firma.Data** PPM na **Dependencies** i wejdź do zarządzania pakietami zainstalowanymi. Następnie obniż wersję **Microsoft.EntityFrameworkCore** na taką jak masz w **Firma.Intranet** (np. 6.0.14) i wybierz **Aktualizuj**.
5. PPM na **Firma.Data** -> Właściwości i wybierz tam **.NET Core 3.0** a następnie

naciśnij **CTRL+S**.

6. Dokonaj kompilacji projektu. Popraw wszystkie błędy dodając odpowiednie pakiety (**Quick Actions**).

Łączenie z odpowiednim Contextem

1. Wejdź do projektu **Firma.Intanet**.
2. Edytuj **Program.cs** i zmodyfikuj poniższą sekcję według wzoru:
3.
```
builder.Services.AddDbContext<FirmaContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("FirmaContext") ?? throw new InvalidOperationException("Connection string 'FirmaContext' not found.")));
```

=====

4. Edytuj plik **appsettings.json** i zmień sekcję **ConnectionStrings** według wzoru:

```
  "ConnectionStrings": {
"FirmaContext": "Server=(localdb)\\mssqllocaldb;Database=FirmaContext-65b1cc0e-0da8-47a7-a5c7-cefe884f5bff;Trusted_Connection=True;MultipleActiveResultSets=true"
  }
```

Tworzenie bazy danych

1. Dokonaj kompilacji lub bildowania wszystkich projektów.
2. W pliku **appsettings.json** sprawdź ustawienia bazy danych (przy utworzeniu bazy jej nazwa powinna być nowa) w projekcie **Firma.Intranet**.

```
"ConnectionStrings": {
    "FirmaContext": "Server=(localdb)\\mssqllocaldb;Database="
```

3. PPM na **Firma.Intranet** i ustaw **Set as StarUp Project**.
4. Z menu **Tools** wybierz **NuGet Package Manager** -> **Package Manager Console.**
5. W **Package Manager Console** wybierz **Default project** (górna część okna): **Firma.Data.**
6. W konsoli wpisz: **Add-migration M1.**
7. W pliku **Firma.Data->Migrations** usunąć całą sekcję **using.**
8. W tym pliku naciśnij prawym klawiszem myszki na **Migration** (klasa dziedzicząca) i **Quick Actions and Refactorings** i wybierz **Install pack. „Microsoft.EntityFrameworkCore...”** use local version 8.0.3**.**
9. Dodaj stosowne using-i.
10. Podobnie w tym pliku naciśnij prawym klawiszem myszki na błędy w pliku **FirmaContextModelSnapshot** i **Quick Actions and Refactorings** i wybierz **Install**

„Microsoft.EntityFrameworkCore...” use local version 8.0.3

11. Dokonaj buildowania projektu **Firma.Data**.
12. W **Package Manager Console** wybierz **Default project** (górna część okna): **Firma.Data.**
13. W konsoli wpisz: **Update-database.**

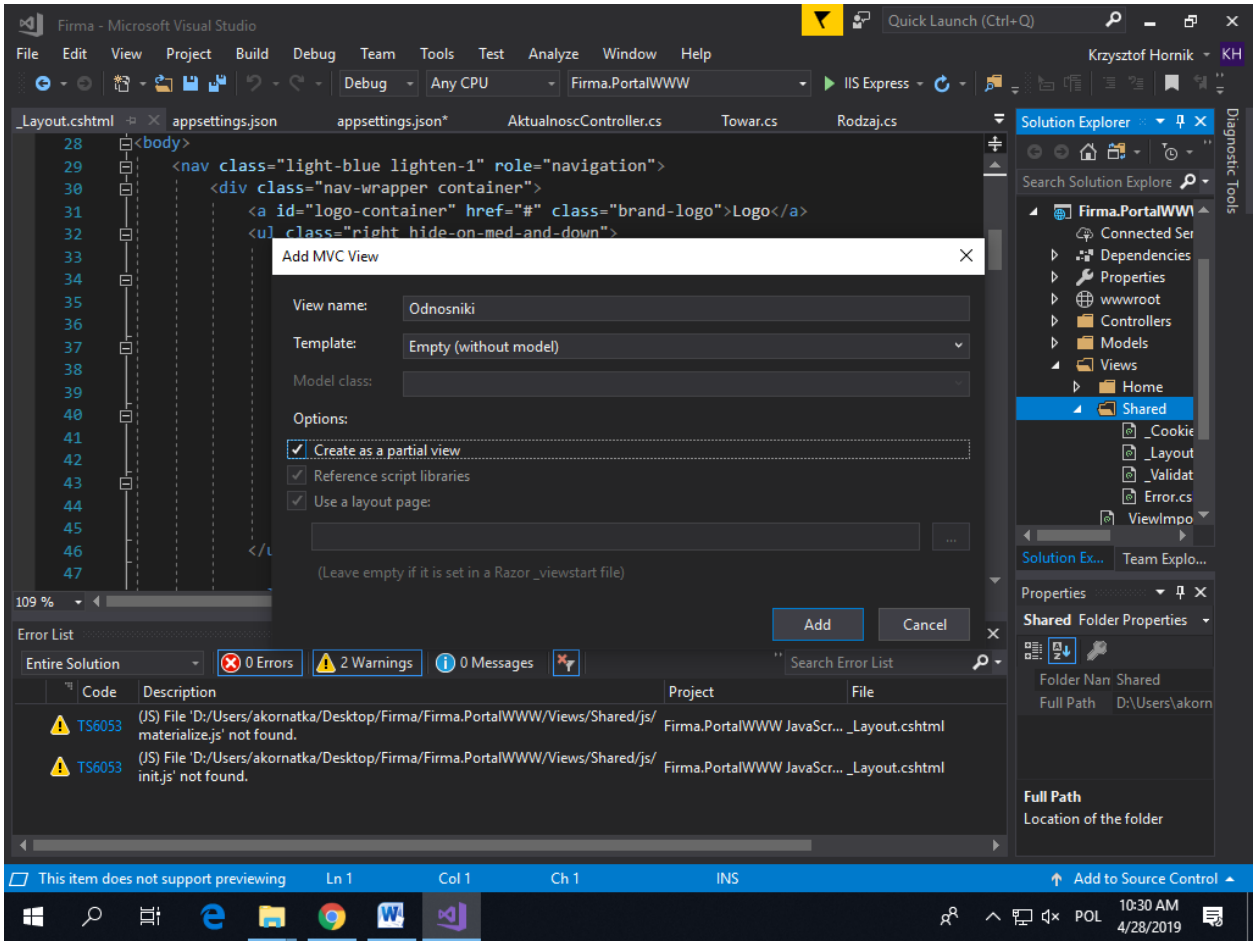
Firma.PortalWWW – wyświetlanie zawartości

1. W ***Solution Explore*** naciśnij prawym klawiszem myszy na projekt ***Firma.PortalWWW*** i zaznacz ***Set as StartUp Project***.
2. Edytuj plik appsettings.json i dodaj sekcję `"ConnectionStrings"` według wzoru z tego samego pliku z projektu **Firma.Intranet** (ta sekcja powinna być taka sama jak w **Firma.Intranet**).
3. PPM na **Dependencies** i **Add Reference,** a nstępnie **Solution->** zaznacz **Firma.Data.**
4. Edytuj plik **Program.cs** i zmodyfikuj jego kod według wzoru (dodaj też odpowiednie using-i):

```
using Firma.Data.Data;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.DependencyInjection;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddDbContext<FirmaContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("FirmaContext")));


// Add services to the container.
builder.Services.AddControllersWithViews();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    // The default HSTS value is 30 days. You may want to change this for
production scenarios, see https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();
```

```
app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();
```

5. Rozwiń catalog ***Views->Shared***.
6. PPM na folder ***Shared*** i wybierz ***Add->View (RazorView)***. Widok nazwij ***Odnosniki***. Ustaw parametry tego widoku według wzoru:

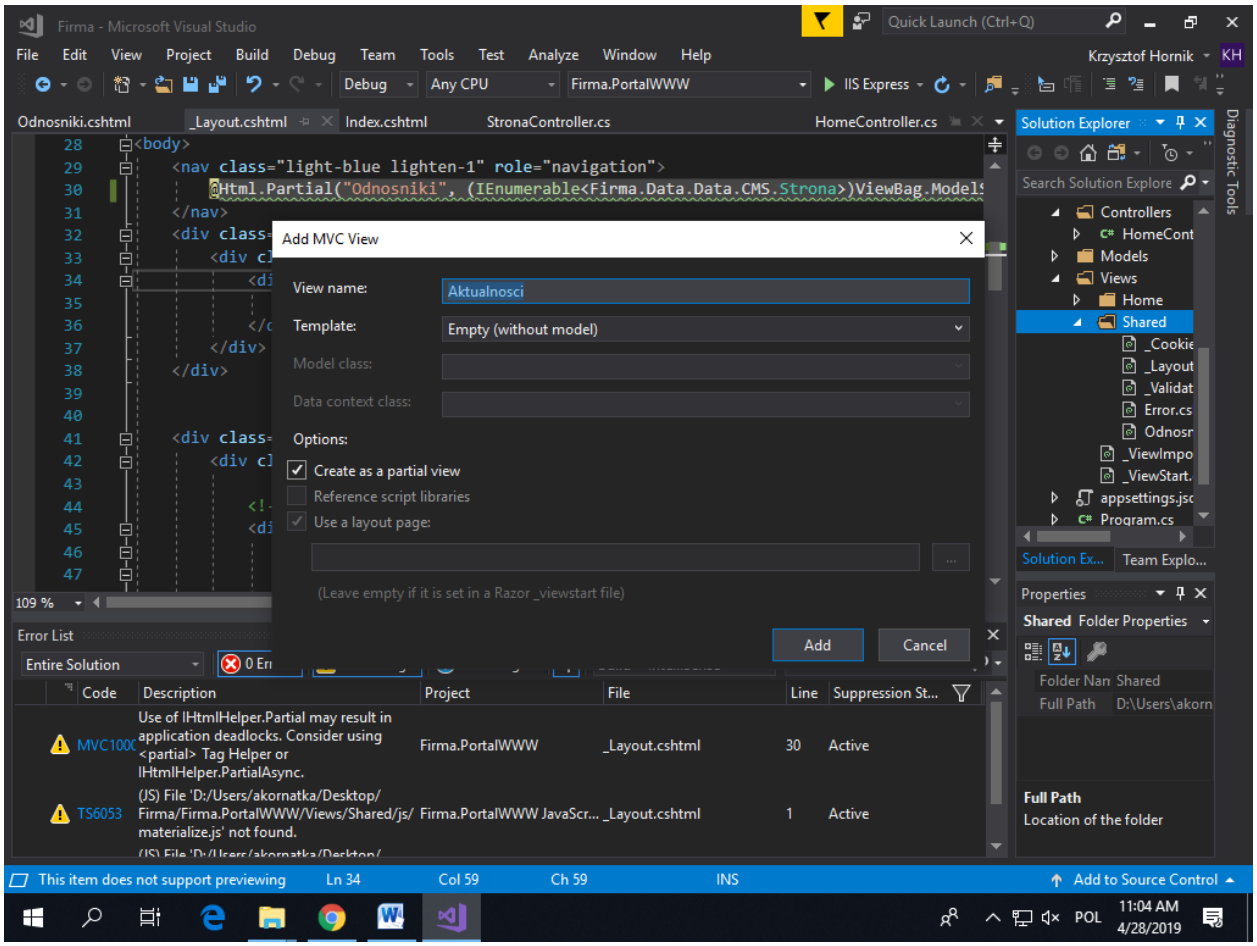


Następnie uzupełnij kod tego widoku według wzoru:

```
@model IEnumerable<Firma.Data.Data.CMS.Strona>
<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
    <ul class="navbar-nav flex-grow-1">
        @foreach (var item in Model)
        {
            <li class="nav-item">
                <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-
```

```
action="Index" asp-route-id="@item.IdStrony">@item.LinkTytul</a>
            </li>
        }
    </ul>
</div>
```

7. Naciśnij prawym klawiszem myszy na folder ***Shared*** i wybierz ***Add->View (RazorView)***. Widok nazwij ***Aktualnosci***. Ustaw parametry tego widoku według wzoru:



Następnie uzupełnij kod tego widoku według wzoru:

```
@model IEnumerable<Firma.Data.Data.CMS.Aktualnosc>
<div class="container align-content-center">
    <div class="container text-center">
        <hr />
        <div class="row">
            @foreach (var item in Model)
            {
                <div class="col-12 col-md-6 col-lg-4">
                    <div class="card">
```

```
                        <img src="~/content/cloud--22.png" class="card-img-top"
alt="CA">
                        <div class="card-body">
                            <h5 class="card-title">@item.Tytul</h5>
                            <p class="card-text">@item.Tresc</p>
                            <a href="#" class="btn btn-outline-primary">Go
somewhere</a>
                        </div>
                    </div>
                </div>

            }
        </div>
        <hr />
    </div>
</div>
```

8. Edytuj plik ***_Layout.cshtml*** (katalog ***Views->Shared***).

   Zmodyfikuj jego kod według wzoru:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - Firma.PortalWWW</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
    <link rel="stylesheet" href="~/Firma.PortalWWW.styles.css" asp-append-
version="true" />
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white
border-bottom box-shadow mb-3">
            <div class="container-fluid">
                <a class="navbar-brand" asp-area="" asp-controller="Home" asp-
action="Index">Firma.PortalWWW</a>
                <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target=".navbar-collapse" aria-controls="navbarSupportedContent"
                        aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                @await Html.PartialAsync("Odnosniki",
(IEnumerable<Firma.Data.Data.CMS.Strona>)ViewBag.ModelStrony)
            </div>
        </nav>
    </header>
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>
    @await Html.PartialAsync("Aktualnosci",
(IEnumerable<Firma.Data.Data.CMS.Aktualnosc>)ViewBag.ModelAktualnosci)
    <footer class="border-top footer text-muted">
        <div class="container">
            &copy; 2024 - Firma.PortalWWW - <a asp-area="" asp-controller="Home" asp-
action="Privacy">Privacy</a>
```

```
        </div>
    </footer>
    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>
    @await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```

9. Edytuj plik ***HomeController.cs*** (zakładka ***Controllers***) i uzupełnij jego kod według wzoru:

```
using Firma.Data.Data;
using Firma.PortalWWW.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Diagnostics;


namespace Firma.PortalWWW.Controllers
{
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;
        private readonly FirmaContext _context;
public HomeController(ILogger<HomeController> logger, FirmaContext context)
        {
            _logger = logger;
            _context = context;
        }

        public async Task<IActionResult> Index(int? id)
        {
            ViewBag.ModelStrony                                                    =await
_context.Strona.OrderBy(s=>s.Pozycja).ToListAsync();
            //ViewBag.ModelStrony =
            //      (
            //          from strona in _context.Strona
            //          orderby strona.Pozycja
            //          select strona
            //      ).ToList();
            ViewBag.ModelAktualnosci = await _context.Aktualnosc.OrderByDescending(a =>
a.Pozycja).Take(3).ToListAsync();
```

```
            //ViewBag.ModelAktualnosci =
            //    (
            //        from aktualnosc in _context.Aktualnosc
            //        orderby aktualnosc.Pozycja descending
            //        select aktualnosc
            //    ).Take(3).ToList();
            if (id == null)
                id = 1;//koniecznie powinna być strona o id=1 w bazie
            var item = await _context.Strona.FindAsync(id);
            return View(item);
        }
        public IActionResult OFirmie()
        {
            return View();
        }
        public IActionResult Privacy()
        {
            return View();
        }

        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore =
true)]
        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId = Activity.Current?.Id ??
HttpContext.TraceIdentifier });
        }
    }
}
```

10. W katalogu ***Views->Home*** edytuj plik ***Index.cshtml*** i uzupełnij jego kod według wzoru:

```
@model Firma.Data.Data.CMS.Strona
@{
    ViewData["Title"] = @Model.Tytul;
}
<div class="container align-content-center">
    <h4>@Model.Tytul </h4>
    @Model.Tresc
    <br />
    Data i godzina edycji: @DateTime.Now.ToString()
```

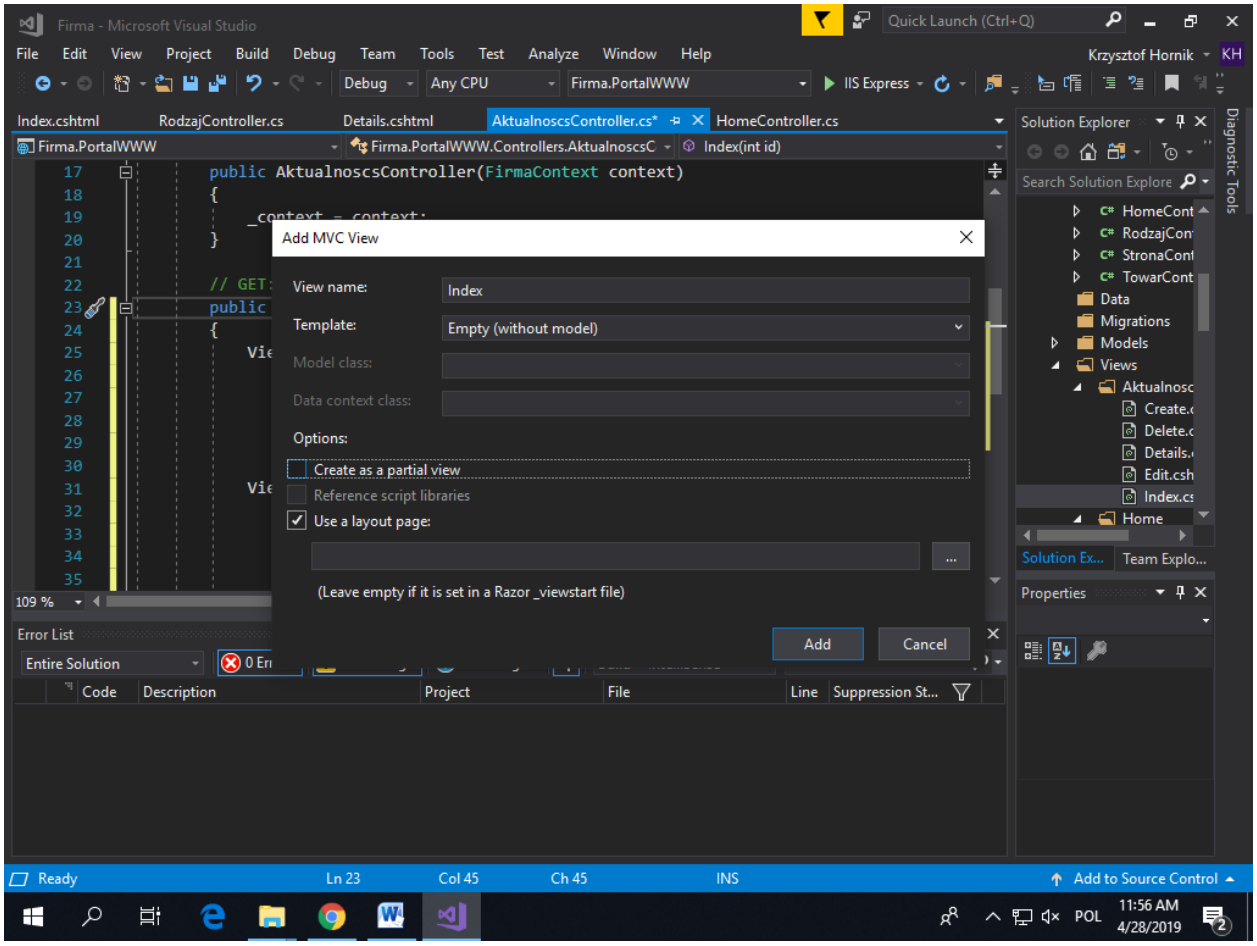
```
</div>
```

11. Naciśnij prawym klawiszem myszki na folder ***Controllers*** i wybierz ***Add->Controller (MVC Controller - Empty)***. Kontroler nazwij **`AktualnoscController`**.

Następnie utwórz kod tego kontrolera według wzoru:

```
using Firma.Data.Data;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace Firma.PortalWWW.Controllers
{
    public class AktualnoscController : Controller
    {
        private readonly FirmaContext _context;
        public AktualnoscController(FirmaContext context)
        {
            _context = context;
        }
        public async Task<IActionResult> Index(int id)
        {
            ViewBag.ModelStrony = await _context.Strona.OrderBy(s => s.Pozycja).ToListAsync();
            ViewBag.ModelAktualnosci = await _context.Aktualnosc.OrderByDescending(a => a.Pozycja).Take(3).ToListAsync();
            var item = await _context.Aktualnosc.FirstOrDefaultAsync(a => a.IdAktualnosci == id);
            if (item == null)
            {
                return NotFound();
            }
            return View(item);
        }
    }
}
```

12. W klasie ***AktualnosciController*** naciśnij prawym klawiszem myszki na nazwę funkcji ***Index*** i wybierz ***Add->View (RazorView)***. Ustaw parametry tego widoku według wzoru:



Następnie uzupełnij kod tego widoku według wzoru:

```
@model Firma.Data.Data.CMS.Aktualnosc
@{
    ViewData["Title"] = @Model.Tytul;
}
<div class="container align-content-center">
    <h4>@Model.Tytul </h4>
    @Model.Tresc
    <br />
    Data i godzina edycji: @DateTime.Now.ToString()
</div>
```

13. Edytuj Views->Shared->Aktualnosci.cshtml i zmień link w aktualnościach według wzoru:

```
@model IEnumerable<Firma.Data.Data.CMS.Aktualnosc>
<div class="container align-content-center">
    <div class="container text-center">
```

```
        <hr />
        <div class="row">
            @foreach (var item in Model)
            {
                <div class="col-12 col-md-6 col-lg-4">
                    <div class="card">
                        <img   src="~/content/cloud--22.png"   class="card-img-top"
alt="CA">
                        <div class="card-body">
                            <h5 class="card-title">@item.Tytul</h5>
                            <p class="card-text">@item.Tresc</p>
                            <a class="btn btn-outline-primary" asp-area="" asp-
controller="Aktualnosc" asp-action="Index" asp-route-
id="@item.IdAktualnosci">@item.LinkTytul</a>
                        </div>
                    </div>
                </div>

            }
        </div>
        <hr />
    </div>
</div>
```

Solution Firma – wspólne klasy do obsługi bazy danych

1. W oknie ***SolutionExplorer*** PPM na całą **Solution Firma** i wybierz ***Add->New Project*** ->**Class Library, C#**. Projekt nazwij **Firma.Data**.
2. PPM na **Firma.Data i Add->New Folder.** Folder nazwij **Data.**
3. PPM na **Data i Add->New Folder.** Folder nazwij **CMS.**
4. PPM na **CMS i Add->Class.** Klasę nazwij **Aktualnosc.** Utwórz kod tej klasy według wzoru:

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Firma.Data.CMS
{
    public class Aktualnosc
    {
        [Key]
        public int IdAktualnosci { get; set; }
        [Required(ErrorMessage = "Wpisz tytuł aktualności")]
        [MaxLength(20, ErrorMessage = "Tytuł powinien zawierać max 20 znaków")]
        [Display(Name = "Tytuł odnośnika")]
        public required string LinkTytul { get; set; }
        [Required(ErrorMessage = "Wpisz tytuł aktualności")]
        [MaxLength(30, ErrorMessage = "Tytuł aktualności powinien zawierać max 30
znaków")]
        [Display(Name = "Tytuł")]
        public required string Tytul { get; set; }
        [Display(Name = "Treść")]
        [Column(TypeName = "nvarchar(MAX)")]
        public required string Tresc { get; set; }
        [Display(Name = "Pozycja Wyswietlania")]
        [Required(ErrorMessage = "Wpisz pozycję wyświetlania")]
        public int Pozycja { get; set; }
    }
}
```

5. PPM na **CMS i Add->Class.** Klasę nazwij **Strona.** Utwórz kod tej klasy według wzoru:

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Firma.Data.CMS
{
    public class Strona
    {
        [Key]
        public int IdStrony { get; set; }

        [Required(ErrorMessage = "Wpisz tytuł odnośnika do strony")]
        [MaxLength(10, ErrorMessage = "Tytuł powinien zawierać max 10 znaków")]
        [Display(Name = "Tytuł Odnośnika")]
        public required string LinkTytul { get; set; }

        [Required(ErrorMessage = "Wpisz tytuł strony")]
        [MaxLength(30, ErrorMessage = "Tytuł strony powinien zawierać max 30
znaków")]
        [Display(Name = "Tytuł Strony")]
        public required string Tytul { get; set; }

        [Required(ErrorMessage = "Wpisz treść strony")]
        [Column(TypeName = "nvarchar(MAX)")]
        [Display(Name = "Treść")]
        public required string Tresc { get; set; }

        [Required(ErrorMessage = "Wpisz pozycję")]
        [Display(Name = "Pozycja wyświetlania")]
        public int Pozycja { get; set; }
    }
}
```

6. PPM na **Data i Add->New Folder.** Folder nazwij **Sklep.**
7. PPM na **Sklep i Add->Class.** Klasę nazwij **Rodzaj.** Utwórz kod tej klasy według wzoru:

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Firma.Data.Sklep
{
    public class Rodzaj
    {
        [Key]
        public int IdRodzaju { get; set; }

        [Required(ErrorMessage = "Podaj nazwę rodzaju")]
        [MaxLength(30, ErrorMessage = "Nazwa kategori powinna mieć max 30
znaków")]
        public required string Nazwa { get; set; }

        public string Opis { get; set; } = string.Empty;
        public ICollection<Towar> Towar { get; } = new List<Towar>();
    }
}
```

Uwaga polecam zapoznanie się ze stroną:

https://learn.microsoft.com/en-us/ef/core/modeling/relationships?source=recommendations&tabs=fluent-api%2Cfluent-api-simple-key%2Csimple-key

8. PPM na **Sklep i Add->Class.** Klasę nazwij **Towar.** Utwórz kod tej klasy według wzoru:

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Firma.Data.Sklep
{
    public class Towar
    {
        [Key]
        public int IdTowaru { get; set; }

        [Required(ErrorMessage = "Kod towaru jest wymagany")]
        public required string Kod { get; set; }

        [Required(ErrorMessage = "Nazwa towaru jest wymagana")]
        public required string Nazwa { get; set; }

        [Required(ErrorMessage = "Cena towaru jest wymagana")]
        [Column(TypeName = "money")]
        public decimal Cena { get; set; }

        [Required(ErrorMessage = "Zdjęcie towaru jest wymagane")]
        [Display(Name = "Wybierz zdjęcie")]
        public required string FotoURL { get; set; }

        public string Opis { get; set; } = string.Empty;

        [ForeignKey("Rodzaj")]
        public int IdRodzaju { get; set; }
        public Rodzaj? Rodzaj { get; set; }
    }
}
```

9. PPM na **Data i Add->Class.** Klasę nazwij **FirmaContext.** Utwórz kod tej klasy według wzoru:

```
using Firma.Data.CMS;
using Firma.Data.Sklep;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Firma.Data
{
```

```
    public class FirmaContext : DbContext
    {
        public FirmaContext(DbContextOptions<FirmaContext> options)
            : base(options)
        {
        }

        public DbSet<Rodzaj> Rodzaj { get; set; } = default!;
        public DbSet<Towar> Towar { get; set; } = default!;
        public DbSet<Aktualnosc> Aktualnosc { get; set; } = default!;
        public DbSet<Strona> Strona { get; set; } = default!;
    }
}
```

10. W klasie **FirmaContext** PPM **DbContext** i wybierz Quick Actions a nastepnie **Install package Microsoft.EntityFrameworkCore.dlln(Last)**
11. PPM na **Firma.Data** i wybierz **Build**.
12. W projekcie **Firma.Intranet** wykasuj zawartość folderów **Data**, **Migrations**, oraz z **Models** wykasuj foldery **CMS** oraz **Sklep** z całą zawartością.
13. PPM na **Dependencies** i **Add Project Reference,** a nstępnie **Solution->** zaznacz **Firma.Data.**
14. Wejdź do wszystkich widokow utworzonych dla **Aktualności**, **Stron**, **Rodzajów** i **Towarów** i zmień ich nagłówki według wozru:

    `@model Firma.Data.Data.CMS.Aktualnosc`

    `@model IEnumerable<Firma.Data.Data.CMS.Aktualnosc>`
15. Wejdź do wszystkich kontrolerów utworzonych dla **Aktualności**, **Stron**, **Rodzajów** i **Towarów** i popraw kod, tak żeby kontrolery uzywały klas z projektu **Firma.Data** (Quick Actions oraz zmiana nazwy klasy na `FirmaContext`).
16. Podobnie popraw plik **Program.cs**
17. W ***Solution Explore*** naciśnij prawym klawiszem myszy na projekt ***Firma.Intranet*** i zaznacz ***Set as StartUp Project***.
18. Dokonaj kompilacji projektu **Firma.Intranet.**

Uwaga w przypadku niezgodności Pakietów wykonaj następujące kroki:

1. Z menu **Tools** wybierz **NuGet Package Manager** -> **Package Manager Console.**
2. W **Package Manager Console** wybierz **Default project** (górna część okna): **Firma.Intranet**.
3. W konsoli wpisz `dotnet restore`
4. W **Firma.Data** PPM na **Dependencies** i wejdź do zarządzania pakietami zainstalowanymi. Następnie obniż wersję **Microsoft.EntityFrameworkCore** na taką jak masz w **Firma.Intranet** (np. 6.0.14) i wybierz **Aktualizuj**.
5. PPM na **Firma.Data** -> Właściwości i wybierz tam **.NET Core 3.0** a następnie

naciśnij **CTRL+S**.

6. Dokonaj kompilacji projektu. Popraw wszystkie błędy dodając odpowiednie pakiety (**Quick Actions**).

Łączenie z odpowiednim Contextem

1. Wejdź do projektu **Firma.Intanet**.
2. Edytuj **Program.cs** i zmodyfikuj poniższą sekcję według wzoru:
3.
```
builder.Services.AddDbContext<FirmaContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("FirmaContext") ?? throw new InvalidOperationException("Connection string 'FirmaContext' not found.")));
```

=====

4. Edytuj plik **appsettings.json** i zmień sekcję **ConnectionStrings** według wzoru:

```
  "ConnectionStrings": {
"FirmaContext": "Server=(localdb)\\mssqllocaldb;Database=FirmaContext-65b1cc0e-0da8-47a7-a5c7-cefe884f5bff;Trusted_Connection=True;MultipleActiveResultSets=true"
  }
```

Tworzenie bazy danych

1. Dokonaj kompilacji lub bildowania wszystkich projektów.
2. W pliku **appsettings.json** sprawdź ustawienia bazy danych (przy utworzeniu bazy jej nazwa powinna być nowa) w projekcie **Firma.Intranet**.

```
"ConnectionStrings": {
    "FirmaContext": "Server=(localdb)\\mssqllocaldb;Database="
```

3. PPM na **Firma.Intranet** i ustaw **Set as StarUp Project**.
4. Z menu **Tools** wybierz **NuGet Package Manager** -> **Package Manager Console.**
5. W **Package Manager Console** wybierz **Default project** (górna część okna): **Firma.Data.**
6. W konsoli wpisz: **Add-migration M1.**
7. W pliku **Firma.Data->Migrations** usunąć całą sekcję **using.**
8. W tym pliku naciśnij prawym klawiszem myszki na **Migration** (klasa dziedzicząca) i **Quick Actions and Refactorings** i wybierz **Install pack. „Microsoft.EntityFrameworkCore...”** use local version 8.0.3**.**
9. Dodaj stosowne using-i.
10. Podobnie w tym pliku naciśnij prawym klawiszem myszki na błędy w pliku **FirmaContextModelSnapshot** i **Quick Actions and Refactorings** i wybierz **Install**

„Microsoft.EntityFrameworkCore...” use local version 8.0.3

11. Dokonaj buildowania projektu **Firma.Data**.
12. W **Package Manager Console** wybierz **Default project** (górna część okna): **Firma.Data.**
13. W konsoli wpisz: **Update-database.**

Migracje

Utworzenie bazy

1. Wejdź do projektu **Firma.Data**.
2. PPM na **Dependencies-> Manage NuGet Packages** -> **Browse** -> zainstaluj pakiet **Microsoft.EntityFrameworkCore.Relational** w wersji takie samej jaki masz pakiet **Microsoft.EntityFrameworkCore**.
3. PPM na **Dependencies-> Manage NuGet Packages** -> **Browse** -> zainstaluj pakiet **Microsoft.EntityFrameworkCore.SqlServer** w wersji takie samej jaki masz pakiet **Microsoft.EntityFrameworkCore**.
4. W pliku **appsettings.json** sprawdź ustawienia bazy danych (przy utworzeniu bazy jej nazwa powinna być nowa) w projekcie **Firma.PortalWWW i Firma.Intranet**.
5. PPM na **Firma.Intranet** i ustaw **Set as StarUp Project**.
6. Z menu **Tools** wybierz **NuGet Package Manager** -> **Package Manager Console.**
7. W **Package Manager Console** wybierz **Default project** (górna część okna): **Firma.Data.**
8. W konsoli wpisz: **Add-migration M1.**
9. W konsoli wpisz: **Update-database**

Modyfikacja bazy

1. PPM na **Firma.PortalWWW** i ustaw **Set as StarUp Project**.
2. Z menu **Tools** wybierz **NuGet Package Manager** -> **Package Manager Console.**
3. W **Package Manager Console** wybierz **Default project** (górna część okna): **Firma.Data.**
4. W konsoli wpisz: **Add-migration M2.**
5. W konsoli wpisz: **Update-database.**

Sklep internetowy - 1

Przygotowania

1. Przeczytaj dokumentację **EntityFramework Core**
   https://docs.microsoft.com/pl-pl/ef/core/
2. Otwórz solucję utworzoną podczas poprzednich zajęć.

Firma.PortalWWW – strona główna sklepu

1. Przejdź do projektu **Firma.PortalWWW**.
2. W ***Solution Explorer*** naciśnij prawym klawiszem myszy na projekt ***Firma.PortalWWW*** i zaznacz ***Set as StartUp Project***.
3. W oknie ***SolutionExplorer*** rozwiń projekt ***Firma.PortalWWW.***
4. Naciśnij prawym klawiszem myszki na ***Views->Shared*** i wybierz ***Add->View (Razor View – Empty)***. Widok nazwij ***_SklepLayout***.
5. Wzorując się na kodzie pliku **_Layout.cshtml** uzupełnij kod tego pliku według wzoru:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - Firma.PortalWWW</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
    <link rel="stylesheet" href="~/Firma.PortalWWW.styles.css" asp-append-
version="true" />
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-
white border-bottom box-shadow mb-3">
            <div class="container-fluid">
                <a class="navbar-brand" asp-area="" asp-controller="Home" asp-
action="Index">Firma.PortalWWW</a>
                <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target=".navbar-collapse" aria-
controls="navbarSupportedContent"
                        aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                @await Html.PartialAsync("Odnosniki",
(IEnumerable<Firma.Data.Data.CMS.Strona>)ViewBag.ModelStrony)
            </div>
```

```
            </nav>
        </header>
        <div class="container">
            <main role="main" class="pb-3">
                @RenderBody()
            </main>
        </div>
        @await Html.PartialAsync("Aktualnosci",
(IEnumerable<Firma.Data.Data.CMS.Aktualnosc>)ViewBag.ModelAktualnosci)
        <footer class="border-top footer text-muted">
            <div class="container">
                &copy; 2024 - Firma.PortalWWW - <a asp-area="" asp-controller="Home"
asp-action="Privacy">Privacy</a>
            </div>
        </footer>
        <script src="~/lib/jquery/dist/jquery.min.js"></script>
        <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
        <script src="~/js/site.js" asp-append-version="true"></script>
        @await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```

6. Dokonaj kompilacji projektu.
7. Naciśnij prawym klawiszem myszki na folder ***Controllers*** i wybierz ***Add->Controller (MVC Controller - Empty).*** Kontroler nazwij ***SklepController***.

Uzupełnij kod tego kontrolera według wzoru:

```
using Firma.Data.Data;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace Firma.PortalWWW.Controllers
{
    public class SklepController : Controller
    {
        private readonly FirmaContext _context;
        public SklepController(FirmaContext context)
        {
            _context = context;
        }
        public async Task<IActionResult> Index(int? id)
        {
            ViewBag.ModelStrony = await _context.Strona.OrderBy(s =>
s.Pozycja).ToListAsync();
            ViewBag.Rodzaje = await _context.Rodzaj.ToListAsync();
            if(id == null)
            {
                id = 1;
            }
            var items=await
_context.Towar.Where(t=>t.IdRodzaju==id).ToListAsync();
            return View(items);
        }
    }
}
```

8. Edytuj plik ***SklepController*** i naciśnij prawym klawiszem myszki na nazwę funkcji ***Index***, wybierz ***Add->View (Razor View – Empty)***. Widok nazwij ***Index***.

Uzupełnij kod tego widoku według wzoru:

```
@model IEnumerable<Firma.Data.Data.Sklep.Towar>
@{
    ViewData["Title"] = "Sklep";
    Layout = "~/Views/Shared/_SklepLayout.cshtml";
}
<table class="table">
    <thead>
        <tr>
            <th scope="col">Foto</th>
            <th scope="col">Nazwa</th>
            <th scope="col">Cena</th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model)
        {
            <tr>
                <td>Zdjecie</td>
                <td>@item.Nazwa</td>
                <td>@item.Cena</td>
            </tr>
        }
    </tbody>
   </table>
```

9. Naciśnij prawym klawiszem myszki na **Views->Shared**, wybierz ***Add->View (Razor View)***. Widok nazwij ***RodzajeMenu, Empty, Create as a partial view.***

Wzorując się na Shared->Odnosniki uzupełnij kod tego widoku według wzoru:

```
@model IEnumerable<Firma.Data.Data.Sklep.Rodzaj>
<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
    <ul class="navbar-nav flex-grow-1">
        @foreach (var item in Model)
        {
            <li class="nav-item">
                <a class="nav-link text-dark" asp-area="" asp-controller="Sklep"
asp-action="Index" asp-route-id="@item.IdRodzaju">@item.Nazwa</a>
            </li>
        }
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-
action="Index">Home</a>
        </li>
    </ul>
</div>
```

10. Edytuj plik ***Odnosniki.cshtml*** i uzupełnij jego kod według wzoru:

```
@model IEnumerable<Firma.Data.Data.CMS.Strona>
<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
    <ul class="navbar-nav flex-grow-1">
        @foreach (var item in Model)
        {
            <li class="nav-item">
                <a class="nav-link text-dark" asp-area="" asp-controller="Home"
asp-action="Index" asp-route-id="@item.IdStrony">@item.LinkTytul</a>
            </li>
        }
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Sklep" asp-
action="Index">Sklep</a>
        </li>
    </ul>
</div>
```

Zadanie do samodzielnego wykonania

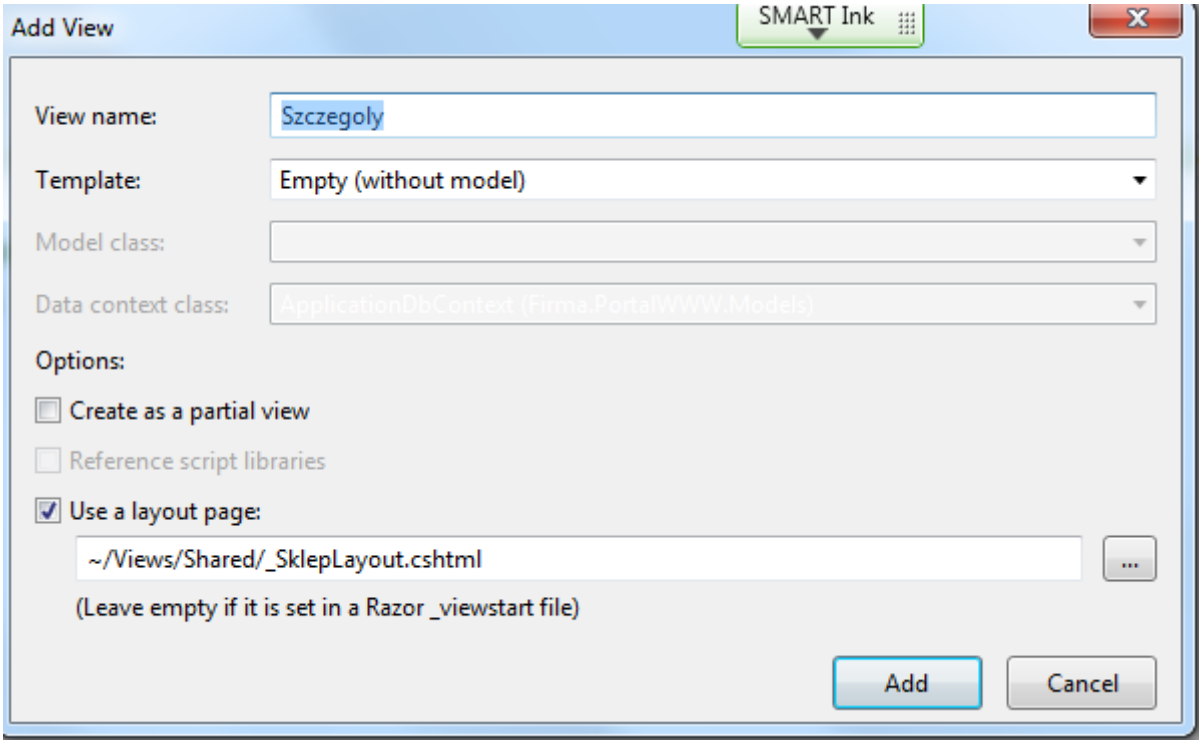
Popraw wygląd prezentowych treści.

Szczegóły towaru

1. Wejdź do pliku ***SklepController.cs*** (katalog ***Controllers***) i dodaj do tego pliku funkcję ***Szczegoly*** według wzoru:

```
public async Task<IActionResult> Szczegoly(int id)
        {
             ViewBag.Rodzaje =await _context.Rodzaj.ToListAsync();
            return View(await _context.Towar.Where(t => t.IdTowaru ==
id).FirstOrDefaultAsync());
        }
```

2. W pliku ***SklepController.cs*** naciśnij prawym klawiszem myszy na nazw funkcji ***Szczegoly*** i wybierz ***Add View (Razor View)***. Ustaw parametry tego widoku według wzoru:



3. Edytuj plik ***Szczegoly.cshtml*** (katalog ***Views->Sklep)*** i zmodyfikuj jego kod według wzoru:

```
@model Firma.Data.Data.Sklep.Towar
@{
    ViewData["Title"] = "Szczegoly towary";
    Layout = "~/Views/Shared/_SklepLayout.cshtml";
}
<div>
    Foto
</div>
<div>
    @Model.Nazwa
</div>
<div>
    @Model.Kod
</div>
<div>
    @Model.Cena
</div>
<div>
    @Model.Opis
</div>
```

Sklep internetowy - Tworzenie komponentu (inny sposób na Rodzaje)

1. Zapoznaj się z dokumentacją:
   https://docs.microsoft.com/pl-pl/aspnet/core/mvc/views/view-components?view=aspnetcore-2.2

2. Ustaw jako projekt startowy **Firma.PortalWWW**.
3. PPM na **Controllers** i **Add Controller (Empty)** `RodzajeMenuComponent.`
   Uzupełnij kod tej klasy według wzoru:

```
using Firma.Data.Data;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Firma.PortalWWW.Controllers
{
    public class RodzajeMenuComponent : ViewComponent
    {
        private readonly FirmaContext _context;
        public RodzajeMenuComponent(FirmaContext context)
        {
            _context = context;
        }
        public async Task<IViewComponentResult> InvokeAsync()
        {
            return View("RodzajeMenuComponent", await _context.Rodzaj.ToListAsync());
        }
    }
}
```

4. PPM na **Views->Sklep** i add **Folder** o nazwie **Components.**
5. PPM na **Components** i add **Folder** o nazwie **RodzajeMenuComponent**.
6. PPM na **RodzajeMenuComponent** I **Add View (RazorView-Empty)** o nazwie **RodzajeMenuComponent.**
   Uzupełnij kod tego widoku według wzoru:

```
@model IEnumerable<Firma.Data.Data.Sklep.Rodzaj>
<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
    <ul class="navbar-nav flex-grow-1">
        @foreach (var item in Model)
        {
            <li class="nav-item">
```

```
                <a class="nav-link text-dark" asp-area="" asp-controller="Sklep"
asp-action="Index" asp-route-id="@item.IdRodzaju">@item.Nazwa</a>
            </li>
        }
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-
action="Index">Home</a>
        </li>
    </ul>
      </div>
```

7. Edytuj plik **Views->Shared->_SklepLayout.cshtml** i zmień jego kod według wzoru:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - Sklep</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css"
/>
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
    <link rel="stylesheet" href="~/Firma.PortalWWW.styles.css" asp-append-
version="true" />
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light
bg-white border-bottom box-shadow mb-3">
            <div class="container-fluid">
                <a class="navbar-brand" asp-area="" asp-controller="Home" asp-
action="Index">Firma.PortalWWW</a>
                <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target=".navbar-collapse" aria-
controls="navbarSupportedContent"
                        aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                @await Component.InvokeAsync("RodzajeMenuComponent")
            </div>
        </nav>
    </header>
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>

    <footer class="border-top footer text-muted">
        <div class="container">
            &copy; 2024 - Firma.PortalWWW - <a asp-area="" asp-
controller="Home" asp-action="Privacy">Privacy</a>
        </div>
    </footer>
```

```
    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>
    @await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```

8. Edytuj **Controller -> SklepController** i zmień jego kod według wzoru:

```
using Firma.Data.Data;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Threading.Tasks;

namespace Firma.PortalWWW.Controllers
{
    public class SklepController : Controller
    {
        private readonly FirmaContext _context;
        public SklepController(FirmaContext context)
        {
            _context = context;
        }

        public async Task<IActionResult> Index(int? id)
        {

            //ViewBag.Rodzaje = await _context.Rodzaj.ToListAsync()
            if (id == null)
            {
                id = 1;
            }

            var towaryDanegoRodzaju=await
_context.Towar.Where(t=>t.IdRodzaju== id).ToListAsync();

            return View(towaryDanegoRodzaju);
        }
        public async Task<IActionResult> Szczegoly(int id)
        {
            //ViewBag.Rodzaje = await _context.Rodzaj.ToListAsync();
            return View(await _context.Towar.Where(t => t.IdTowaru ==
id).FirstOrDefaultAsync());
        }

    }
}
```

Układanie kodu

Przygotowania

1. Otwórz projekt z poprzednich zajęć.
2. Otwórz pobraną solucję. Solucja zawiera projekty ***Firma.Intranet*** i ***Firma.PortalWWW***.

Realizacja

1. PPM na całą solucj i **Add**->**New Project** -> **Class Libary**. Projekt nazwij **Firma.Interfaces**.
2. W projekcie **Firma.Interfaces** PPM na **Dependencies** i **Add Project Reference** i wybierz **Firma.Data**.
3. PPM na projekt **Firma.Interfaces** i **Add**->**New Folder**. Folder nazwij **CMS**.
4. PPM na **CMS i Add** -> **Class** -> **Interface.** Interface nazwij **IAktualnoscService**. Utwórz kod tego interfejsu według wzoru:

```
using Firma.Data.Data.CMS;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Firma.Interfaces.CMS
{
    public interface IAktualnoscService
    {
        Task<IList<Aktualnosc>> GetAktualnosciByPozycjaTake(int ilePobrac);
        Task<Aktualnosc?> GetAktualnosc(int idAktualnosci);
    }
}
```

5. PPM na całą solucję i **Add**->**New Project** -> **Class Libary**. Projekt nazwij **Firma.Services**.
6. W projekcie **Firma. Services** PPM na **Dependencies** i **Add Project Reference** i wybierz **Firma.Data**.
7. W projekcie **Firma. Services** PPM na **Dependencies** i **Add Project Reference** i wybierz **Firma.Interfaces**.
8. PPM na projekt **Firma. Services** i **Add**->**New Folder**. Folder nazwij **Abstrakcja**.
9. PPM na **Abstrakcja i Add** -> **Class.** Klasę nazwij **BaseService**. Utwórz kod tego serwisu według wzoru:

```
using Firma.Data.Data;
using Microsoft.Extensions.Configuration;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Firma.Services.Abstrakcja
{
    public abstract class BaseService
    {
        protected readonly FirmaContext _context;
        //protected readonly IServiceProvider serviceProvider;
        //protected readonly Lazy<IUzytkownikService> uzytkownik;
        //protected readonly Lazy<ICacheProvider> cacheProvider;
        //protected readonly Lazy<DapperProvider> dapperProvider;
        //protected readonly Lazy<IConfiguration> configuration;
        //protected readonly Lazy<IWyjatekObslugaService> wyjatekObslugaService;

        public BaseService(FirmaContext context)
        {
            _context = context;
        }
    }
}
```

10. PPM na projekt **Firma. Services** i **Add**->**New Folder**. Folder nazwij **CMS**.
11. PPM na **CMS i Add** -> **Class.** Klasę nazwij **AktualnoscService**. Utwórz kod tego serwisu według wzoru:

```
using Firma.Data.Data;
using Firma.Data.Data.CMS;
using Firma.Interfaces.CMS;
using Firma.Services.Abstrakcja;
using Microsoft.EntityFrameworkCore;

namespace Firma.Services.CMS
{
    public class AktualnoscService : BaseService, IAktualnoscService
    {
        public AktualnoscService(FirmaContext context)
        : base(context)
        {
        }

        public async Task<Aktualnosc?> GetAktualnosc(int idAktualnosci)
        {
            var aktualnosc = await _context.Aktualnosc.FirstOrDefaultAsync(a =>
a.IdAktualnosci == idAktualnosci);
            return aktualnosc;
        }

        public async Task<IList<Aktualnosc>> GetAktualnosciByPozycjaTake(int
ilePobrac)
        {
```

```
            var aktualnosci = await _context.Aktualnosc.OrderByDescending(a =>
a.Pozycja).Take(ilePobrac).ToListAsync();
            return aktualnosci;
        }
    }
}
```

12. W projekcie **Firma.PortalWWW** PPM na **Dependencies** i **Add Project Reference** i wybierz **Firma.Services**.
13. PPM na projekt **Firma.PortalWWW** i **Add**-> **Class**. Klasę nazwij **DependencyInjectionFactory**. Utwórz kod tej klasy według wzoru:

```
using Firma.Interfaces.CMS;
using Firma.Services.CMS;

namespace Firma.PortalWWW
{
    public static class DependencyInjectionFactory
    {
        public static void Resolve(IServiceCollection services, IConfiguration
conf)
        {
            services.AddScoped<IAktualnoscService, AktualnoscService>();
        }
    }
}
```

14. W projekcie **Firma.PortalWWW** edytuj plik **Program.cs** i zmodyfikuj jego górną część według wzoru:

```
using Firma.Data.Data;
using Firma.PortalWWW;
using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);
builder.Services.AddDbContext<FirmaContext>(options =>


options.UseSqlServer(builder.Configuration.GetConnectionString("FirmaContext")));

DependencyInjectionFactory.Resolve(builder.Services, builder.Configuration);
// Add services to the container.
builder.Services.AddControllersWithViews();

…
```

15. W projekcie **Firma.PortalWWW** edytuj plik **Controllers** -> **HomeController** i zmodyfikuj jego kod według wzoru:

```
using Firma.Data.Data;
using Firma.Interfaces.CMS;
using Firma.PortalWWW.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Diagnostics;

namespace Firma.PortalWWW.Controllers
{
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;
        private readonly FirmaContext _context;
        private readonly IAktualnoscService _aktualnoscService;

        public HomeController(ILogger<HomeController> logger, FirmaContext
context, IAktualnoscService aktualnoscService)
        {
            _logger = logger;
            //Uwaga context nie będzie potrzebny jak dodamy podobny serwis do
strony
            _context = context;
            _aktualnoscService = aktualnoscService;
        }

        public async Task<IActionResult> Index(int? id)
        {
            ViewBag.ModelStrony =await
_context.Strona.OrderBy(s=>s.Pozycja).ToListAsync();
            //ViewBag.ModelAktualnosci = await
_context.Aktualnosc.OrderByDescending(a => a.Pozycja).Take(3).ToListAsync();
            ViewBag.ModelAktualnosci = await
_aktualnoscService.GetAktualnosciByPozycjaTake(3);
            if (id == null)
                id = 1;//koniecznie powinna być strona o id=1 w bazie
            var item = await _context.Strona.FindAsync(id);
            return View(item);
        }

        public IActionResult OFirmie()
        {
            return View();
        }

        public IActionResult Privacy()
        {
            return View();
        }

        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None,
NoStore = true)]
        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId = Activity.Current?.Id ??
HttpContext.TraceIdentifier });
        }
    }
}
```

16. W projekcie **Firma.PortalWWW** edytuj plik **Controllers** -> **AktualnoscController** i zmodyfikuj jego kod według wzoru:

```
using Firma.Data.Data;
using Firma.Interfaces.CMS;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace Firma.PortalWWW.Controllers
{
    public class AktualnoscController : Controller
    {
        //Uwaga context nie będzie potrzebny jak dodamy podobny serwis do strony
        private readonly FirmaContext _context;
        private readonly IAktualnoscService _aktualnoscService;
        public AktualnoscController(FirmaContext context, IAktualnoscService
aktualnoscService)
        {
            //Uwaga context nie będzie potrzebny jak dodamy podobny serwis do
strony
            _context = context;
            _aktualnoscService = aktualnoscService;
        }
        public async Task<IActionResult> Index(int id)
        {
            ViewBag.ModelStrony = await _context.Strona.OrderBy(s =>
s.Pozycja).ToListAsync();
            //ViewBag.ModelAktualnosci = await
_context.Aktualnosc.OrderByDescending(a => a.Pozycja).Take(3).ToListAsync();
            ViewBag.ModelAktualnosci = await
_aktualnoscService.GetAktualnosciByPozycjaTake(3);
            //var item = await _context.Aktualnosc.FirstOrDefaultAsync(a =>
a.IdAktualnosci == id);
            var item = await _aktualnoscService.GetAktualnosc(id);
            if (item == null)
            {
                return NotFound();
            }
            return View(item);
        }
    }
}
```

Widok Index do tego kontrolera pozostaje bez zmian:

```
@model IEnumerable<Firma.Data.Data.CMS.Aktualnosc>
<div class="container align-content-center">
    <div class="container text-center">
        <div class="row">
            @foreach(var item in Model)
            {
                <div class="col-12 col-md-6 col-lg-3">
                    <h2>@item.Tytul</h2>
                    <p>
                        @item.Tresc
```

```
                    </p>
                    <a class="btn btn-outline-primary" asp-area="" asp-
controller="Aktualnosc" asp-action="Index" asp-route-
id="@item.IdAktualnosci">@item.LinkTytul</a>
                </div>
            }
        </div>
    </div>
</div>
```

17. Podobnie utwórz interfejs, serwis i zmień kontrolery dla **Strony**.

```
namespace Firma.Interfaces.CMS
{
    public interface IStronaService
    {
        Task<IList<Strona>> GetStrony();
        Task<Strona?> GetStrona(int? id);
    }
}
```

```
namespace Firma.Services.CMS
{
    public class StronaService : BaseService, IStronaService
    {
        public StronaService(FirmaContext context)
            : base(context)
        {
        }

        public async Task<IList<Strona>> GetStrony()
        {
            var strony= await _context.Strona.OrderBy(s =>
s.Pozycja).ToListAsync();
            return strony;
        }
        public async Task<Strona?> GetStrona(int? id)
        {
            var strona = await _context.Strona.FindAsync(id);
            return strona;
        }
    }
```

```
namespace Firma.PortalWWW
{
    public static class DependencyInjectionFactory
    {
        public static void Resolve(IServiceCollection services,IConfiguration
conf)
        {
            services.AddScoped<IAktualnoscService,AktualnoscService>();
            services.AddScoped<IStronaService, StronaService>();
        }
    }
```

```
}
```

```
namespace Firma.PortalWWW.Controllers
{
    public class AktualnoscController : Controller
    {
        private readonly IAktualnoscService _aktualnoscService;
        private readonly IStronaService _stronaService;
        public AktualnoscController(IAktualnoscService aktualnoscService,
IStronaService stronaService)
        {

            _aktualnoscService = aktualnoscService;
            _stronaService = stronaService;
        }
        public async Task<IActionResult> Index(int id)
        {
            ViewBag.ModelStrony = await _stronaService.GetStrony();
            ViewBag.ModelAktualnosci = await
_aktualnoscService.GetAktualnosciByPozycjaTake(3);
            var item = await _aktualnoscService.GetAktualnosc(id);
            if (item == null)
            {
                return NotFound();
            }
            return View(item);
        }

    }
```

```
namespace Firma.PortalWWW.Controllers;

public class HomeController : Controller
{
    private readonly ILogger<HomeController> _logger;
    private readonly IStronaService _stronaService;
    private readonly IAktualnoscService _aktualnoscService;
    public HomeController(ILogger<HomeController> logger, IAktualnoscService
aktualnoscService, IStronaService stronaService)
    {
        _logger = logger;
        _stronaService = stronaService;
        _aktualnoscService = aktualnoscService;
    }

    public async Task<IActionResult> Index(int? id)
    {

        ViewBag.ModelStrony = await _stronaService.GetStrony();

        ViewBag.ModelAktualnosci = await
_aktualnoscService.GetAktualnosciByPozycjaTake(4);
        if (id == null)
        {
```

```
            id = 1;
        }
        var item = await _stronaService.GetStrona(id);
        return View(item);
    }

    public IActionResult OpisFirmy()
    {
        return View();
    }
    public IActionResult HistoriaFirmy()
    {
        return View();
    }

    public IActionResult Privacy()
    {
        return View();
    }

    [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore =
true)]
    public IActionResult Error()
    {
        return View(new ErrorViewModel { RequestId = Activity.Current?.Id ??
HttpContext.TraceIdentifier });
    }
}
```

18. Utwórz podobnie interfejsy, serwisy i zmień kontrolery dla **Towarów** i **Rodzajów** według wzoru:

Interfejsy

```
namespace Firma.Interfaces.Sklep
{
    public interface IRodzajService
    {
        Task<IList<Rodzaj>> GetRodzaje();
    }
}
```

```
namespace Firma.Interfaces.Sklep
{
    public interface ITowarService
    {
        Task<Towar?> GetTowar(int idTowaru);
        Task<IList<Towar>> GetTowaryDanegoRodzaju(int idRodzaju);
    }
}
```

Serwisy

```
namespace Firma.Services.Sklep
{
    public class RodzajService : BaseService, IRodzajService
    {
        public RodzajService(FirmaContext context)
        : base(context)
        {
        }

        public async Task<IList<Rodzaj>> GetRodzaje()
        {
            var rodzaje = await _context.Rodzaj.ToListAsync();
            return rodzaje;
        }
    }
}

===

namespace Firma.Services.Sklep
{
    public class TowarService : BaseService, ITowarService
    {
        public TowarService(FirmaContext context)
        : base(context)
        {
        }

        public async Task<Towar?> GetTowar(int idTowaru)
        {
            var towar = await _context.Towar.Where(t => t.IdTowaru ==
idTowaru).FirstOrDefaultAsync();
            return towar;
        }

        public async Task<IList<Towar>> GetTowaryDanegoRodzaju(int idRodzaju)
        {
            var towary = await _context.Towar.Where(t => t.IdRodzaju ==
idRodzaju).ToListAsync();
            return towary;
        }

    }
}
```

DependencyInjectionFactory

```
namespace Firma.PortalWWW
{
    public static class DependencyInjectionFactory
    {
        public static void Resolve(IServiceCollection services, IConfiguration
conf)
        {
            services.AddScoped<IAktualnoscService, AktualnoscService>();
            services.AddScoped<IStronaService, StronaService>();
            services.AddScoped<ITowarService, TowarService>();
            services.AddScoped<IRodzajService, RodzajService>();
        }
    }
}
```

Zmiany w kontrolerach

```
namespace Firma.PortalWWW.Controllers
{
    public class SklepController : Controller
    {
        private readonly ITowarService _towarService;
        public SklepController(ITowarService towarService)
        {
            _towarService = towarService;
        }
        public async Task<IActionResult> Index(int? id)
        {
            if (id == null)
            {
                id = 1;
            }
            var towaryDanegoRodzaju=await
_towarService.GetTowaryDanegoRodzaju(id.Value);
            return View(towaryDanegoRodzaju);
        }
        public async Task<IActionResult> Szczegoly(int id)
        {
            return View(await _towarService.GetTowar(id));
        }
    }
}
```

```
namespace Firma.PortalWWW.Controllers
{
    public class RodzajeMenuComponent : ViewComponent
    {
        private readonly IRodzajService _rodzajService;
        public RodzajeMenuComponent(IRodzajService rodzajService)
        {
```

```
            _rodzajService = rodzajService;
        }
        public async Task<IViewComponentResult> InvokeAsync()
        {
            return View("RodzajeMenuComponent", await
_rodzajService.GetRodzaje());
        }
    }
}
```