Лабораторная работа №8

Маршрутизация. Передача состояния с помощью сессии и

1. Цель работы.

Знакомство с системой маршрутизации ASP.NET Core

Знакомство с механизмом сессии (сеанса) ASP.NET Core

Время выполнения работы: 6 часов

2. Общие сведения.

3. Выполнение работы

3.1. Исходные данные

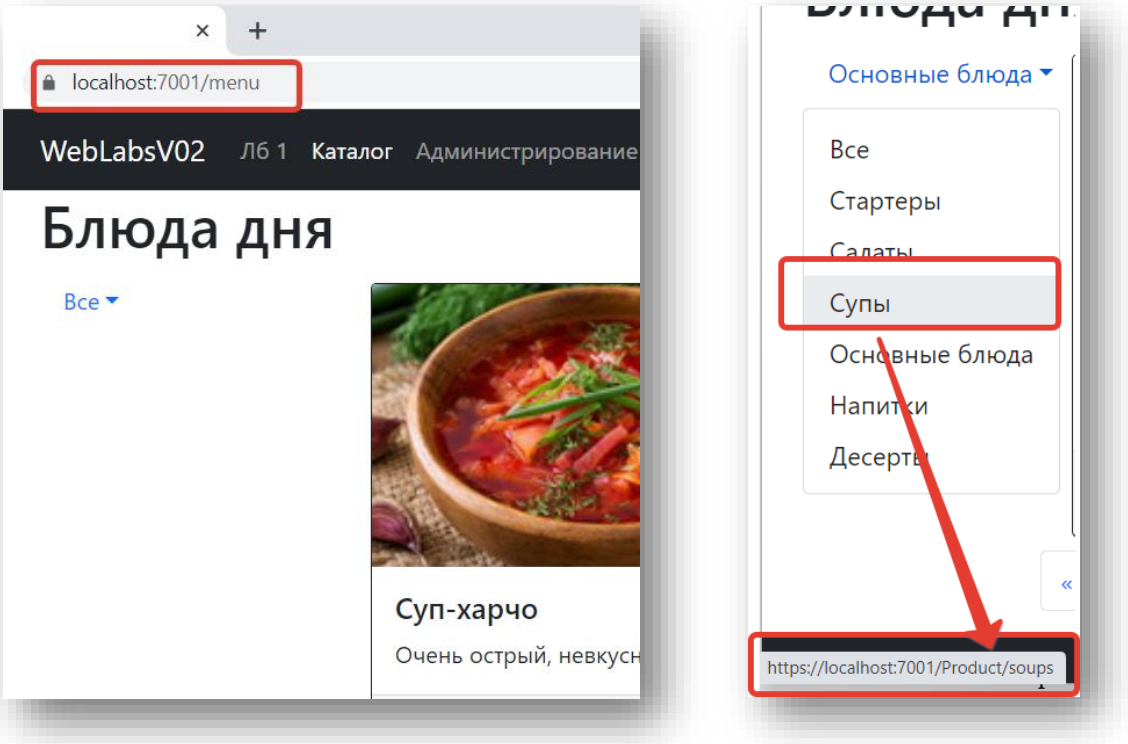
Используйте проект из лабораторной работы №7.

3.2. Задание №1. Маршрутизация

Зарегистрируйте маршрут, чтобы при обращении к методу Index контроллера Product адресная строка браузера выглядела так:

localhost:xxxx/Catalog или **localhost:xxxx/Catalog/имя_категории**.

Вместо имени «Catalog» можно использовать любое имя в контексте Вашей предметной области:



3.2.1. Рекомендации к заданию 1

Используйте маршрутизацию с помощью атрибутов.

3.3. Задание №2 Кэширование ответов API

Требуется в проекте XXX.API кэшировать ответы при запросе списка объектов.

Для кэширования использовать Hybrid Cache (для .Net версии 8.0 и выше – см. https://learn.microsoft.com/en-us/aspnet/core/performance/caching/hybrid?view=aspnetcore-9.0#cache-storage) или Distributed Memory Cache (для .Net версий ниже 8.0 – см. https://learn.microsoft.com/en-us/aspnet/core/performance/caching/distributed?view=aspnetcore-9.0)

В качестве внешнего распределенного хранилища использовать Redis.

3.3.1. Добавление Redis

Предлагается использовать контейнер docker. Для этого в файл compose.yaml добавьте сервис:

```
redis:
```

```
    image: redis:7.2-alpine
    container_name: redis
    ports:
      - "6379:6379"
    volumes:
      - redis_data:/data
    networks:
      - postgres_network
    restart: unless-stopped
```

и том:

```
volumes:
  postgres_data:
    driver: local
  redis_data:
    driver: local
```

3.3.2. Подготовка проекта XXX.API

В файле appsettings.json проекта XXX.API добавьте строку подключения:

```
"Redis": "localhost:6379"
```

Добавьте пакеты NuGet:

- Microsoft.Extensions.Caching.Hybrid
- Microsoft.Extensions.Caching.StackExchangeRedis

В классе Program зарегистрируйте сервисы:

```
builder.Services.AddHybridCache();
builder.Services.AddStackExchangeRedisCache(opt =>
{
    opt.InstanceName = "labs_";
    opt.Configuration = builder
                            .Configuration
                            .GetConnectionString("Redis");
});
```

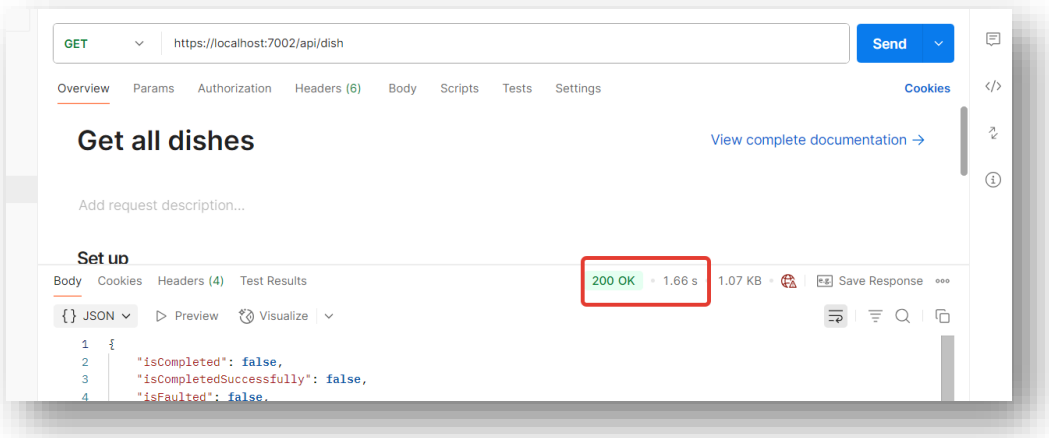
3.3.3. Кэширование данных

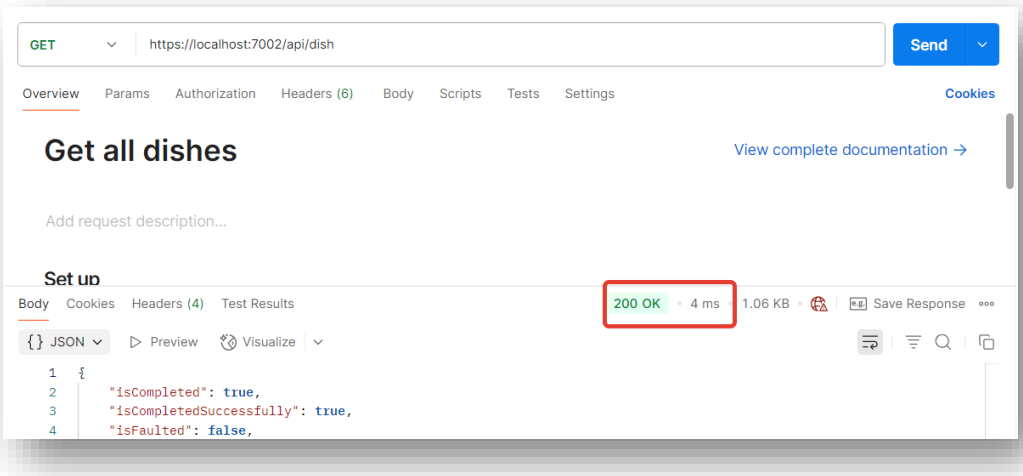
Измените конечную точку получения списка объектов, чтобы список кэшировался:

```
group.MapGet("/{category:alpha?}", async (IMediator mediator,
                                          HybridCache cache,
                                          string ? category,
                                          int pageNo=1) =>
{
    var data =
cache.GetOrCreateAsync($"dishes_{category}_{pageNo}",
        async token => await mediator.Send(
                            new GetListOfProducts(category, pageNo)),
        options: new HybridCacheEntryOptions
        {
            Expiration=TimeSpan.FromMinutes(1),
            LocalCacheExpiration = TimeSpan.FromSeconds(30)
        }
        );
    return TypedResults.Ok(data);
})
.WithName("GetAllDishes")
.WithOpenApi()
.AllowAnonymous();
```

Запустите проект.

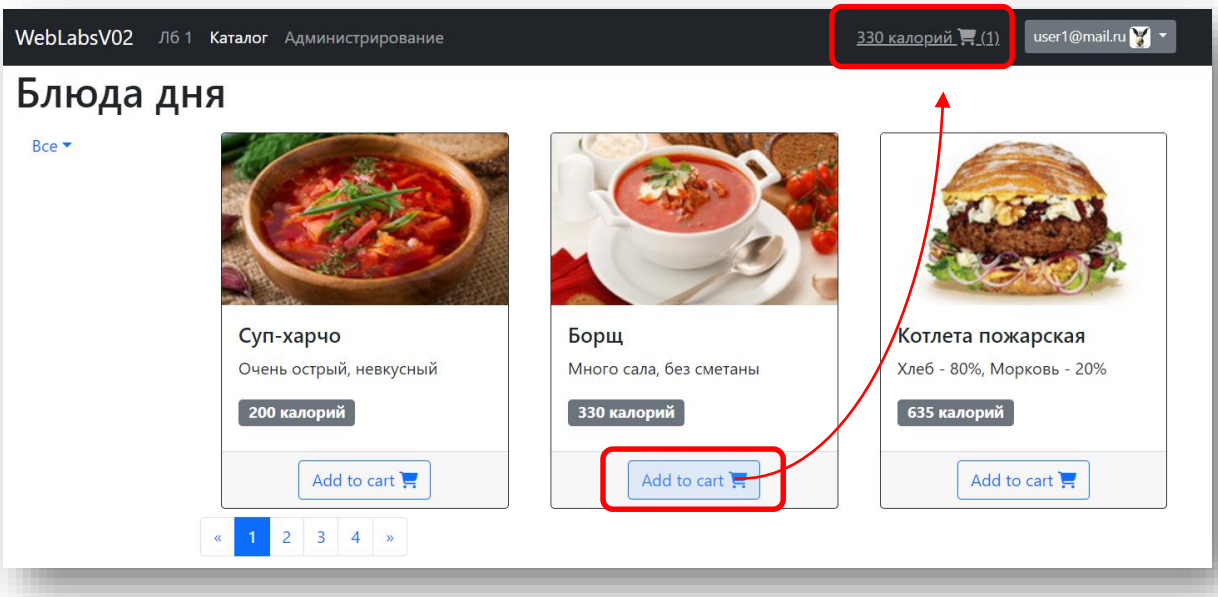
Выполняйте запросы к API. Обратите внимание на время отклика при разных интервалах между запросами (больше одной минуты или меньше). Если интервал запросов меньше 30 секунд, то результат берется из локального кэша и время отклика минимальное. При интервале от 30 сек до 1 минуты результат берется из распределенного кэша. Если интервал больше 1 минуты, то результат берется из базы данных.





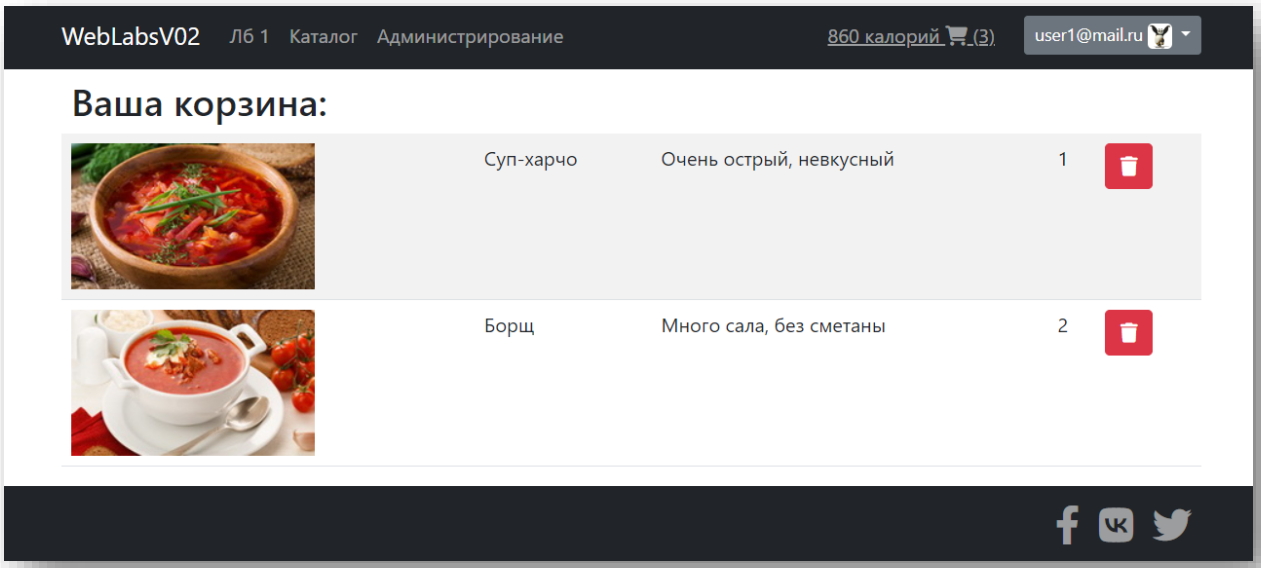
3.4. Задание №3. Корзина заказов.

В проекте XXX.UI при клике на кнопку «В корзину» выбранный объект добавляется в корзину заказов. ***При этом в меню пользователя должна правильно отображаться информация о корзине заказов***.



Добавление в корзину должно осуществляться только для пользователя, вошедшего в систему.

При клике на корзину в меню пользователя должен отобразиться список объектов в заказе с возможностью удаления объектов из заказа, например:



Корзину заказов обычно хранят в базе данных.

Но для знакомства с механизмами сохранения состояний ***будем сохранять корзину заказов в сессии.***

3.4.1. Рекомендации к заданию №3

Для работы с сессией требуется сервис кэширования (он уже есть в проекте) и сервис сессий. В классе Program добавьте сервис :

```
builder.Services.AddDistributedMemoryCache();
builder.Services.AddSession();
```

и укажите использование сессии:

```
app.UseSession();
```

Создайте контроллер Cart для работы с корзиной заказов.

В проекте XXX.Domain создайте класс – CartItem, - описывающий один элемент корзины со свойствами: объект (в примерах это Dish) и количество.

В проекте XXX.Domain опишите класс корзины заказов - Cart. Класс должен содержать список элементов корзины, предоставлять данные о количестве объектов в корзине и суммарную величину количественной характеристики объектов в корзине (в примерах – это сумма калорий), а также реализовывать добавление, удаление в корзину и очистку корзины, например:

```
public class Cart
{
    /// <summary>
    /// Список объектов в корзине
    /// key - идентификатор объекта
    /// </summary>
    public Dictionary<int,CartItem> CartItems { get; set; } = new();
    /// <summary>
    /// Добавить объект в корзину
    /// </summary>
    /// <param name="dish">Добавляемый объект</param>
    public virtual void AddToCart(Dish dish)
    {
            . . .
    }
    /// <summary>
    /// Удалить объект из корзины
    /// </summary>
    /// <param name="id"> id удаляемого объекта</param>
    public virtual void RemoveItems(int id)
    {
        . . .
    }
    /// <summary>
    /// Очистить корзину
    /// </summary>
    public virtual void ClearAll()
    {
        . . .;
    }
    /// <summary>
    /// Количество объектов в корзине
    /// </summary>
    public int Count { get => CartItems.Sum(item => item.Value.Count); }
    /// <summary>
    /// Общее количество калорий
    /// </summary>
    public double TotalCalories { get=>
CartItems.Sum(item=>item.Value.Item.Calories*item.Value.Count); }
    }
```

При добавлении объекта в корзину, если такой объект уже имеется, нужно только увеличить количество в объекте CartItem

Для сохранения объектов в сессии создайте расширяющие методы, как описано в https://learn.microsoft.com/en-us/aspnet/core/fundamentals/app-state?view=aspnetcore-7.0#set-and-get-session-values .

В контроллере Cart корзину заказов сохраняйте в сессии, например:

```
[Authorize]
[Route("[controller]/add/{id:int}")]
public async Task<ActionResult> Add(int id, string returnUrl)
{
    Cart cart = HttpContext.Session.Get<Cart>("cart") ?? new();
    var data = await _productService.GetProductByIdAsync(id);
    if (data.Success)
    {
```

```
            cart.AddToCart(data.Data);
            HttpContext.Session.Set<Cart>("cart", cart);
            }

        return Redirect(returnUrl);
}
```

Для правильного отображения информации о корзине отредактируйте CartViewComponent. В методе Invoke получите корзину из сессии и передайте ее в представление компонента.

3.5. Задание №4. Сервис корзины заказов

Требуется, чтобы объект класса Cart «внедрялся» в конструктор классов CartController и CartViewComponent. При этом CartController и CartViewComponent не должны напрямую работать с сессией. Это ослабит связь между классами и упростит рефакторинг и тестирование.

Измените код CartController и CartViewComponent для использования внедренного объекта Cart:

```
public class CartController : Controller
{
    private readonly IProductService _productService;
    private readonly Cart _cart;

    public CartController(IProductService productService,
                          Cart cart)
    {
        _productService = productService;
        _cart = cart;
    }
     . . .
    public async Task<ActionResult> Add(int id, string returnUrl)
    {
        var data = await _productService.GetProductByIdAsync(id);
        if (data.Success)
        {
            _cart.AddToCart(data.Data);
        }


        return Redirect(returnUrl);
    }
```

3.5.1. Рекомендации к заданию 3

Опишите сервисный класс (например, SessionCart), который наследуется от класса Cart и который получает корзину заказа из сессии. Переопределите в нем виртуальные методы так, чтобы при изменении данных изменения сохранялись в сессии. Зарегистрируйте класс Cart в качестве Scoped сервиса.

Пример реализации описан в книге «*Adam Freeman, Pro ASP.NET Core 6: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages.: Apress: 2022.*» (см. глава 9 «SportsStore: Completing the Cart»)

4. Контрольные вопросы

1. Какие еще механизмы, кроме сессии, есть в ASP.Net Core для сохранения состояний (сохранения данных между запросами)?
2. Где хранятся данные сессии?
3. Сколько времени хранятся данные в сессии?
4. Как прочитать/записать данные Cookie?
5. Сколько времени хранятся данные Cookie?
6. Какие преимущества дает распределенный (Distributed) кэш?
7. Что такое «Статический сегмент маршрута»? Приведите пример.
8. Как в контроллере получить значение сегмента маршрута?
9. Как в тэге <a> с помощью tag-helper указать значение сегмента маршрута?
10. Как зарегистрировать маршрут в классе Program?
11. Как зарегистрировать маршрут с помощью атрибутов?
12. Что такое ограничения (constraints) маршрута?
13. Как получить адрес конечной точки в коде контроллера или в представлении?