

Mariusz Wieczorek

mariusz.wieczorek@kabat.pl

Repozytorium, UnitOfWork, Serwisy

Jak w praktyce użyć tych wzorców

asp.net Core

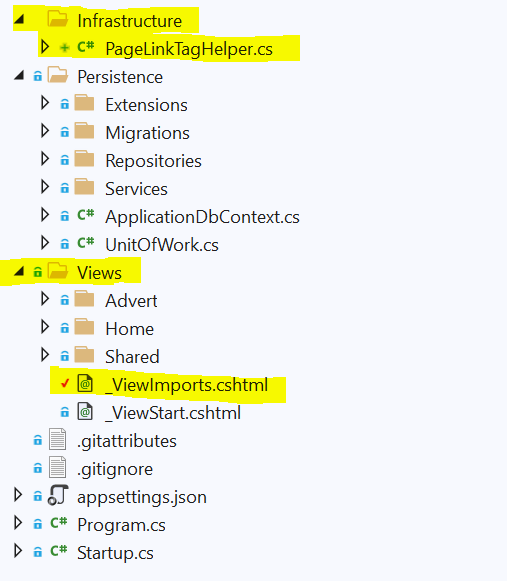
Klasa Atrybutu Pomocniczego

**Dodanie klasy atrybutu pomocniczego**

Tworzymy katalog **Infrastructure** - w nim przechowujemy klasy komponentów aplikacji nie powiązanych z domeną

W tym katalogu tworzymy plik: **PageLinkTagHelper.cs**

Atrybuty pomocnicze znaczników w odróżnieniu od widoków i kontrolerów muszą być rejestrowanie robimy to w pliku: **Views\\_ViewImports.cshtml**



|  |
| --- |
| @using MyAdvert  @using MyAdvert.Models  @using MyAdvert.Core.ViewModels  @addTagHelper \*, Microsoft.AspNetCore.Mvc.TagHelpers  @addTagHelper MyAdvert.Infrastructure.\*, MyAdvert |

**Dodawanie Stronicowania ( paginacji )**

Aby zapewnić możliwość prawidłowego działania atrybutów pomocniczych znaczników musimy przekazać informacje o liczbie dostępnych stron, bieżącej stronie oraz całkowitej liczbie pozycji w repozytorium.

Najprostszym sposobem zrealizowania tego zadania jest utworzenie modelu widoku, który będzie używany do przekazywania danych między kontrolerem i widokiem.  
Tworzymy katalog Models/ViewModels i tworzymy w nim plik klasy PagingInfo.cs

|  |
| --- |
| namespace MyAdvert.Core.ViewModels  {  public class PagingInfo  {  public int TotalItems { get; set; }  public int ItemsPerPage { get; set; }  public int CurrentPage { get; set; }  public int TotalPages =>  (int)Math.Ceiling((decimal)TotalItems / ItemsPerPage);  }  } |

Musimy przekazać obiekt klasy **PagingInfo** do widoku, dołączamy go więc do modelu widoku listy ogłoszeń.

|  |
| --- |
| namespace MyAdvert.Core.ViewModels  {  public class AdvertsViewModel  {  public IEnumerable<Advert> Adverts { get; set; }  public IEnumerable<Category> Categories { get; set; }  public FilterAdverts FilterAdverts { get; set; }  public PagingInfo PagingInfo { get; set; }  }  } |

Metoda w Repozytorium

|  |
| --- |
| public IEnumerable<Advert> GetAdverts(FilterAdverts filterTasks, PagingInfo pagingInfo)  {  var adverts = \_context.Adverts  .Include(x => x.Category).AsQueryable();  if (filterTasks.IsActive == true)  adverts = adverts.Where(x => x.IsActive == filterTasks.IsActive);  if (filterTasks.CategoryId != 0)  adverts = adverts.Where(x => x.CategoryId == filterTasks.CategoryId);  if (!string.IsNullOrWhiteSpace(filterTasks.Title))  adverts = adverts.Where(x => x.Title.Contains(filterTasks.Title));  if (pagingInfo != null)  {  adverts = adverts  .Skip((pagingInfo.CurrentPage - 1) \* pagingInfo.ItemsPerPage)  .Take(pagingInfo.ItemsPerPage);  }  return adverts.OrderBy(x => x.StartDate).ToList();  } |

Wyświetlanie łączy stron  
Mamy model widoku, który zawiera dane stronicowania  
został zaktualizowany kontroler aby te dane zostały przekazane do widoku  
czyli metoda List wywołuje metodę View przekazując do niej repozytorium oraz model widoku  
Pozostało dodanie elementu HTML, który będzie przetwarzany przez atrybut pomocniczy znacznika  
w celu utworzenia łączy stron

do pliku Views\Product\List.cshtml dodajemy:

<div page-model="@Model.PagingInfo" page-action="List"></div>

jest to bardzo prosty zestaw linków bez żadnych ozdobników