Загрузка файлов

Рассмотрим, как загружать файлы на сервер в ASP.NET Core. Все загружаемые файлы в ASP.NET Core представлены типом IFormFile из пространства имен Microsoft.AspNetCore.Http. Соответственно, для получения отправленного файла в контроллере необходимо использовать IFormFile. Затем с помощью методов IFormFile мы можем произвести различные манипуляции файлом - получить его свойства, сохранить, получить его поток и т.д.

При работе с файлами прежде всего следует выбрать способ сохранения файлов. Существует два способа - в БД и в файловой системе. Здесь рассмотрим пример сохранения в БД.

Пусть у нас есть следующая модель Person:

```
public class Person
{
    public int Id { get; set; }
    public string Name { get; set; }
    public byte[] Avatar { get; set; }
}
```

Для простоты примера в модели Person всего три свойства: id, имя пользователя и массив байтов файла, который будет храниться в бд и который будет представлять аватар пользователя. В базе данных свойство Avatar будет сопоставляться со столбцом с типом varbinary(max).

Пусть у нас будет следующий контекст данных FilesContext:

```
public class FilesContext: DbContext
{
    public DbSet<Person> People { get; set; }
    public FilesContext(DbContextOptions<ApplicationContext> options)
        : base(options)
    {
        Database.EnsureCreated();
    }
}
```

Для создания модели Person определим вспомогательную модель PersonViewModel:

```
public class PersonViewModel
{
    public string Name { get; set; }
    public IFormFile Avatar { get; set; }
}
```

Не будем здесь полностью приводить код представления, самое важное – указать верный тип отправляемых данных формой и поля для загрузки файла:

```
<form asp-action="Create" asp-controller="Home" method="post"
enctype="multipart/form-data">
  …
   <input name="Avatar" type="file" class="form-control" />
  …
</form>
```

При выводе списка объектов, если файл представляет изображение, то мы можем с помощью метода Convert.ToBase64String преобразовать массив байтов в нужный формат для вывода изображения.

Далее изменим контроллер HomeController:

```
using Microsoft.AspNetCore.Mvc;
using System.Linq;
using FileUploadApp.Models;
using System.IO;

namespace FileUploadApp.Controllers
{
   public class HomeController: Controller
   {
     FilesContext _context;

     public HomeController(FilesContext context)
     {
       _context = context;
     }
     …
     [HttpPost]
     public IActionResult Create(PersonViewModel pvm)
     {
       Person person = new Person { Name = pvm.Name };
       if (pvm.Avatar != null)
       {
         byte[] imageData = null;

         // считываем переданный файл в массив байтов
          using (var binaryReader =
          new BinaryReader(pvm.Avatar.OpenReadStream()))
         {
            imageData=binaryReader.ReadBytes((int)pvm.Avatar.Length);
         }

         // установка массива байтов
         person.Avatar = imageData;
       }
       _context.People.Add(person);
       _context.SaveChanges();

       return RedirectToAction("Index");
     }
   }
}
```

В данном случае объект IFormFile с помощью класса BinaryReader считывается в массив байтов, который мы затем можем сохранить в базу данных.

Подробнее: https://metanit.com/sharp/aspnet5/21.3.php