

## **Лабораторная работа №7**

### **Razor Pages. Сессии. Middleware (6 часов)**

#### **1. Цель работы.**

Знакомство со страницами Razor.

Изучение способов сохранения объектов в сессии.

Создание и регистрация своих сервисов.

Знакомство с механизмом логирования ASP.NET Core.

Знакомство с конвейером Middleware.

Создание своего Middleware

**Время выполнения работы: 6 часов**

#### **2. Общие сведения.**

#### **3. Выполнение работы**






##### **3.1. Исходные данные**

Используйте проект из лабораторной работы №6.

##### **3.2. Задание №1**

При клике на меню «Администрирование» должен отобразиться список всех объектов с функциями добавления, редактирования и удаления. **Раздел администрирования оформить в виде страниц Razor**

Список должен содержать изображение объекта, название и группу.

| <a href="#">+ Добавить</a>  |                     |                |                                     |
|---|---------------------|----------------|-------------------------------------|
|   | DishName            | Group          |                                     |
|  | Компот              | Напитки        | <a href="#">✎</a> <a href="#">✖</a> |
|  | Макароны по-флотски | Основные блюда | <a href="#">✎</a> <a href="#">✖</a> |
|  | Котлета пожарская   | Основные блюда | <a href="#">✎</a> <a href="#">✖</a> |
|  | Борщ                | Супы           | <a href="#">✎</a> <a href="#">✖</a> |
|  | Суп-харчо           | Супы           | <a href="#">✎</a> <a href="#">✖</a> |

Реализовать функции добавления, удаления и редактирования объектов.

При сохранении файла изображения в папку Images **в качестве имени файла использовать id объекта.**

При обращении к страницам Edit (редактирование) и Delete (удаление) id должен передаваться в качестве сегмента маршрута, а не в строке запроса, например: Admin/Edit/2

На страницах Edit и Delete при выборе группы (категории) в списке должно отображаться название, а не номер (GroupId).

При удалении объекта предусмотреть удаление файла изображения.

### 3.2.1. Рекомендации к заданию №1

Страницы администрирования поместите в область (Area) «Admin» (см. параметры запроса, указанные в компоненте MenuViewComponent)

В папку Admin/Pages скопируйте файлы \_ViewStart.cshtml и \_ViewImports.cshtml из папки Views проекта.

Для создания страниц воспользуйтесь Scaffold (автоматическим генерированием страниц) – выпадающее меню по клику правой клавиши мыши «Add->New Scaffolded Item». Отредактируйте сгенерированную страницу в соответствии с заданием.

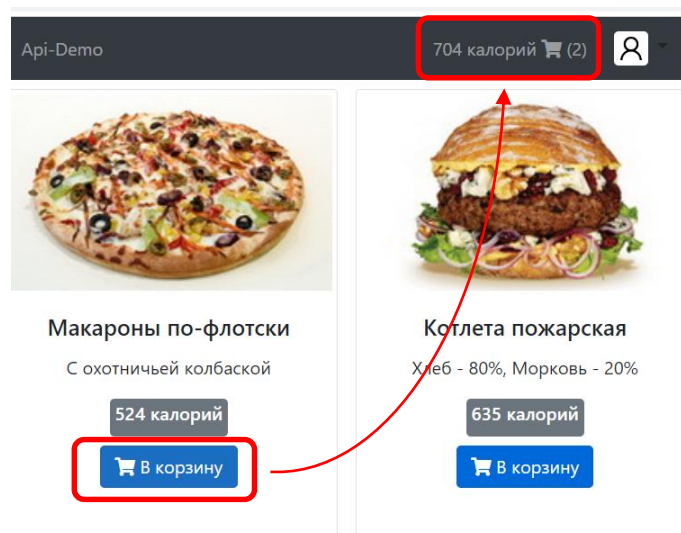
(Необязательно, для Bootstrap версии 4.3) Для оформления выбора файла изображения используйте классы Bootstrap (<https://getbootstrap.com/docs/4.3/components/forms/#file-browser>). *Это уже делалось на странице регистрации пользователя.* При использовании

класса bootstrap «custom-file» используется дополнительный скрипт «bs-custom-file-input.js». Для этого скачайте файл с сайта <https://www.npmjs.com/package/bs-custom-file-input>, включите его в проект, в папку scripts и подключите скрипт на страницах Create и Edit. Воспользуйтесь секцией Scripts страницы макета. Для активирования скрипта там же пропишите команду `bsCustomFileInput.init();`

### 3.3. Задание №2





Реализуйте функции работы с корзиной заказов.

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:

| Корзина   |                     |          |   |
|---|---------------------|----------|---|
|   |                     | Quantity |   |
|  | Компот              | 2        |  |
|  | Макароны по-флотски | 1        |  |

***Корзину заказов сохраняйте в сессии.***

### 3.3.1. Рекомендации к заданию №2

Для работы с сессией в классе Startup (.Net 5.0) или Program (.net 6.0) добавьте сервис :

```
services.AddDistributedMemoryCache();
services.AddSession(opt =>
{
    opt.Cookie.HttpOnly = true;
    opt.Cookie.IsEssential = true;
});
```

и укажите использование сессии:

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

Создайте контроллер Cart для работы с корзиной заказов.

Создайте класс – CartItem, - описывающий один элемент корзины со свойствами: объект (в примерах это Dish) и количество.

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
{
    public Dictionary<int, CartItem> Items { get; set; }
    public Cart()
    {
        Items = new Dictionary<int, CartItem>();
    }
}
```

```

    }
    /// <summary>
    /// Количество объектов в корзине
    /// </summary>
    public int Count
    {
        get
        {
            return Items.Sum(item => item.Value.Quantity);
        }
    }
    /// <summary>
    /// Количество калорий
    /// </summary>
    public int Calories
    {
        get
        {
            return Items.Sum(item => item.Value.Quantity *
item.Value.Dish.Calories);
        }
    }

    /// <summary>
    /// Добавление в корзину
    /// </summary>
    /// <param name="dish">добавляемый объект</param>
    public virtual void AddToCart(Dish dish)
    {
        // реализация метода
    }

    /// <summary>
    /// Удалить объект из корзины
    /// </summary>
    /// <param name="id">id удаляемого объекта</param>
    public virtual void RemoveFromCart(int id)
    {
        Items.Remove(id);
    }

    /// <summary>
    /// Очистить корзину
    /// </summary>
    public virtual void ClearAll()
    {
        Items.Clear();
    }
}

/// <summary>
/// Клас описывает одну позицию в корзине

```

```

/// </summary>
public class CartItem
{
    public Dish Dish { get; set; }
    public int Quantity { get; set; }
}

```

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://docs.microsoft.com/en-us/aspnet/core/fundamentals/app-state?view=aspnetcore-2.2#session-state>.

В контроллере Cart корзину заказов сохраняйте в сессии, например:

```

[Authorize]
public IActionResult Add(int id, string returnUrl)
{
    var cart = HttpContext.Session.Get<Cart>("cart") ?? new Cart();
    var dish = _context.Dishes.Find(id);
    if(dish!=null)
    {
        cart.AddToCart(dish);
        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.

### 3.4. Задание №3

Создайте сервис CartService, который получает корзину заказа из сессии и сохраняет корзину в сессии. С помощью созданного сервиса реализуйте механизм внедрения зависимости для получения объекта корзины в контроллере Cart.

### 3.4.1. Рекомендации к заданию №3

Создайте сервис (класс) `CartService`, который наследуется от класса `Cart`. Сделайте методы `AddToCart`, `RemoveFromCart` и `ClearAll` в классе `Cart` **виртуальными**. В сервисе `CartService` **переопределите** эти методы с сохранением изменений в сессии.

Объект сессии храните в сервисе.

Для получения объекта `HttpContext.Session` понадобится объект `HttpContextAccessor`. Для возможности внедрения (Dependency Injection) в классе `Startup`, в методе `ConfigureServices` добавьте:

```
services.AddSingleton<IHttpContextAccessor, HttpContextAccessor>();
```

В этом случае объект `Session` можно буде получить так:

```
provider.GetRequiredService<IHttpContextAccessor>()?
    .HttpContext
    .Session
```

где `provider` – объект `IServiceProvider`

Сделайте статический метод `GetCart(IServiceProvider provider)`, который вернет сервис в виде объекта `Cart`.

Пример реализации см. в Приложении.

Добавьте сервис (Scoped service) в методе `ConfigureServices` класса `Startup` (.Net5.0):

```
services.AddSingleton<IHttpContextAccessor, HttpContextAccessor>();
services.AddScoped<Cart>(sp => CartService.GetCart(sp));
```

или в классе `Program` (.Net6.0):

```
builder.services
    .AddSingleton<IHttpContextAccessor, HttpContextAccessor>();
builder.services.AddScoped<Cart>(sp => CartService.GetCart(sp));
```

Внедрите объект `Cart` в конструктор контроллера `CartController`.

```
public CartController(ApplicationDbContext context, Cart cart)
{
    _context = context;
    _cart = cart;
}
```

Поскольку сервис сам сохраняет в/читает из сессии, то в контроллере уже не нужно будет выполнять эти действия.

Измените CartViewComponent, чтобы получать корзину из сервиса:

```
public CartViewComponent(Cart cart)
{
    _cart = cart;
}
```

### 3.5. Задание №4

Самостоятельно изучите систему логирования в ASP.Net Core

<https://docs.microsoft.com/en-us/aspnet/core/fundamentals/logging/?view=aspnetcore-3.1>

Самостоятельно изучите использование провайдера логирования serilog в ASP.Net для записи логов в файл

<https://www.tutorialsteacher.com/core/aspnet-core-logging>

Подключите serilog в проект.

Логер **не** должен записывать сообщения категории Microsoft.

Проверьте работу логера в методе Index контроллера Product: записывайте в файл информацию о переданных значениях group и page.

**После проверки уберите логирование в контроллере Product.**

#### 3.5.1. Рекомендации к заданию №4

Загрузите в проект NuGet пакет «serilog.extensions.logging.file»

Отменить логирование можно в классе Program:

```
public static IHostBuilder CreateHostBuilder(string[] args)
{
    return Host.CreateDefaultBuilder(args)
        .ConfigureWebHostDefaults(webBuilder =>
        {
            webBuilder.UseStartup<Startup>();
        })
        .ConfigureLogging(lp =>
        {
            lp.ClearProviders();
            lp.AddFilter("Microsoft", LogLevel.None);
        });
}
```



```
        });  
    }
```

Для проверки работы логера внедрите через конструктор контроллера объект `ILogger`.

### 3.6. Задание №5

Логирование должно выполняться для **всех** запросов, на которые получен ответ с кодом состояния, отличным от 200(OK). В файл логирования должно записываться:

- Url запроса;
- код состояния ответа

#### 3.6.1. Рекомендации к заданию №5

Для логирования опишите Middleware, которое на входе будет получать текущее время, а на выходе проверять код состояния, и, если код отличен от 200, записывать заданную информацию в файл логирования.

Для получения логера в Middleware внедрите в метод `InvokeAsync` объект `ILoggerFactory`.

Для получения кода состояния используйте свойство объекта `HttpContext context.Response.StatusCode`.

Для получения Url запроса используйте свойства объекта `HttpContext context.Request.Path` и `context.Request.QueryString`.

Для первоначальной проверки работы созданного Middleware логируйте все запросы (без проверки кода состояния). Проверку кода состояния добавьте потом, когда убедитесь, что Middleware работает.

Для удобства добавления в конвейер созданного Middleware создайте расширяющий метод (например `UseLogging`) для `IApplicationBuilder`

Пример записи в log-файле:

```
2022-08-07T15:47:28.5378997+03:00 40000022-0005-ff00-b63f-84710c7967bb [INF]  
Request /Cart/Add/2?returnUrl=%2FCatalog returns status code 302 (e8389392)  
2022-08-07T15:47:28.7193212+03:00 40000019-0007-fd00-b63f-84710c7967bb [INF]  
Request /lib/font-awesome/webfonts/fa-brands-400.woff2 returns status code 404  
(c415d959)
```



## Приложение

### Пример реализации класса CartService

```
public class CartService : Cart
{
    private string sessionKey = "cart";
    /// <summary>
    /// Объект сессии
    /// Не записывается в сессию вместе с CartService
    /// </summary>
    [JsonIgnore]
    ISession Session { get; set; }
    /// <summary>
    /// Получение объекта класса CartService
    /// </summary>
    /// <param name="sp">объект IServiceProvider</param>
    /// <returns>объекта класса CartService, приведенный к типу Cart</returns>
    public static Cart GetCart(IServiceProvider sp)
    {
        // получить объект сессии
        var session = sp.GetRequiredService<IHttpContextAccessor>()
            .HttpContext
            .Session;
        // получить CartService из сессии
        // или создать новый для возможности тестирования
        var cart = session?.Get<CartService>("cart")
            ?? new CartService();

        cart.Session = session;
        return cart;
    }
    // переопределение методов класса Cart
    // для сохранения результатов в сессии
    public override void AddToCart(Dish dish)
    {
        base.AddToCart(dish);
        Session?.Set<CartService>(sessionKey, this);
    }
    public override void RemoveFromCart(int id)
    {
        base.RemoveFromCart(id);
        Session?.Set<CartService>(sessionKey, this);
    }
    public override void ClearAll()
    {
        base.ClearAll();
        Session?.Set<CartService>(sessionKey, this);
    }
}
```