

Современные ВЕБ-технологии

02.03.03 -Математическое обеспечение и администрирование информационных систем,
направленность (профиль) -разработка и администрирование информационных систем

<http://vikchas.ru>

Лекция 4. Тема «Модели, представления и контроллеры - MVC. Среда разработки ASP.NET Core MVC»

Часовских Виктор Петрович
доктор технических наук, профессор кафедры
ШИиКМ, ФГБОУ ВО «Уральский государственный
экономический университет

Екатеринбург 2025

MVC архитектура и [ASP.NET](#) Core MVC

1. Модели, представления и контроллеры (MVC)

MVC (Model-View-Controller)— это архитектурный паттерн проектирования, который разделяет приложение на три основных компонента:

Модель (Model)

- Представляет данные и бизнес-логику приложения
- Не зависит от пользовательского интерфейса
- Уведомляет представления об изменении состояния

Пример модели в [ASP.NET](#) Core:

C#

```
public class Product
{
    public int Id { get; set; }
    public string Name { get; set; }
    public decimal Price { get; set; }
    public string Description { get; set; }
    public int CategoryId { get; set; }
    public Category Category { get; set; }
}
```

Представление (View)

- Отвечает за отображение данных пользователю
- Получает данные от модели через контроллер
- Содержит минимум логики, в основном для отображения

Пример представления (Razor) в [ASP.NET](#) Core:

HTML

@model Product

<h2>Информация о продукте</h2>

```
<div>
    <h4>@Model.Name</h4>
    <p>Цена: @Model.Price.ToString("C")</p>
    <p>Описание: @Model.Description</p>
    <p>Категория: @Model.Category.Name</p>
</div>
```

Контроллер (Controller)

- Обработывает запросы пользователя
- Взаимодействует с моделью для получения/обновления данных
- Выбирает подходящее представление для отображения данных

Пример контроллера в [ASP.NET Core](#):

C#

```
public class ProductsController : Controller
{
    private readonly ApplicationDbContext _context;

    public ProductsController(ApplicationDbContext context)
    {
        _context = context;
    }

    public IActionResult Details(int id)
    {
        var product = _context.Products
            .Include(p => p.Category)
            .FirstOrDefault(p => p.Id == id);

        if (product == null)
        {
```

```
        return NotFound();  
    }  
  
    return View(product);  
}  
}
```

Схема взаимодействия в MVC:

1. Пользователь отправляет запрос, который обрабатывается контроллером
2. Контроллер обращается к модели для получения данных
3. Контроллер передает данные в представление
4. Представление отображает данные пользователю

2. Среда разработки ASP.NET Core MVC

Необходимые инструменты

- **.NET SDK**(последняя стабильная версия)
- **Visual Studio**или**Visual Studio Code**с расширениями для C#
- **SQL Server**(локальный или Express) или другая СУБД для работы с данными

Установка .NET SDK

1. Скачайте последнюю версию SDK с официального сайта:<https://dotnet.microsoft.com/download>
2. Запустите установщик и следуйте инструкциям
3. После установки проверьте версию в командной строке:

```
dotnet --version
```

Создание проекта ASP.NET Core MVC

Использование командной строки:

BASH

```
# Создание нового MVC проекта
```

```
dotnet new mvc -n MyMvcProject
```

Переход в директорию проекта

cd MyMvcProject

Запуск приложения

dotnet run

Использование Visual Studio:

1. Запустите Visual Studio
2. Выберите “Создать новый проект”
3. В поиске введите “[ASP.NET](#) Core Web Application”
4. Укажите имя проекта и расположение
5. Выберите шаблон “Web Application (Model-View-Controller)”
6. Нажмите “Создать”

Структура проекта [ASP.NET](#) Core MVC

MyMvcProject/

|

| — Controllers/ # Папка с контроллерами

| | — HomeController.cs # Стандартный контроллер домашней
страницы

|

| — Models/ # Папка для классов моделей

| | — ErrorViewModel.cs # Модель для отображения ошибок

|

| — Views/ # Папка для представлений (Razor)

| | — Home/ # Представления для HomeController

| | | — Index.cshtml # Главная страница

| | | — Privacy.cshtml # Страница конфиденциальности

| | — Shared/ # Общие представления и шаблоны

| | | — _Layout.cshtml # Мастер-страница

```
| | — _ViewImports.cshtml # Импорт пространств имен
| | — _ViewStart.cshtml # Начальная точка для всех представлений
|
| — wwwroot/ # Статические файлы (CSS, JS, изображения)
| | — css/
| | — js/
| | — lib/
|
| — appsettings.json # Настройки приложения
| — Program.cs # Точка входа приложения
| — Startup.cs # Настройка сервисов и middleware
```

Настройка приложения (Program.cs в .NET 6+)

C#

```
var builder = WebApplication.CreateBuilder(args);
```

```
// Добавление сервисов MVC
```

```
builder.Services.AddControllersWithViews();
```

```
// Добавление контекста базы данных
```

```
builder.Services.AddDbContext<ApplicationDbContext>(options =>
```

```
options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnec
tion")));
```

```
var app = builder.Build();
```

```
// Настройка конвейера HTTP-запросов
```

```
if (app.Environment.IsDevelopment())
```

```
{
```

```
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Home/Error");
        app.UseHsts();
    }

    app.UseHttpsRedirection();
    app.UseStaticFiles();
    app.UseRouting();
    app.UseAuthorization();

    app.MapControllerRoute(
        name: "default",
        pattern: "{controller=Home}/{action=Index}/{id?}");

    app.Run();
```

Создание простого CRUD приложения

1. Создание модели:

C#

```
public class Movie
{
    public int Id { get; set; }
    public string Title { get; set; }
    public DateTime ReleaseDate { get; set; }
    public string Genre { get; set; }
    public decimal Price { get; set; }
}
```

2. Создание контекста базы данных:

C#

```
public class ApplicationDbContext : DbContext
{
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext>
options)
        : base(options)
    {
    }

    public DbSet<Movie> Movies { get; set; }
}
```

3. Создание контроллера:

C#

```
public class MoviesController : Controller
{
    private readonly ApplicationDbContext _context;

    public MoviesController(ApplicationDbContext context)
    {
        _context = context;
    }

    // GET: Movies
    public async Task<IActionResult> Index()
    {
        return View(await _context.Movies.ToListAsync());
    }
}
```

```
// GET: Movies/Details/5
public async Task<IActionResult> Details(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var movie = await _context.Movies
        .FirstOrDefaultAsync(m => m.Id == id);
    if (movie == null)
    {
        return NotFound();
    }

    return View(movie);
}
```

```
// Методы Create, Edit, Delete
}
```

Миграции базы данных

BASH

Добавление миграции

```
dotnet ef migrations add InitialCreate
```

Обновление базы данных

```
dotnet ef database update
```

3. Преимущества [ASP.NET](#) Core MVC

- **Кроссплатформенность:** работает на Windows, macOS, Linux
- **Высокая производительность:** один из самых быстрых веб-фреймворков
- **Модульность:** сервисы могут быть добавлены по необходимости
- **Встроенная поддержка DI:** внедрение зависимостей из коробки
- **Богатая экосистема:** большое количество пакетов в NuGet
- **Безопасность:** регулярные обновления и исправления уязвимостей