Лабораторная работа №5

Razor Pages. Передача файлов

1. Цель работы.

Знакомство со сценариями построения веб-приложений, основанных на страницах. Знакомство с механизмом передачи файлов от клиента на сервер.

2. Задача работы

Научиться создавать страницы Razor. Научиться передавать файлы на сервер. Научиться передавать файлы в REST – сервис.

Время выполнения работы: 2 часа

3. Выполнение работы. Страницы администратора

3.1. Постановка задачи

В проекте XXX.UI требуется создать страницы администратора, позволяющие манипулировать данными объектов вашей предметной области в базе данных (в рамках лабораторных работ страницы для групп (категорий) объектов создавать не нужно)

Страницы должны быть выполнены по сценарию **Razor Pages**.

Страницы администратора должны располагаться в области (Areas) с названием Admin (см. п. 3.1 лабораторной работы №2).

3.2. Подготовка проекта.

Чтобы вручную не создавать страницы администратора, предлагается воспользоваться механизмом Scaffold.

Однако, данный механизм предполагает использование контекста базы данных. Для того, чтобы обойти данное ограничение предлагается в проекте XXX.UI **ВРЕМЕННО** описать контекст базы данных, с нужными сущностями.

После создания страниц администратора контекст БД можно будет удалить из проекта.

Установите в проекте XXX.UI пакеты NuGet: Microsofr.EntityFrameworkCore и пакет любого провайдера, например, Microsofr.EntityFrameworkCore.SqLite.

Опишите контекст БД:

```
public class TempDbContext : DbContext
{
    public DbSet<Dish> Dishes { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
    {
        base.OnConfiguring(optionsBuilder);
        optionsBuilder.UseSqlite("");
    }
}
```

3.3. Создание страниц администратора

Сгенерируйте страницы (**Razor pages**) администратора для выполнения операций CRUD над объектами вашей предметной области согласно п.3.1.

В коде моделей всех страниц замените использование контекста базы данных на использование **IProductService**

Для использования страницы макета и базовых пространств имен в папку, в которой находятся страницы администратора, скопируйте файлы **_ViewImports.cshtml** и **_ViewStart.cshtml** из папки Views

В разметке страницы Index вместо имени файла изображения укажите тэг для вывода изображения на страницу.

На странице Index оформите ссылки переключения между страницами с помощью иконок и стилей Bootstrap.

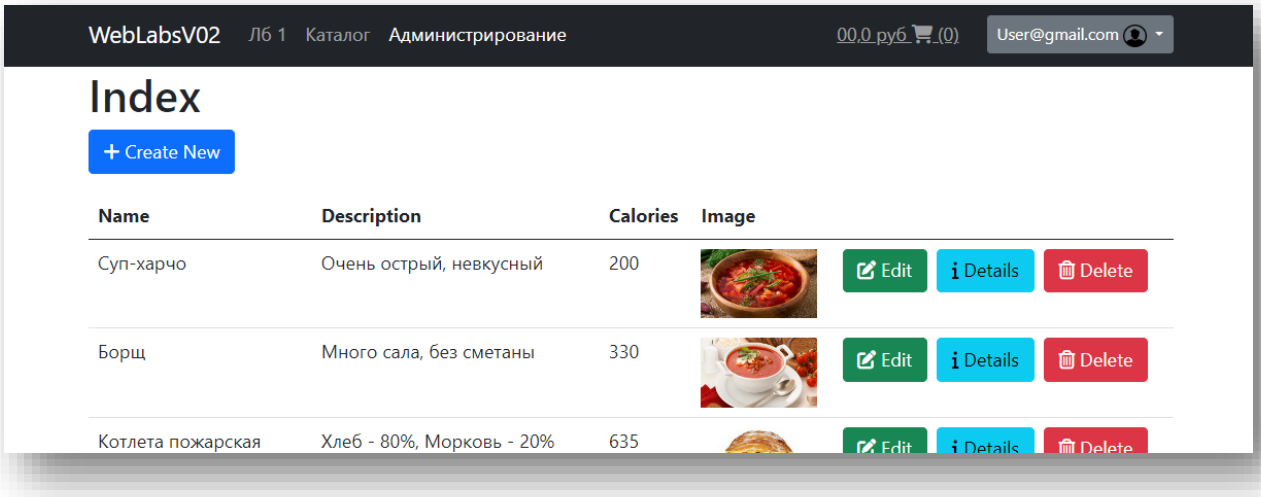
В классе **Program** зарегистрируйте использование страниц Razor:

```
builder.Services.AddRazorPages();
```

и

```
app.MapRazorPages();
```

Запустите проект. Перейдите на страницу администрирования. Убедитесь, что страницы работают корректно.



4. Выполнение работы. Передача файлов

4.1. Постановка задачи

Страницы создания и редактирования объекта должны предоставлять возможность передачи изображения объекта.

Изображения должны сохраняться в проекте XXX.API, в папке wwwroot/Images.

Для исключения дублирования имен файлов назначать файлам изображения случайные имена.

При удалении объекта или при редактировании объекта, при замене изображения, предыдущий файл изображения должен удаляться.

Соглашение об именовании: при передаче файла через http(https) предлагается использовать имя **file**

4.2. Use-case сохранения файла (проект XXX.API)

В папку Use-Cases добавьте файл SaveImage.cs, в котором опишите запрос и обработчик:

```
public sealed record SaveImage(IFormFile file) : IRequest<string>;

public class SaveImageHandler(
        IWebHostEnvironment env,
```

```
            IHttpContextAccessor httpContextAccessor)
                            : IRequestHandler<SaveImage, string>
{
    public Task<string> Handle(SaveImage request,
                            CancellationToken cancellationToken)
    {
        . . .;
    }
}
```

Внедренный объект IWebHostEnvironment используйте для получения пути к папке wwwroot (Web root path).

Внедренный объект IHttpContextAccessor используйте для получения объекта HttpContext. Из объекта HttpContext можно получить url хоста приложения Api для формирования праильного URL сохраненного изображения.

Запрос должен вернуть URL сохраненного изображения

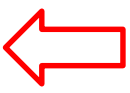
4.3. Конечная точка API для сохранения объекта с файлом изображения.

Для доступа клиента к файлам добавьте в классе program проекта XXX.API компонент Middleware для обработки запросов статических файлов (app.MapStaticAssets() или app.UseStaticFiles()).

Для возможности передачи файла данные должны передаваться в виде MultipartFormData. По умолчанию конечная точка API берет данные из тела сообщения. Но, если передавать файл, то данные нужно брать из формы.

Обязательно. Чтобы отменить использование Antiforgery токена при получении данных формы, измените регистрацию группы:

```
var group = routes.MapGroup("/api/Dish")
                  .WithTags(nameof(Dish))
                 .DisableAntiforgery();
```



Пример модификации конечной точки для получения объекта и файла изображения:

```
group.MapPost("/", async (
                          [FromForm] string dish,
                          [FromForm]IFormFile? file,
```

```
                                AppDbContext db,
                                IMediator mediator) =>
{
    var newDish = JsonSerializer.Deserialize<Dish>(dish);
    if (file!=null)
        newDish.Image = await mediator.Send(new SaveImage(file));
    db.Dishes.Add(newDish);
    await db.SaveChangesAsync();
    return TypedResults.Created($"/api/Dish/{newDish.Id}", newDish);
})
.WithName("CreateDish")
.WithOpenApi();
```

4.4. Доработка сервиса ApiProductService (проект xxx.UI)

Добавьте в методы создания и редактирования объектов возможность передачи файла в запросе к API.

Пример:

```
public async Task<ResponseData<Dish>> CreateProductAsync(
                        Dish product,
                        IFormFile? formFile)
{
    product.Image = "Images/noimage.jpg";
    var request = new HttpRequestMessage
    {
        Method = HttpMethod.Post,
        RequestUri = _httpClient.BaseAddress
    };
    var content = new MultipartFormDataContent();

    // Добавить файл изображения
    if (formFile!=null)
    {

       var streamContent = new StreamContent(formFile.OpenReadStream());
       content.Add(streamContent, "file", formFile.FileName);
    }

    // Добавить объект
    var data = new StringContent(JsonSerializer.Serialize(product));
    content.Add(data, "dish");

    request.Content = content;
    var response = await _httpClient.SendAsync(request,
                                               CancellationToken.None);
    if (response.IsSuccessStatusCode)
    {
        var responseData =
           await response.Content.ReadFromJsonAsync<ResponseData<Dish>>
                                                   (_serializerOptions);
        return responseData; // dish;
    }
```

```
    _logger.LogError($"-----> object not created. Error:
{response.StatusCode.ToString()}");
    return ResponseData<Dish>
        .Error($"Объект не добавлен. Error:
{response.StatusCode.ToString()}");
}
```

4.5. Доработка страниц Create и Update

В коде модели страниц Create и Update добавьте свойство

```
[BindProperty]
public IFormFile? Image { get; set; }
```

Передайте объект Image в сервис IProductService при создании/изменении объекта.

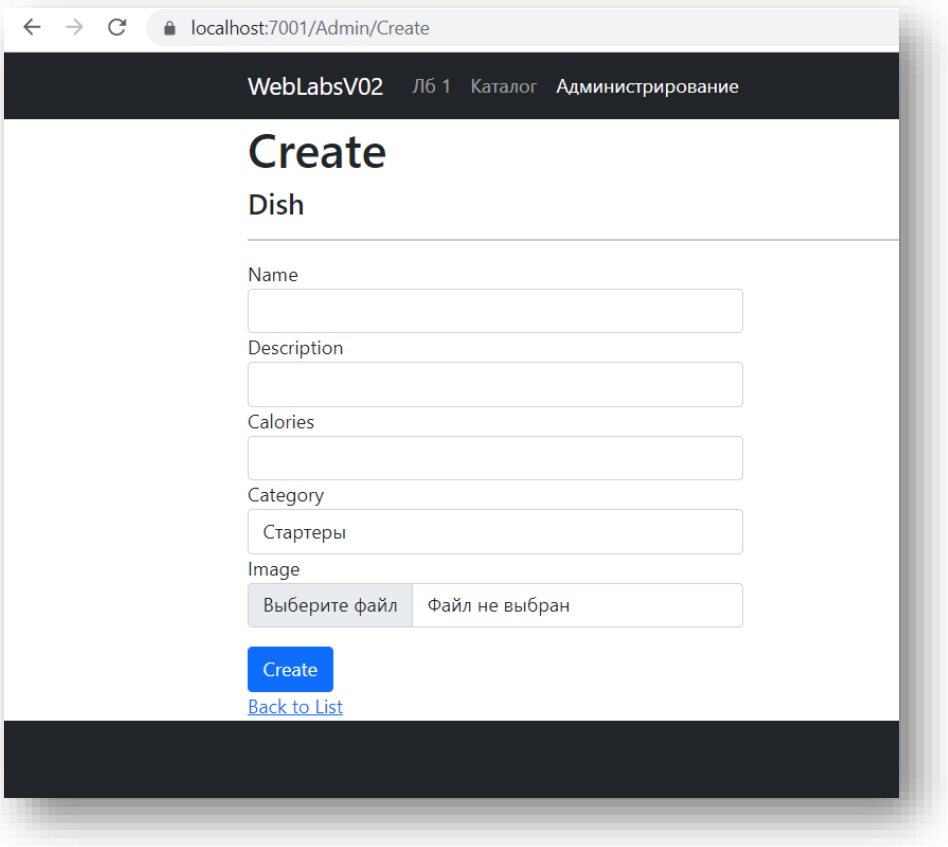
Пример:

```
await _service.CreateProduct(Dish, Image);
```

В разметке страниц Create и Update добавьте тэг для выбора и передачи файла. «Привяжите» его к свойству модели Image с помощью тэг-хелпера `asp-for`

Примечание: для передачи файлов в тэге form необходимо установить атрибут `enctype="multipart/form-data"`

Оформите тэг с помощью классов Bootstrap (см. https://getbootstrap.com/docs/5.3/forms/form-control/#file-input)



Запустите проект. Убедитесь, что изображение сохраняется на сервере.

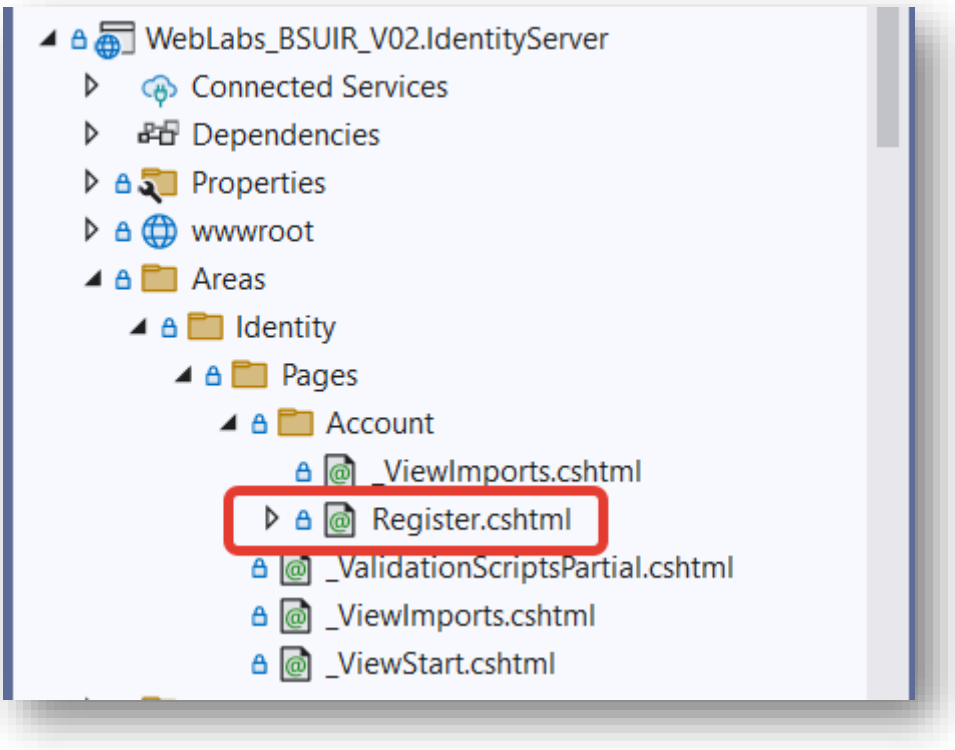
4.5.1. Оформление страницы редактирования (***необязательно***)

При редактировании объекта необходимо выводить текущее изображение объекта

5. Контрольные вопросы

1) Чем сценарий Razor Pages отличается от MVC?
2) Что такое модель страницы Razor?
3) Как обрабатываются запросы к страницам Razor?
4) Как в разметке страницы Razor получить доступ к свойствам модели страницы?
5) Как привязать данные формы к модели страницы?
6) Для чего используется директива @page ?

7) Какой Url будет у страницы Register на рисунке?



8) Чем отличается адрес страницы «/index» от «./index»?

9) Как указать дополнительный сегмент маршрута к странице (например, id)?

10) Какой Middleware используется для передачи статического контента клиенту?

11) Какой интерфейс реализует объект, полученный сервером, при передаче файла от клиента на сервер?

12) Что такое поставщики файлов (FileProvider)?

13) Как получить путь к папке «wwwroot»?

14) Форма передает файл. В каком виде этот файл будет представлен на сервере?

15) Какой атрибут нужно установить в тэге <form>, чтобы можно было передавать файлы?