

Лабораторная работа №3

Работа с данными

1. Цель работы.

Изучение механизмов обмена данными между контроллером и представлением.

2. Задача работы

Научиться подготавливать данные в контроллере и передавать их представлению для отображения.

Время выполнения работы: 2 часа

3. Выполнение работы

Используйте проект из лабораторной работы №2.

3.1. Описание предметной области

Выберите **любую** предметную область.

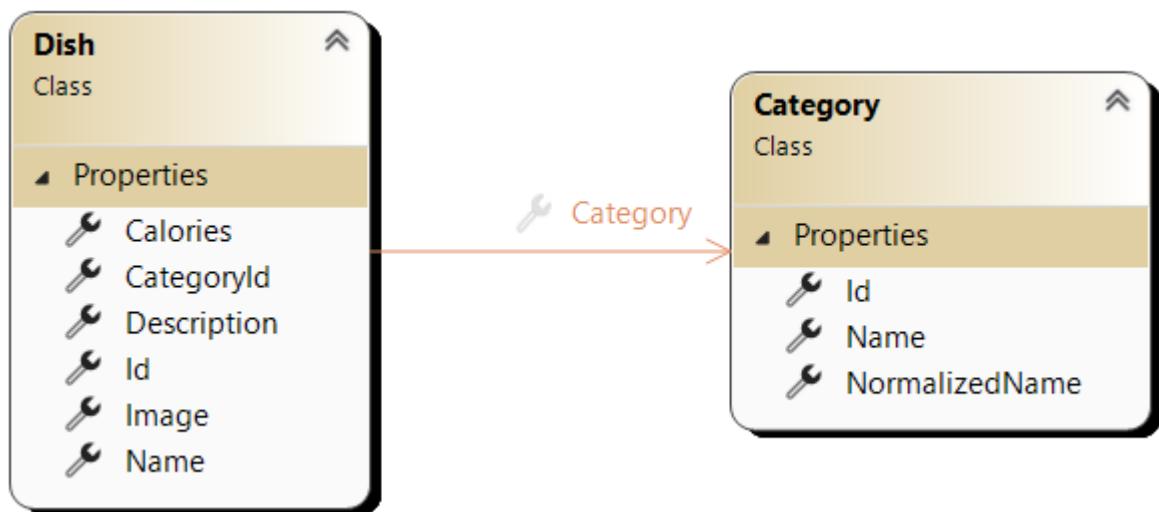
Добавьте в решение новый проект – библиотеку классов .Net. Имя проекта **XXX.Domain**, где XXX – имя вашего решения.

В созданном проекте создайте папку **Entities**.

Для одной сущности из выбранной предметной области в папке Entities проекта создайте класс, содержащий следующие свойства:

- ID – уникальный номер;
- Название – короткое название конкретного объекта;
- Описание – дополнительное описание конкретного объекта;
- Категория – свойство для объединения объектов в группы;
- Цена/Вес/Расстояние – выберите любой параметр, который можно в дальнейшем обработать математически, например, просуммировать;
- Изображение – путь к файлу изображения объекта
- Mime тип изображения

В той же папке создайте класс, описывающий категорию объекта. Отношение должно быть один-ко-многим: одна категория описывает много объектов (см. Рисунок)



Примечание: в классе Category свойство **NormalizedNome** – это имя на английском языке, в нотации «kebab», которое будет использоваться как часть маршрута для фильтрации по категориям. Так адрес запроса будет более понятным, по сравнению с передачей **Id** выбранной категории

Примечание: здесь и далее используется предметная область – меню кафе. Сущностные классы – блюдо и категория блюда (первые блюда, салаты, напитки и т.д.)

Примечание: свойство **Image** и навигационное свойство **Category** в классе **Dish** описаны как nullable.

В проекте XXX.UI сделайте ссылку на созданную библиотеку классов
В файл _ViewImports.cshtml подключите пространство имен
`@using XXX.Domain.Entities`
Добавьте в проект файл GlobalUsings.cs. Укажите в нем глобальное подключение пространства имен `XXX.Domain.Entities`

3.2. Вспомогательные классы

В проект **XXX.Domain** добавьте папку Models

Данные контроллером будут получаться с помощью сервисов.

В папке Models опишите вспомогательные классы:

responseData – класс, описывающий формат данных, получаемых от сервисов:

```
public class responseData<T>
{
    // запрашиваемые данные
    public T? Data { get; set; }
    // признак успешного завершения запроса
    public bool Successfull { get; set; } = true;
    // сообщение в случае неуспешного завершения
    public string? ErrorMessage { get; set; }
    /// <summary>
    /// Получить объект успешного ответа
    /// </summary>
    /// <param name="data">передаваемые данные</param>
    /// <returns></returns>
    public static responseData<T> Success(T data)
    {
        return new responseData<T> { Data = data };
    }
    /// <summary>
    /// Получение объекта ответа с ошибкой
    /// </summary>
    /// <param name="message">Сообщение об ошибке</param>
    /// <param name="data">Передаваемые данные</param>
    /// <returns></returns>
    public static responseData<T> Error(string message,
                                         T? data=default)
    {
        return new responseData<T> { ErrorMessage = message,
Successfull = false, Data = data };
    }
}
```

ListModel – класс, описывающий данные, используемые при получении списка объектов:

```
public class ListModel<T>
{
    // запрошенный список объектов
    public List<T> Items { get; set; } = new();
    // номер текущей страницы
```

```

public int CurrentPage { get; set; } = 1;
// общее количество страниц
public int TotalPages { get; set; } = 1;
}

```

Свойства CurrentPage и TotalPages понадобятся при разбиении общего списка на страницы

Укажите в файле GlobalUsings.cs глобальное подключение пространства имен `XXX.Domain.Models`

3.3. Подготовка для регистрации пользовательских сервисов

Для регистрации сервисов опишите расширяющий метод для класса `WebAppBuilder`.

Добавьте в проект `XXX.UI` папку `Extensions`.

Добавьте в папку `Extensions` класс `HostingExtensions`, в котором опишите расширяющий метод `RegisterCustomServices`:

```

public static class HostingExtensions
{
    public static void RegisterCustomServices(
        this WebApplicationBuilder builder)
    {
    }
}

```

В этом методе будем регистрировать созданные сервисы.

В классе `Program` вызовите метод `RegisterCustomServices`:

```
builder.RegisterCustomServices();
```

3.4. Описание сервисов

Для регистрации сервисов опишите расширяющий метод для класса `WebAppBuilder`. Для

В проект `XXX.UI` добавьте папку `Services`

Методы сервисов должны возвращать объекты класса `responseData`.

В папку Services добавьте папку CategoryService (имя папки выберите в зависимости от выбранной вами предметной области). В ней создайте интерфейс ICategoryService (имя интерфейса выберите в зависимости от выбранной вами предметной области), который описывает метод получения списка всех категорий:

```
public interface ICategoryService
{
    /// <summary>
    /// Получение списка всех категорий
    /// </summary>
    /// <returns></returns>
    public Task<ResponseData<List<Category>>> GetCategoryListAsync();
}
```

В папке CategoryService опишите класс MemoryCategoryService, реализующий интерфейс ICategoryService. Это сервис, имитирующий работу с реальными данными. Метод GetCategoryListAsync должен вернуть коллекцию объектов класса Category:

```
public class MemoryCategoryService : ICategoryService
{
    public Task<ResponseData<List<Category>>> GetCategoryListAsync()
    {
        var categories = new List<Category>
        {
            new Category {Id=1, Name="Стarterы",
NormalizedName="starters"},  

            new Category {Id=2, Name="Салаты",
NormalizedName="salads"},  

            new Category {Id=3, Name="Супы", NormalizedName="soups"},  

            new Category {Id=4, Name="Основные блюда",
NormalizedName="main-dishes"},  

            new Category {Id=5, Name="Напитки",
NormalizedName="drinks"},  

            new Category {Id=6, Name="Десерты",
NormalizedName="desserts"}  

        };
        var result = ResponseData<List<Category>>.Success(categories);
        return Task.FromResult(result);
    }
}
```

Зарегистрируйте сервис ICategoryService как scoped сервис в классе HostingExtensions.

Укажите в файле GlobalUsings.cs глобальное подключение пространства имен `XXX.UI.Services.CategoryService`

В папку Services добавьте папку ProductService (имя папки выберите в зависимости от выбранной вами предметной области). В ней создайте интерфейс IPIService (имя интерфейса выберите в зависимости от выбранной вами предметной области), который описывает функции, необходимые для работы приложения:

```
public interface IPIService
{
    /// <summary>
    /// Получение списка всех объектов
    /// </summary>
    /// <param name="categoryNormalizedNames">нормализованное имя категории для фильтрации</param>
    /// <param name="pageNo">номер страницы списка</param>
    /// <returns></returns>
    public Task<ResponseData<ListModel<Dish>>> GetProductListAsync(string? categoryNormalizedNames, int pageNo=1);

    /// <summary>
    /// Поиск объекта по Id
    /// </summary>
    /// <param name="id">Идентификатор объекта</param>
    /// <returns>Найденный объект или null, если объект не найден</returns>
    public Task<ResponseData<Dish>> GetProductByIdAsync(int id);

    /// <summary>
    /// Обновление объекта
    /// </summary>
    /// <param name="id">Id изменяемого объекта</param>
    /// <param name="product">объект с новыми параметрами</param>
    /// <param name="formFile">Файл изображения</param>
    /// <returns></returns>
    public Task UpdateProductAsync(int id, Dish product, IFormFile? formFile);
    /// <summary>
    /// Удаление объекта
    /// </summary>
    /// <param name="id">Id удаляемого объекта</param>
    /// <returns></returns>
    public Task DeleteProductAsync(int id);

    /// <summary>
    /// Создание объекта
    /// </summary>
    /// <param name="product">Новый объект</param>
    /// <param name="formFile">Файл изображения</param>
    /// <returns>Созданный объект</returns>
}
```

```
public Task<ResponseData<Dish>> CreateProductAsync(Dish product, IFormFile?
formFile);
}
```

В папке ProductService опишите класс MemoryProductService, реализующий интерфейс IProductService. Это сервис, имитирующий работу с реальными данными. На данном этапе достаточно реализовать метод:

- **GetProductListAsync**

Данные нужно описать в коде класса MemoryProductService в виде коллекций. В конструкторе класса нужно заполнить коллекции данными. Для установления связи с объектами класса Category внедрите в конструктор объект ICategoryService.

Для правильной работы приложения поместите в папку wwwroot/Images файлы изображений объектов коллекции.

Пример заполнения коллекции данными:

```
public class MemoryProductService : IProductService
{
    List<Dish> _dishes;
    List<Category> _categories;

    public MemoryProductService(ICategoryService categoryService)
    {
        _categories=categoryService.GetCategoryListAsync()
            .Result
            .Data;
        SetupData();
    }
    . .

    /// <summary>
    /// Инициализация списков
    /// </summary>
    private void SetupData()
    {
        _dishes = new List<Dish>
        {
            new Dish {Id = 1, Name="Суп-харчо",
                      Description="Очень острый, невкусный",
                      Calories =200, Image="../Images/Суп.jpg",
                      Category=
                _categories.Find(c=>c.NormalizedName.Equals("soups"))},
            new Dish { Id = 2, Name="Борщ",
                      Description="Много сала, без сметаны",
                      Calories =330, Image="../Images/Борщ.jpg",

```

```

        Category=
    _categories.Find(c=>c.NormalizedName.Equals("soups"))},
    .
    .
    .
}

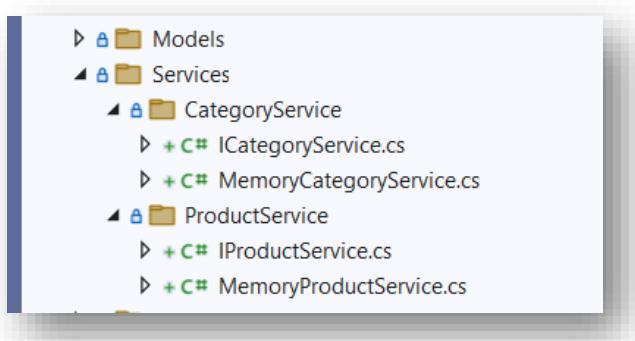
}

```

Зарегистрируйте сервис IProductService как scoped сервис в классе HostingExtensions.

Укажите в файле GlobalUsings.cs глобальное подключение пространства имен `XXX.UI.Services.ProductService`

Вид проекта:



Примечание: при инициализации списка объектов URL изображения (свойство Image) выглядит так: «images/xxx». Для корректной работы необходимо в перке `wwwroot` создать папку **Images** и в созданную папку поместить файлы изображений.

3.5. Вывод списка объектов на страницу приложения

В папку Controllers добавьте контроллер Product (имя класса – ProductController).

Внедрите в конструктор контроллера объекты типа IProductService и ICategoryService.

В методе Index передайте представлению список объектов, например:

```
public async Task<IActionResult> Index()
```

```
{
    var productResponse =
        await _service.GetProductListAsync(null);
    if(!productResponse.Successfull)
        return NotFound(productResponse.ErrorMessage);
    return View(productResponse.Data.Items);
}
```

Добавьте представление Index для метода Index контроллера Product. Используйте шаблон «List». В качестве модели укажите объект вашей предметной области.

В VS Code или VisualStudio for MAC для генерирования представлений необходимо:

a) Установить Code generation tool:

dotnet tool install -g dotnet-aspnet-codegenerator

b) Установить в проект NuGet пакет

dotnet add package Microsoft.VisualStudio.Web.CodeGeneration.Design

Использование **dotnet-aspnet-codegenerator** описано здесь:

<https://learn.microsoft.com/en-us/aspnet/core/fundamentals/tools/dotnet-aspnet-codegenerator?view=aspnetcore-6.0>

Пример создания страницы по шаблону Edit:

Класс модели

dotnet-aspnet-codegenerator view MyEdit Edit -m Movie -dc MovieContext -outDir Views/Movies

В полученном представлении замените вывод имени файла изображения на вывод самого изображения:

Запустите проект. Перейдите на страницу «Каталог». Убедитесь, что страница отображается правильно, и текущий пункт меню выделен:

Пример страницы каталога:

Index						
Create New						
Id	Name	Description	Calories	Image	CategoryId	
1	Суп-харчо	Очень острый, невкусный	200		3	Edit Details Delete
2	Борщ	Много сала, без сметаны	330		3	Edit Details Delete
3	Котлета пожарская	Хлеб - 80%, Морковь - 20%	635		4	Edit Details Delete

3.6. Оформление списка объектов

Требуется оформить список объектов в виде карт (см. <https://getbootstrap.com/docs/5.3/components/card/#card-groups>), по 3 карты в одном ряду.

Один элемент списка должен содержать:

- изображение объекта
- название объекта (Card title)
- описание объекта
- кнопку «Добавить в корзину»

Кнопка «Добавить в корзину» должна передавать управление методу Add контроллера Cart (будут добавлены в дальнейшем). Также должны передаваться **Id** выбранного элемента и параметр **returnurl** – адрес текущей страницы для возврата к списку после добавления в корзину.

Адрес текущей страницы можно получить так:

```
var request = ViewContext.HttpContext.Request;
var returnUrl = request.Path + request.QueryString.ToUriComponent();
```

Пример оформления страницы:

WebLabsV03 Лб 1 Каталог Администрирование 00.0 руб. (0) ⚙

Блюда дня



Суп-харчо
Очень острый, невкусный
200 калорий

Add to cart 



Борщ
Много сала, без сметаны
330 калорий

Add to cart 



Котлета пожарская
Хлеб - 80%, Морковь - 20%
635 калорий

Add to cart 



Примечание: для того, чтобы карты имели одинаковый размер, необходимо, чтобы изображения объектов имели одинаковые соотношения сторон.

Убедитесь, что при наведении курсора на кнопку добавления в корзину указывается правильный адрес:



Макароны по-флотски
С охотничьей колбаской
Add to cart 



Компот
Быстро растворимый, 2 литра
Add to cart 

https://localhost:7001/cart/add/5?returnurl=%2Fproduct

3.7. Добавление фильтра по категориям

Требуется предоставить возможность выбора объектов по категориям. Выбор представить в виде выпадающего списка, который можно оформить в виде компонента Bootstrap

Dropdown

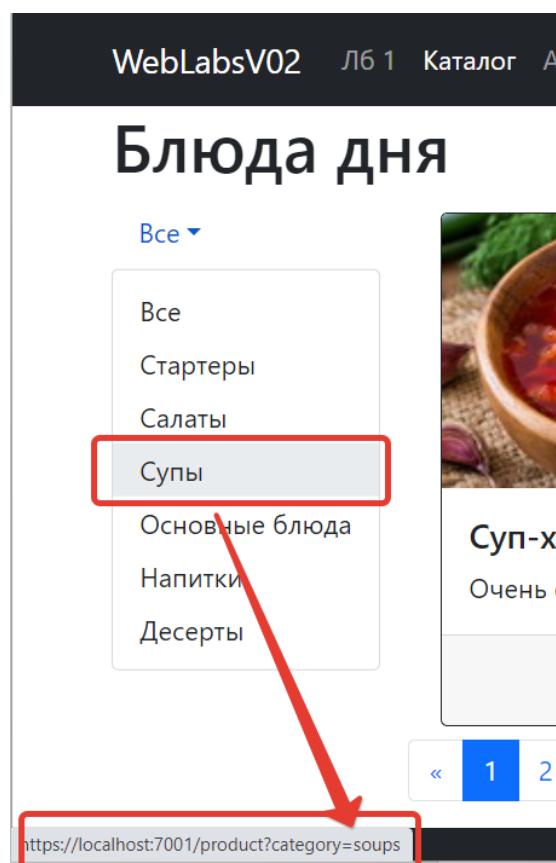
(<https://getbootstrap.com/docs/5.3/components/dropdowns/#single-button>)

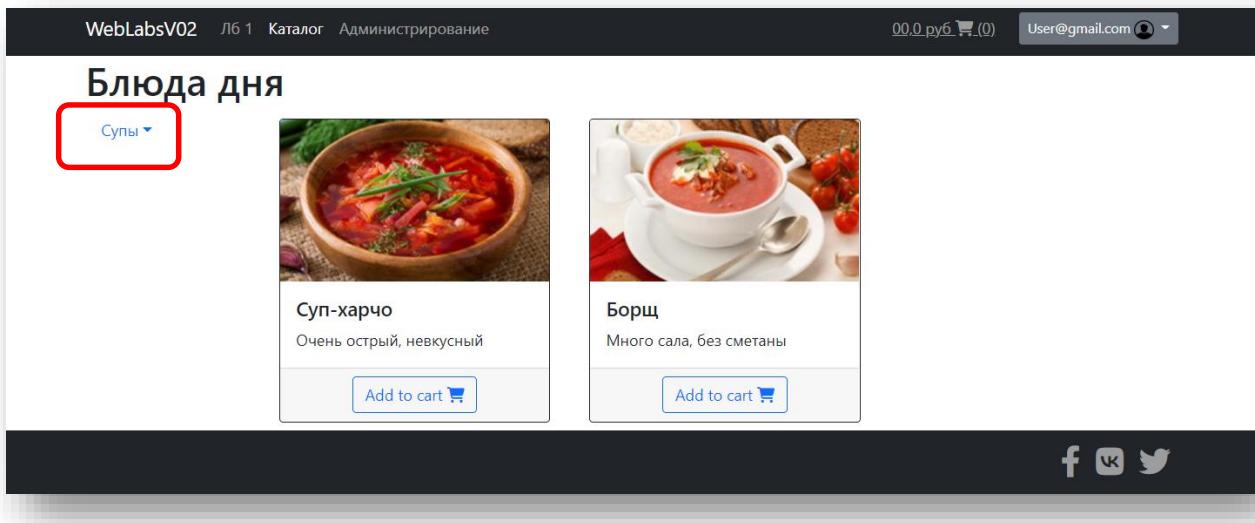
или компонента Nav-item Dropdown

(<https://getbootstrap.com/docs/5.3/components/navs-tabs/#tabs-with-dropdowns>)

При выборе категории в контроллер должен передаваться NormalizedName выбранной категории. В списке категорий должно выводится название текущей категории.

Пример:





Измените метод Index контроллера Product:

```
public async Task<IActionResult> Index(string? category)
```

Для оформления списка категорий в методе Index контроллера Product нужно:

- получить список категорий;
- получить имя текущей категории;
- передать полученные данные в представление с помощью ViewData или ViewBag.

Пример разметки для выбора категории:

```
<!-- выбор категории -->
<div class="col-2 border-2">
    <ul class="nav">
        <li class="nav-item dropdown">
            <a class="nav-link dropdown-toggle"
               data-bs-toggle="dropdown"
               href="#"
               role="button"
               aria-expanded="false">@ViewData["currentCategory"]
            </a>
            <ul class="dropdown-menu">
                <li>
                    <a class="dropdown-item"
                       asp-controller="product"
                       asp-route-category=@null>Все</a>
                </li>
                @foreach (var item in categories)
```

```

    {
        <li>
            <a class="dropdown-item"
                asp-controller="product"
                asp-route-category="@item.NormalizedName">
                @item.Name
            </a>
        </li>
    }
</ul>
</li>
</ul>
</div><!-- выбор категории -->

```

В классе MemoryProductService, в методе GetProductListAsync выполните фильтрацию объектов по категории:

```

var data = _dishes
    .Where(d => categoryNormalized == null ||
        d.Category.NormalizedName.Equals(categoryNormalized))
    .ToList();

```

Запустите проект, проверьте результат

3.8. Разбиение на страницы

Задание: представление Index контроллера Product должно выводить список из не более 3-х объектов. Нужную страницу списка передавать с помощью параметра **pageNo**. Размер страницы (количество объектов на странице) указать в файле **appsettings.json**

В представление Index в качестве модели передавайте объект **ListModel**.

В файле **appsettings.json** добавьте секцию для указания размера страницы, например:

```
"ItemsPerPage": 3
```

Измените метод Index контроллера Product:

```
public async Task<IActionResult> Index(string? category, int pageNo = 1)
```

Внедрите в конструктор класса **MemoryProductService** объект **IConfiguration**:

```
public MemoryProductService(
    [FromServices] IConfiguration config,
    ICategoryService categoryService,
    int pageNo)
```

В методе `GetProductListAsync` класса `MemoryProductService`:

- получите из конфигурации размер страницы
- вычислите общее количество страниц
- выполните выборку нужной страницы из списка объектов (используйте LINQ-запросы `Take` и `Skip`).

Используйте `ListModel` в качестве модели представления `Index`

```
return View(responseData.Data);
```

откорректируйте разметку в представлении `Index`:

```
@model ListModel<Dish>
```

Запустите проект.

Передавайте в строке запроса номер страницы. Проверьте, что разбиение на страницы работает корректно

```
++++++
```

3.9. Кнопки переключения страниц списка

На странице `Index` выполните разметку кнопок переключения страниц списка объектов. В качестве примера можно использовать компонент `Pagination` фреймворка `Bootstrap`:

<https://getbootstrap.com/docs/5.1/components/pagination/>

Если номер кнопки совпадает с текущей страницей, используйте класс «`active`» для изменения стиля кнопки.

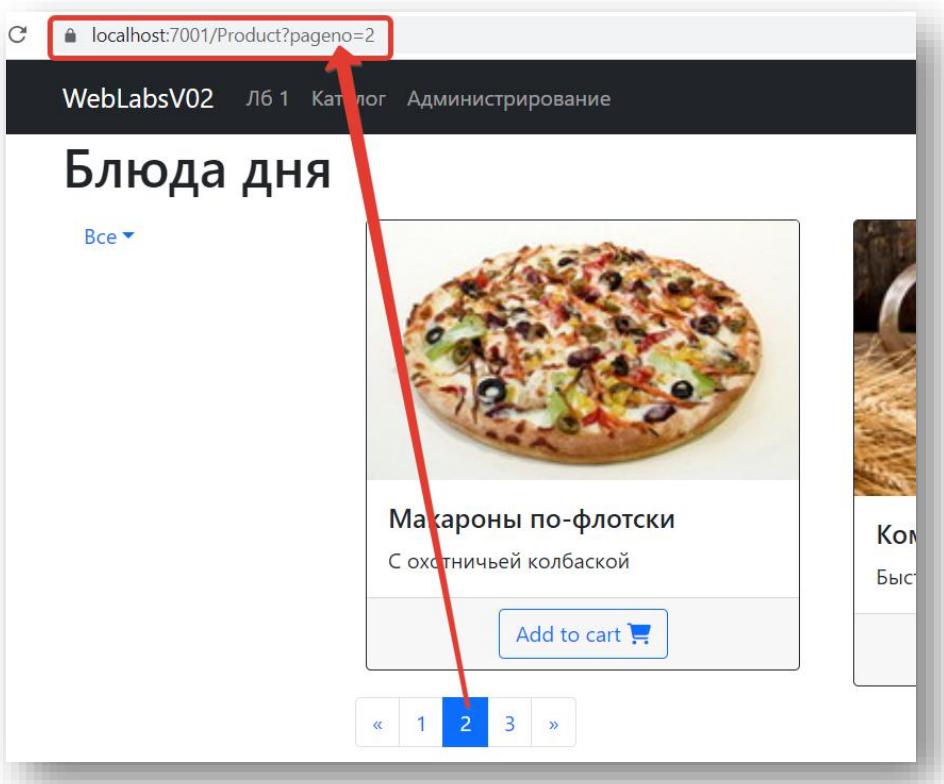
Для формирования правильного адреса необходимо в коде получить имя категории. Его можно получить из запроса:

```
string? category = request.Query["category"]?.ToString();
```

Также необходимо проверить, чтобы кнопка «Предыдущая страница» не ссылалась на номер меньше 1, а кнопка «Следующая страница» не ссылалась за пределы количества страниц:

```
int prev = Model.CurrentPage == 1
    ? 1
    : Model.CurrentPage - 1;
int next = Model.CurrentPage == Model.TotalPages
    ? Model.TotalPages
    : Model.CurrentPage + 1;
```

Запустите проект. Проверьте, что кнопки переключают страницы:



4. Контрольные вопросы

1. Как зарегистрировать сервис в ASP.Net Core?
2. Чем отличается Transient сервис от Scoped сервиса?
3. Как внедрить сервис в метод (Action) контроллера?
4. Как передать данные от клиента в метод контроллера?

5. Где механизм привязки (Model binding) ищет нужные значения?
6. Как получить данные из файла appsettings.json?
7. Как передать данные с помощью IOptions?
8. Как в коде прочитать значение, передаваемое в строке запроса?
9. Как в тэге <a> передать в запрос дополнительные данные с помощью tag-helper?
10. Что такое «Явное выражение Razor»?