Uniwersytet Bielsko-Bialski

LABORATORIUM

Programowanie dla Internetu w technologii ASP.NET

Sprawozdanie nr 5

Warstwa ViewModel

GRUPA: 2B / SEMESTR: 6 / ROK: 3

Kacper Lizak / 59443

Cel ćwieczenia

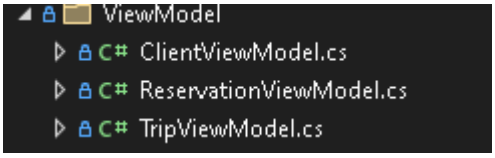
Cel ćwiczenia polega na implementacji warstwy ViewModel w projekcie ASP.NET w celu poprawy separacji logiki prezentacji od logiki biznesowej oraz ułatwienia przekazywania danych między widokami a kontrolerami.

Wprowadzenie

Warstwa ViewModel w aplikacji webowej pełni rolę pośrednika między warstwą prezentacji a warstwą biznesową, zapewniając odpowiednie dane do wyświetlenia w widoku bezpośrednio niezależnie od modelu domenowego. Jest to szczególnie przydatne w przypadku, gdy widok wymaga danych z wielu różnych modeli biznesowych lub gdy potrzebne są dane w innym formacie niż te dostarczone przez model domenowy.

Realizacja

Utworzenie folderu dla ViewModel'ów



Przykład ViewModel'u

```
public class ClientViewModel
{
    public int IdClient { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public int? Pesel { get; set; }
    public string Email { get; set; }
    public string? Phone { get; set; }

    public IEnumerable<ReservationViewModel>? Reservations { get; set; }
}
```

Modyfikacja Controlera dla Client

```
public async Task<IActionResult> Index()
{
    var clients = await _clientServices.GetAllAsync();
    var clientsList = clients.Select(client => new ClientViewModel
    {
        IdClient = client.IdClient,
        FirstName = client.FirstName,
        LastName = client.LastName,
        Email = client.Email,
        Phone = client.Phone
    }).ToList();
    return View(clientsList);
}
```

```
public async Task<IActionResult> Details(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var client = await _clientServices.GetByIdAsync(id);
    if (client == null)
    {
        return NotFound();
    }
    var clientViewModel = new ClientViewModel
    {
        IdClient = client.IdClient,
        FirstName = client.FirstName,
        LastName = client.LastName,
        Email = client.Email,
        Phone = client.Phone
    };

    return View(clientViewModel);
}
```

```
public async Task <IActionResult>
Create([Bind("IdClient,FirstName,LastName,Email,Phone")] ClientViewModel
clientViewModel)
{
    if (ModelState.IsValid)
    {
        var client = new Client
        {
           FirstName = clientViewModel.FirstName,
           LastName = clientViewModel.LastName,
```

```
            Email = clientViewModel.Email,
            Phone = clientViewModel.Phone,
            IdClient = clientViewModel.IdClient


        };
        await _clientServices.InsertAsync(client);
        await _clientServices.SaveAsync();
        return RedirectToAction(nameof(Index));
    }
    return View(clientViewModel);
}
```

```
public async Task<IActionResult> Edit(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var client = await _clientServices.GetByIdAsync(id);
    if (client == null)
    {
        return NotFound();
    }
    var clientViewModel = new ClientViewModel
    {
        IdClient = client.IdClient,
        FirstName = client.FirstName,
        LastName = client.LastName,
        Email = client.Email,
        Phone = client.Phone
    };
    return View(clientViewModel);
}
```

```
 public async Task <IActionResult> Edit(int id,
[Bind("IdClient,FirstName,LastName,Email,PhoneNumber")] ClientViewModel
clientViewModel)
 {
     if (id != clientViewModel.IdClient)
     {
         return NotFound();
     }

     if (ModelState.IsValid)
     {
         var client = new Client
         {
```

```
                IdClient = clientViewModel.IdClient,
                FirstName = clientViewModel.FirstName,
                LastName = clientViewModel.LastName,
                Email = clientViewModel.Email,
                Phone = clientViewModel.Phone
            };
            try
            {
                _clientServices.Update(client);
                await _clientServices.SaveAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!ClientExists(client.IdClient))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }
            return RedirectToAction(nameof(Index));
        }
        return View(clientViewModel);
    }
```

Modyfikacja View dla Client:

Dla index:

```
@model IEnumerable<TripsS.ViewModel.ClientViewModel>
```

dla pozostałych:

```
@model TripsS.ViewModel.ClientViewModel
```

Wnioski

Wnioski: Implementacja warstwy ViewModel w projekcie ASP.NET umożliwia odseparowanie logiki prezentacji od logiki biznesowej, co prowadzi do łatwiejszego zarządzania kodem i unikania nadmiernego powiązania między poszczególnymi komponentami aplikacji. Dzięki zastosowaniu ViewModeli możliwe jest również zachowanie elastyczności w dostarczaniu danych do widoków, co ułatwia rozwój i utrzymanie aplikacji w przyszłości.