

# Распределенные информационные системы

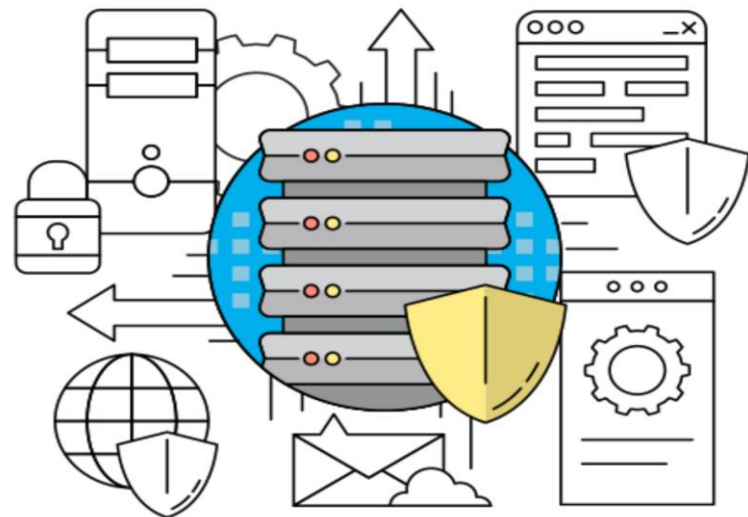
---

Лекция №9

Проектирование защищенной РИС

Фаза 5

Разработка Web-клиента



## 5. Проектирование приложений доступа к данным

---

На текущий момент мы разработали **приложение доступа к данным**, являющееся **desktop-клиентом** («толстым»).

Напомним,

что нашей целью является **разработка защищенной распределённой информационной системы «AirLogger»**, назначение и функции которой мы сформулировали на предыдущих фазах (1-2) в лекции 10, а также спроектировали архитектуру системы и модель данных (фазы 3-4) в лекции 11.

Выяснили, что

- **Архитектура системы AirLogger многоуровневая** (уровень сервера БД, уровень сервера приложений и уровень клиента).
- **В качестве базовой архитектуры используем ASP.Net CORE Architecture**
- Сервер баз данных **MS SQL Server** и **архитектура данных** представлена **ORM-моделью**, содержащей 12 взаимосвязанных сущностей
- **Реплицируемой** сущностью является **UseException**

## 5. Проектирование приложений доступа к данным

---

**Доступ к данным в системе AirLogger осуществляют клиенты двух типов:**

- **«толстый» desktop-клиент** - «Комплексный проект студента», являющийся источником данных,
- **«тонкий» ASP.Net Web-клиент** - «Логгер-консоль», предназначенный для мониторинга исключений и регистрации инцидентов

На стороне desktop-клиента формируются фрагменты реплицируемой базы данных, для чего desktop-клиент использует **MS SQL Server Express**

**Базовые платформы: Windows 10** (допустима Windows 7), **IDE – MS Visual Studio 2019 .NET Core 3**

В этой лекции акцентируем свое внимание на разработке «тонкого» **ASP.Net Web-клиента**.

**Также освоим подход к проектированию, называемый “Model-First”**

## 5. Проектирование приложений доступа к данным

Перед проектированием web-клиента, еще раз перечислим, какие заявлены функции в разрабатываемой системе.

Система **AirLogger** поддерживает следующие функции:

Это делает desktop-клиент



**1.Консолидирует** в рамках одного программного проекта функционал циклов лабораторных работ студента по дисциплинам в области информационной безопасности

**2.Формирует и хранит** лог-журнал исключений, выявленных в ходе выполнения лабораторных работ

**3.Генерирует** фрагменты **распределенной базы данных** исключений

**4.Реплицирует** распределенные данные

**5.Формирует инциденты** по фактам проверки уязвимостей проектов студентов

**6.Формирует аналитический отчет** по выявленным инцидентам

**А зачем нужен web-клиент?**

Это должны делать сервера (сервер приложений и сервер БД)

## 5. Проектирование приложений доступа к данным

---

Назначение практически любого «тонкого» клиента заключается либо в **удаленном мониторинге данных** информационной системы, либо в **удаленном управлении модификацией и/или обработкой данных**.

Причем под управлением здесь понимается отправление некоторых web-запросов на управление, а собственно управление реализуется на стороне сервера.

В системе **AirLogger** может быть множество различных **Web-клиентов**.

**В первую очередь нужно реализовать** мониторинг исключений, которые **реплицированы в базу данных** и поэтому находятся уже в структурированном виде на сервере БД.

Следуя концепции визуального проектирования, первоначально **проектируется интерфейс приложения** и согласуется с заказчиком.

## 5. Проектирование приложений доступа к данным

Пусть согласованный интерфейс стартовой страницы web-клиента выглядит следующим образом:

The screenshot shows a web browser at localhost:5001 displaying the 'Index' page of a 'Logger of exceptions' application. The page has a sidebar with navigation icons and a top navigation bar with links to 'Home' and 'Privacy'. The main content area features a table of exceptions with columns: Message, TargetSite, DateTimeExc, and IndexForm. Each row in the table has a set of management links (Edit, Details, Delete) to its right. Red boxes highlight the table and the management links, with arrows pointing to them from labels at the bottom. The footer contains copyright information and a privacy link.

Message	TargetSite	DateTimeExc	IndexForm	
excep1	method 1	01.01.0001 12:00:00	5	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
excep2	method 2	01.01.0001 12:00:00	6	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
excep3	method 3	01.01.0001 12:00:00	7	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
except4	method 4	01.01.0001 12:00:00	3	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>

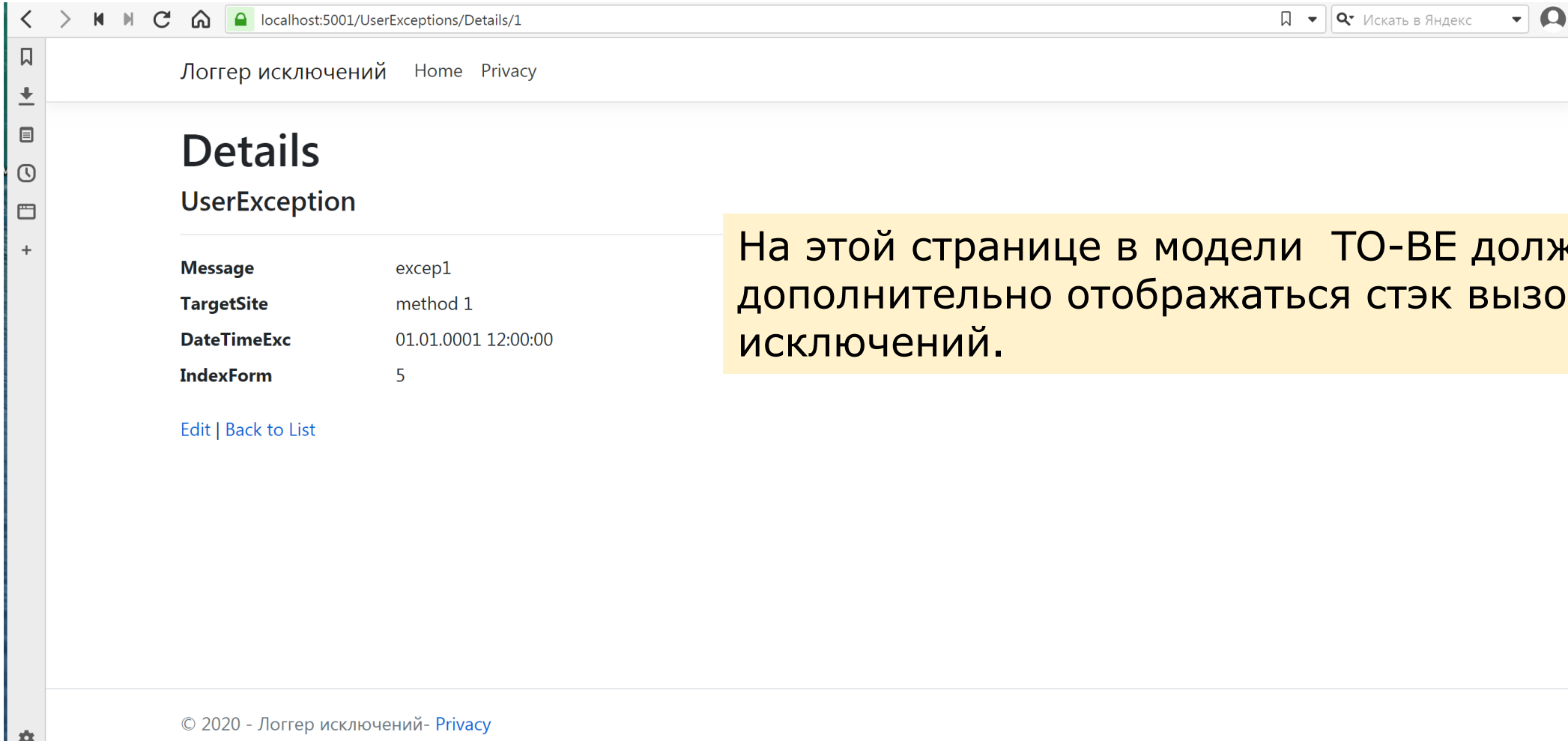
Данные, загружаемые из реплицированной БД

Элементы управления

© 2020 - Логгер исключений- [Privacy](#)

## 5. Проектирование приложений доступа к данным

При нажатии на элемент управления «**Details**», должен осуществляться переход на страницу детализации исключения



The screenshot shows a web browser window with the address bar displaying 'localhost:5001/UserExceptions/Details/1'. The page has a header with 'Логгер исключений' and links for 'Home' and 'Privacy'. The main content area is titled 'Details' and 'UserException'. It contains a table with the following data:

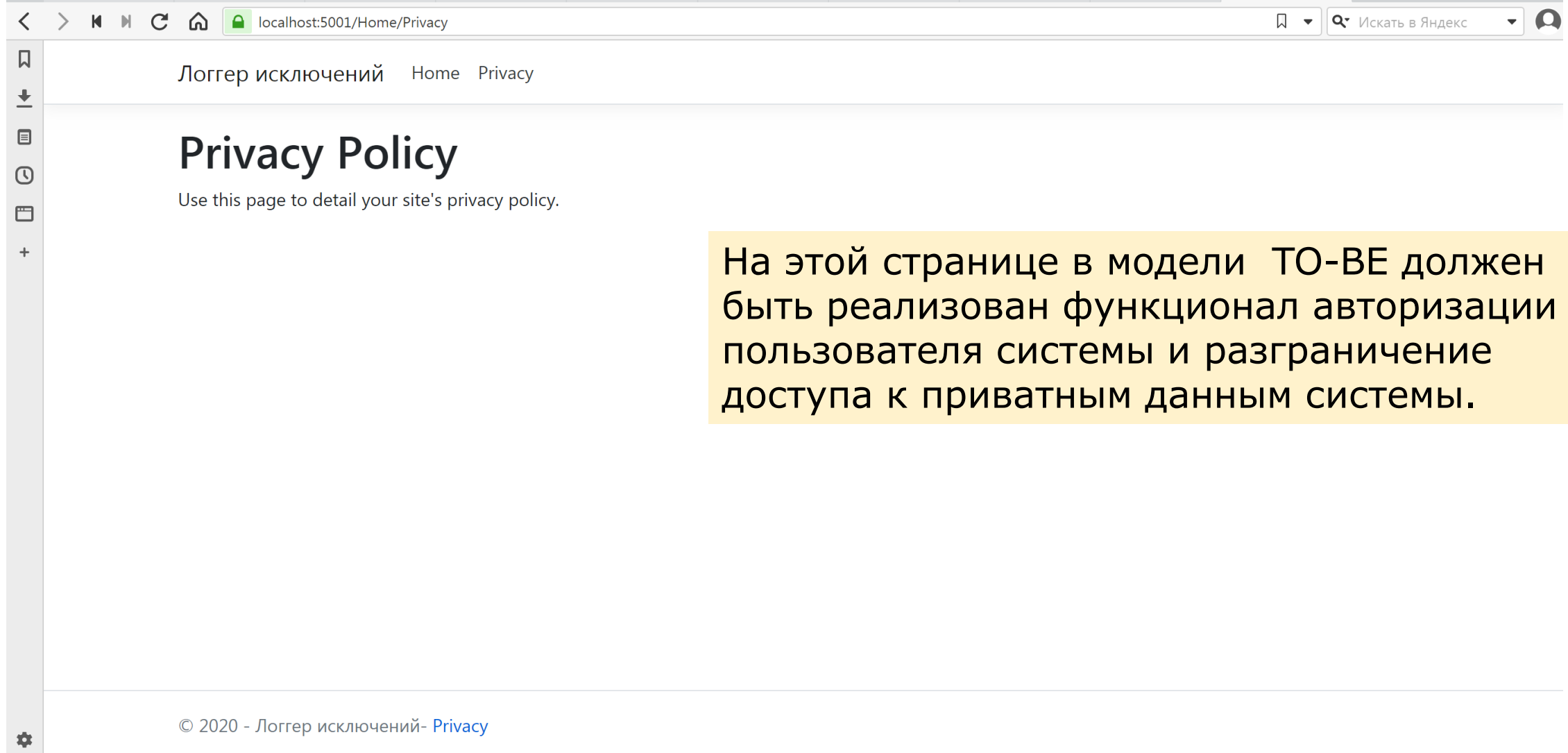
Message	excep1
TargetSite	method 1
DateTimeExc	01.01.0001 12:00:00
IndexForm	5

Below the table, there are links for 'Edit' and 'Back to List'. A yellow callout box on the right side of the page contains the text: 'На этой странице в модели ТО-ВЕ должен дополнительно отображаться стэк вызовов исключений.'

At the bottom of the page, there is a footer with the text: '© 2020 - Логгер исключений- Privacy'.

## 5. Проектирование приложений доступа к данным

При нажатии на гиперссылку «**Privacy**», должен осуществляться переход на страницу детализации исключения





## 5. Проектирование приложений доступа к данным

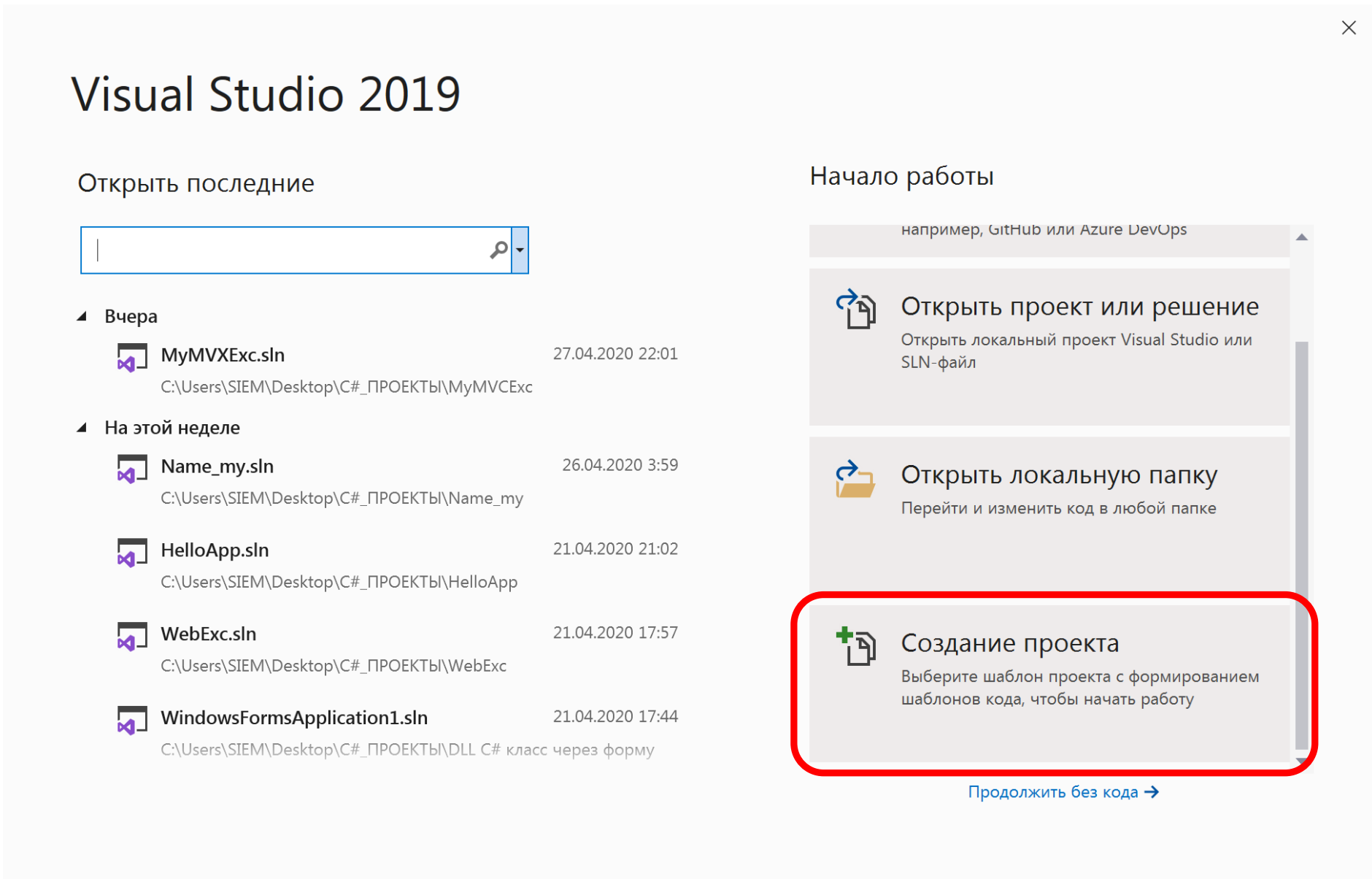
---

Начнем работу с Entity Framework с создания **Web-клиента**, осуществляющего доступ только к **одной сущности разработанной модели данных** (в дальнейшем модернизируем этот клиент на модель, соответствующую разработанной). Будем считать такой проект тестовым.

Будем использовать **подход Model-First**, при котором сначала создается **графическая модель данных EDMX**, а затем создаются объекты в базе данных.

Подход **Model-First** имеет преимущество - он **позволяет абстрагироваться от работы с базой данных и кодом C#**, и заниматься непосредственно вопросами **бизнес-моделирования**.

# Использование мастера EDM для создания модели




# Создание проекта


## Последние шаблоны проектов

 Веб-приложение ASP.NET (.NET Framework) C#


 Консольное приложение (.NET Core) C#


 Windows Forms App (.NET Core) C#

 Веб-приложение ASP.NET Core C#

 Статическая библиотека C++

 Пустой проект C++

 Консольное приложение C++

 Приложение Windows Forms (.NET Framework) C#

 WPF App (.NET Core) C#

Поиск шаблонов (ALT+"B")



[Очистить все](#)

C#

Все платформы

Все типы проектов



Консольное приложение (.NET Core)

Проект для создания приложения командной строки, которое может выполняться в среде .NET Core в Windows, Linux и Mac OS.

C#

Linux

macOS

Windows

Консоль



Веб-приложение ASP.NET Core

Шаблоны проектов для создания веб-приложений ASP.NET Core и веб-API в Windows, Linux и macOS с использованием .NET Core или .NET Framework. Создавайте веб-приложения с использованием Razor Pages и MVC, а также одностраничные приложения с использованием Angular, React и React + Redux.

C#

Linux

macOS

Windows

Облако

Служба

Веб



Приложение Blazor

Шаблоны проектов для создания приложений Blazor, которые выполняются на сервере в приложении ASP.NET Core или в браузере в WebAssembly. Эти шаблоны можно использовать для создания веб-приложений с полнофункциональными динамическими пользовательскими интерфейсами (UI).

C#

Linux

macOS

Windows

Облако

Веб

[Назад](#)

[Далее](#)



# Настроить новый проект

Веб-приложение ASP.NET Core

C#

Linux

macOS

Windows

Облако

Служба

Веб

Имя проекта

WebASPTest

Расположение

C:\Users\SIEM\Desktop\C#\_ПРОЕКТЫ\

...

Дадим проекту имя  
**WebASPTest**



Поместить решение и проект в одном каталоге

Назад

Создать

# Создайте веб-приложение ASP.NET Core

.NET Core

ASP.NET Core 3.1



## Пустой

Пустой шаблон проекта для создания приложения ASP.NET Core. Этот шаблон не имеет содержимого.



## API

Шаблон проекта для создания приложения ASP.NET Core с образцом контроллера для службы HTTP RESTful. Этот шаблон можно также использовать для представлений MVC и контроллеров ASP.NET Core.



## Веб-приложение

Шаблон проекта для создания приложения ASP.NET Core с образцом содержимого ASP.NET Core Razor Pages.



## Веб-приложение (модель-представление-контроллер)

Шаблон проекта для создания приложения ASP.NET Core с образцом представлений MVC и контроллеров ASP.NET Core. Этот шаблон можно также использовать для служб HTTP RESTful.



## Angular

Шаблон проекта для создания приложения ASP.NET Core с Angular.



## React.js

[Получить дополнительные шаблоны проекта](#)

## Аутентификация

без проверки подлинности

[Изменение](#)

## Дополнительно

☒ Настроить для HTTPS

☐ Включить поддержку Docker

(требуется [Docker Desktop](#))

Linux

Автор: Microsoft

Источник: .NET Core 3.1.1

Назад

Создать

Файл Правка Вид Проект Сборка Отладка Тест Анализ Средства Расширения Окно Справка Поиск (Ctrl+Q)

WebASPLooger

Обзор

Подключенные службы

Ссылки на службу

Опубликовать

ASP.NET Core

Учитесь работать с платформой .NET, создав свое первое приложение

Разработка приложения

Документы, примеры и учебники

Архитектура приложений .NET

Обозреватель решений

Обозреватель решений — поиск (Ctrl+;)

Решение "WebASPLooger" (проекты: 1 из 1)

WebASPLooger

- Connected Services
- Properties
- wwwroot
- Зависимости
- Controllers
- Models
- Views
- appsettings.json
- Program.cs
- Startup.cs

Нажмите эту ссылку, чтобы прочитать руководство по MVC

Обозреватель решений

Обозреватель решений — поиск (Ctrl+;)

Решение "WebASPLooger" (проекты: 1 из 1)

WebASPLooger

- Connected Services
- Properties
- wwwroot
- Зависимости
- Controllers
- Models
- Views
- appsettings.json
- Program.cs
- Startup.cs

Свойства

Список ошибок Вывод

Нажмите **CTRL+F5**, чтобы запустить приложение без отладки. Visual Studio отображает следующее диалоговое окно.

# Документация по ASP.NET

Узнайте, как с помощью ASP.NET Core создавать быстрые и безопасные кроссплатформенные или облачные веб-приложения и службы. Изучайте руководства, примеры кода, основные понятия, справочник по API и многое другое.



НАЧАЛО РАБОТЫ

**Создание приложения ASP.NET Core на любой платформе за 5 минут**



ОБЩИЕ СВЕДЕНИЯ

**Обзор ASP.NET Core**



VIDEO

**Учебные видео по ASP.NET Core**



НОВОЕ

**Новые возможности в документации по ASP.NET Core**



НАЧАЛО РАБОТЫ

**Создание первого пользовательского веб-интерфейса**



НАЧАЛО РАБОТЫ

**Создание первого веб-API**



НАЧАЛО РАБОТЫ

**Создание первого веб-приложения в реальном времени**



НАЧАЛО РАБОТЫ

**Создание первого веб-приложения MVC на основе данных**

## Версия

ASP.NET Core 3.1

 Фильтровать по названию


Основные принципы работы с **контроллерами, представлениями и моделями (MVC)** можно прочитать здесь

Действия и представления контроллера

Добавление поиска

Добавление нового поля

Добавление проверки

 Скачать PDF

# Начало работы с MVC ASP.NET Core

16.10.2019 • Время чтения: 10 мин •  Автор: [Рик Андерсон](#) (Rick Anderson)

В этом руководстве описывается веб-разработка MVC ASP.NET Core с контроллерами и представлениями. Если вы не знакомы с веб-разработкой ASP.NET Core, для начала изучите первую главу этого руководства для [Razor Pages](#).

В этом учебнике приводятся основные сведения о веб-приложении MVC ASP.NET Core.

Это приложение служит для управления базой данных названий фильмов. Вы научитесь:

- ✓ Создание веб-приложения.
- ✓ Добавление модели и формирование шаблона.
- ✓ Работа с базой данных.
- ✓ Добавление поиска и проверки.

В конечном итоге вы получите приложение, позволяющее управлять данными фильмов и отображать их.

Были ли сведения на этой странице полезными?

 Да  Нет

## В этой статье

[Предварительные требования](#)[Создание веб-приложения](#)[Справка по Visual Studio](#)



## Шаг 1. Добавление модели в приложение ASP.NET Core

---

Добавим классы модели для управления исключениями.

**Классы модели** приложения используются в Entity Framework Core(EF Core) для **работы с базой данных**.

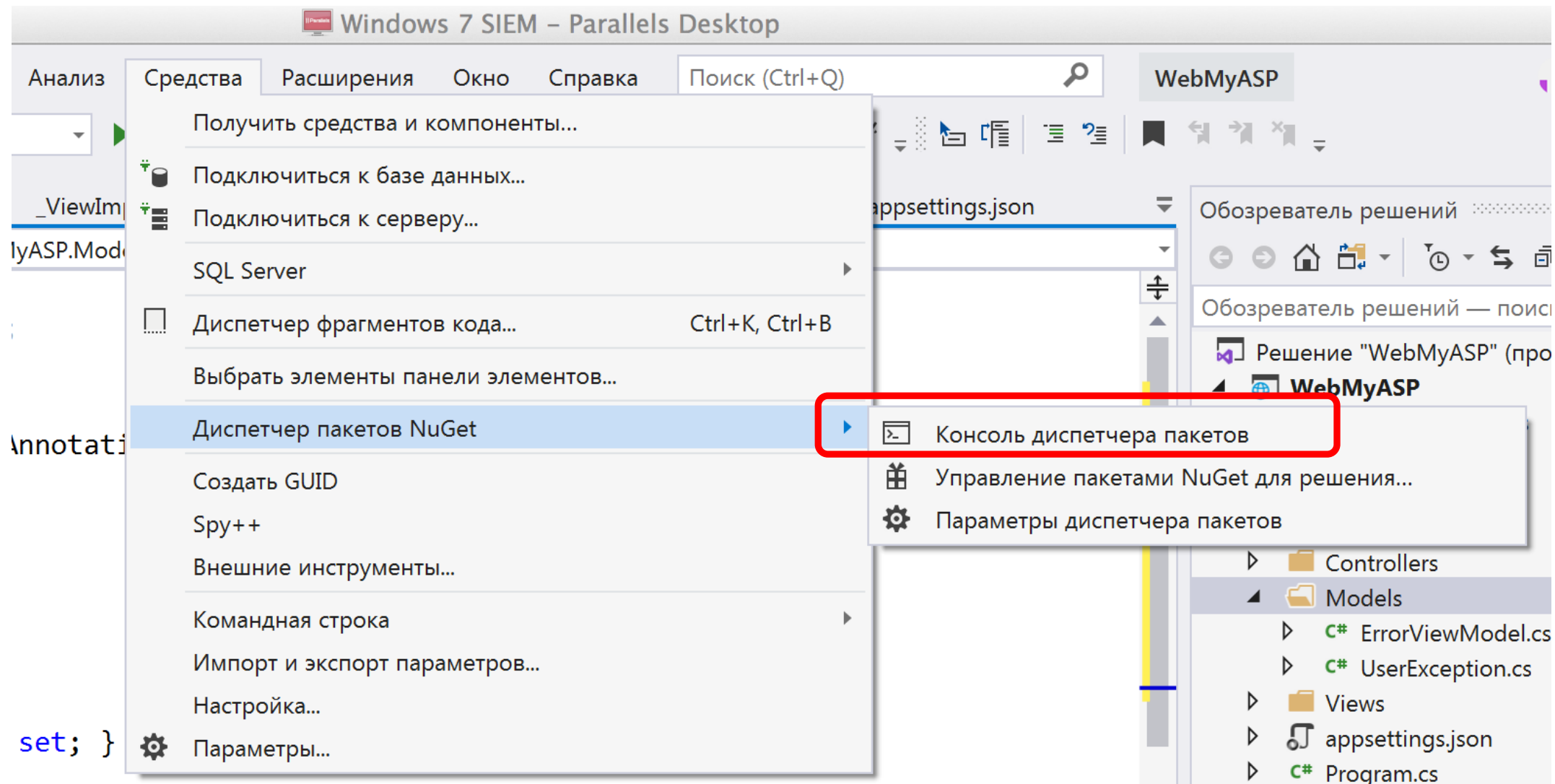
EF Core —это объектно-реляционный сопоставитель (ORM), упрощающий получение доступа к данным.

Эти классы моделей называются **классами POCO** (от plain old CLR objects — "старые добрые объекты CLR"), так как они не зависят от EF Core.

Они определяют **свойства данных**, которые хранятся в базе данных.

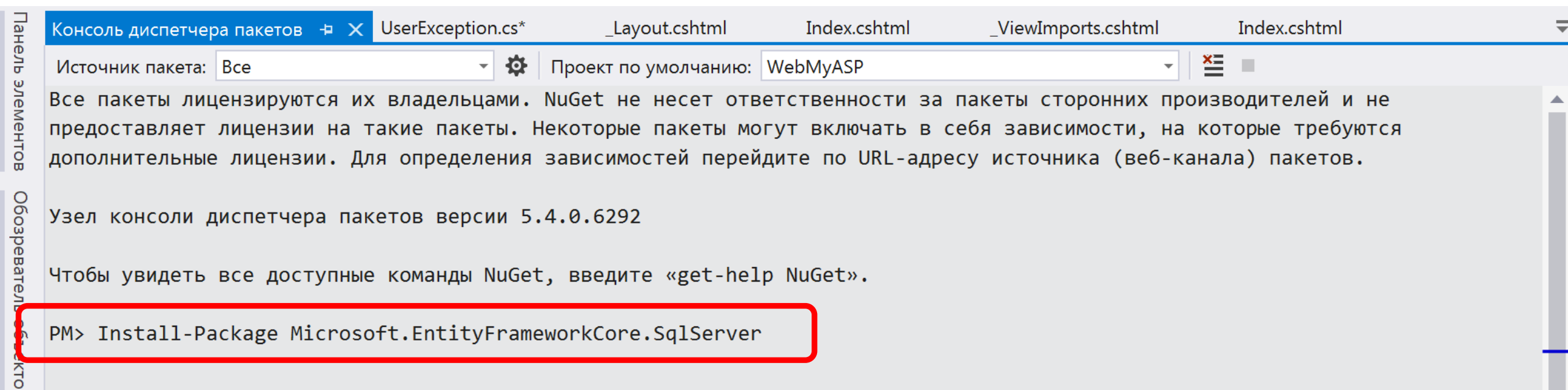
Для их использования нужно загрузить с помощью поставщика пакетов **NuGet** пакет **Microsoft.EntityFrameworkCore.SqlServer**

В меню **Средства** последовательно выберите пункты **Диспетчер пакетов NuGet** > **Консоль диспетчера пакетов** (PMC).



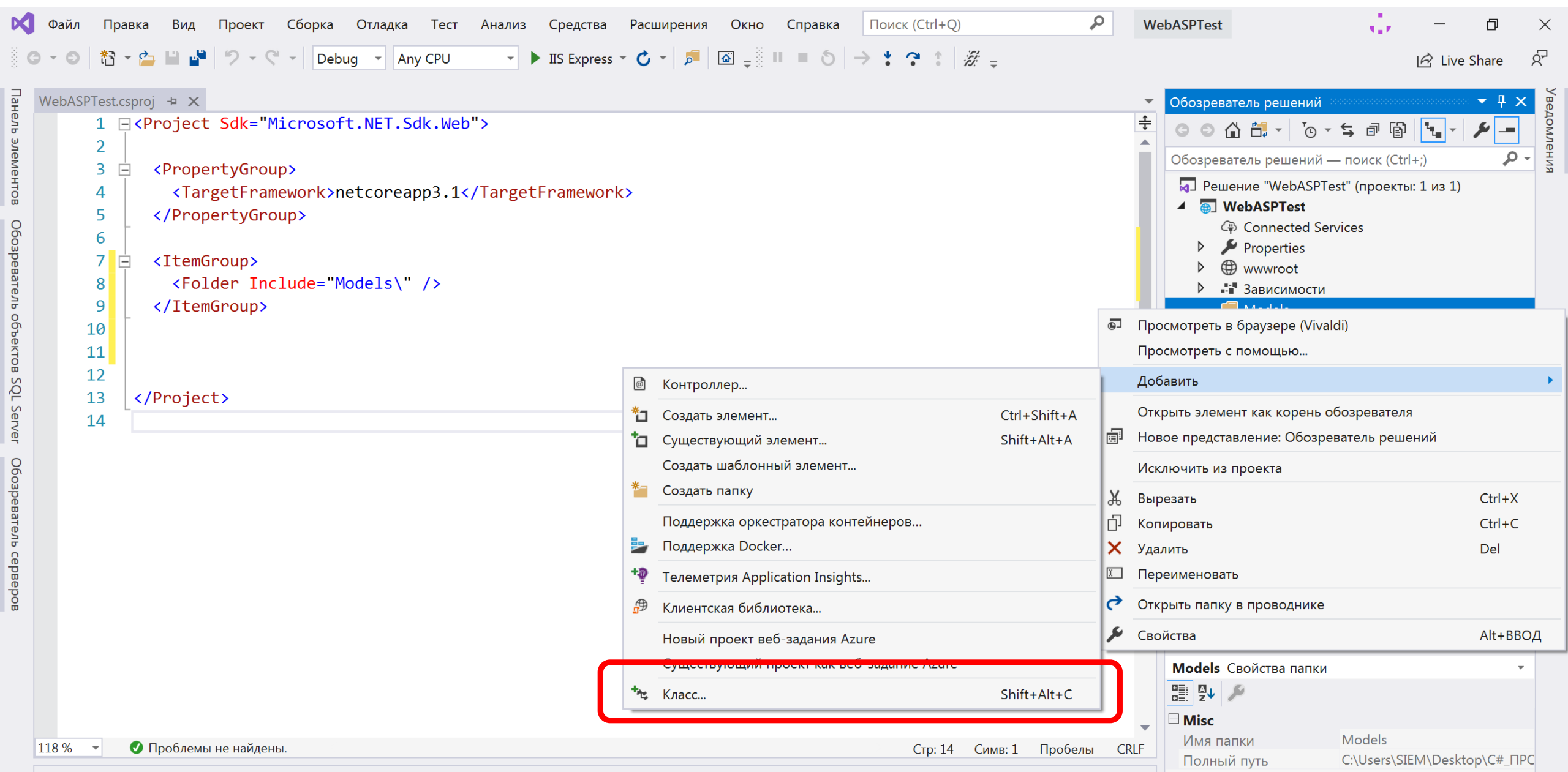
В РМС выполните следующую команду:

`Install-Package Microsoft.EntityFrameworkCore.SqlServer`



После загрузки пакета имеем возможность работать с классами РОСО

Щелкните правой кнопкой мыши папку **Models**. Выберите **Добавить > Класс**. Присвойте классу имя **UserException**.



## Установленные

## Visual C#

## ASP.NET Core

Данные

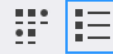
Код

Общие

Веб

В сети

Сортировка: По умолчанию



Поиск (Ctrl+E)



Класс

Visual C#



Интерфейс

Visual C#



Файл с текстом программы

Visual C#

**Тип:** Visual C#

Пустое объявление класса

Имя:

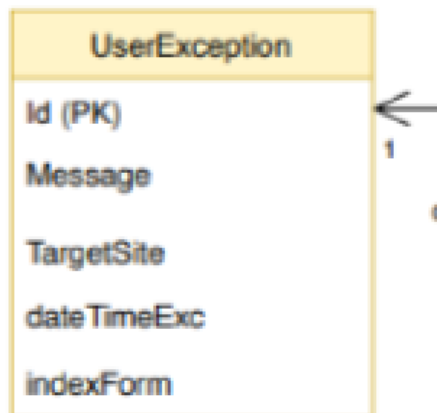
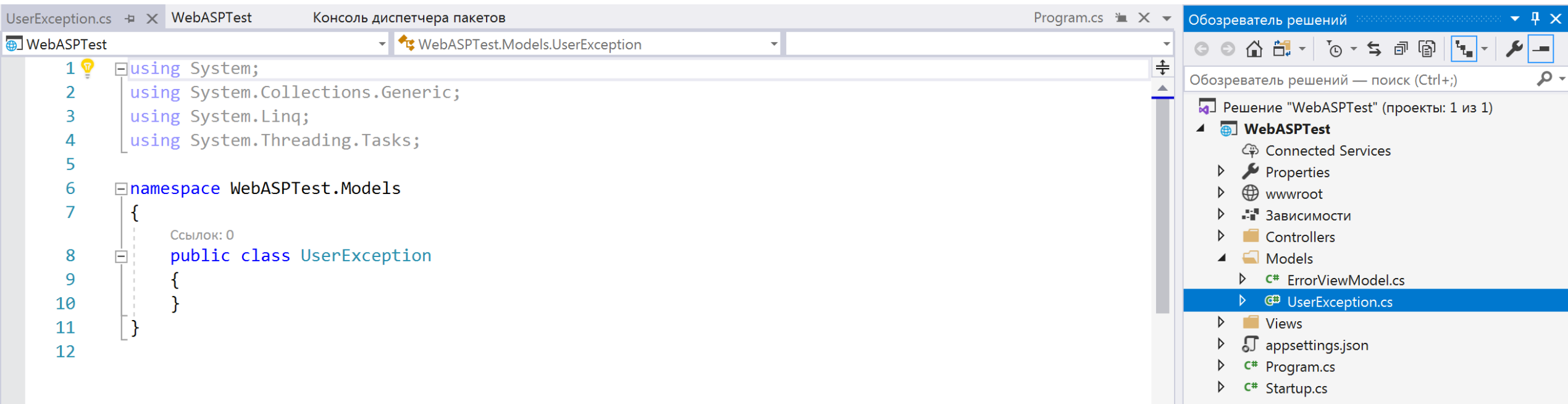
UserExceptions

Добавить

Отмена

# Добавление модели в приложение ASP.NET Core

Мы добавили шаблон **класса POCO** с именем **UserException**



Свойства этого **класса UserException** должны соответствовать свойствам нашей **сущности UserException**

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Threading.Tasks;
5 using System.ComponentModel.DataAnnotations;
```

```
7 namespace WebASPTest.Models
```

```
8 {
9     public class UserException
```

```
10 {
11     public int Id { get; set; }
12     public string Message { get; set; }
13     public string TargetSite { get; set; }
```

```
15     [DisplayFormat(DataFormatString = "{0:dd.MM.yyyy hh:mm:ss}",
16     ApplyFormatInEditMode = true)]
```

```
17     public DateTime DateTimeExc { get; set; }
```

```
18     public int IndexForm { get; set; }
```

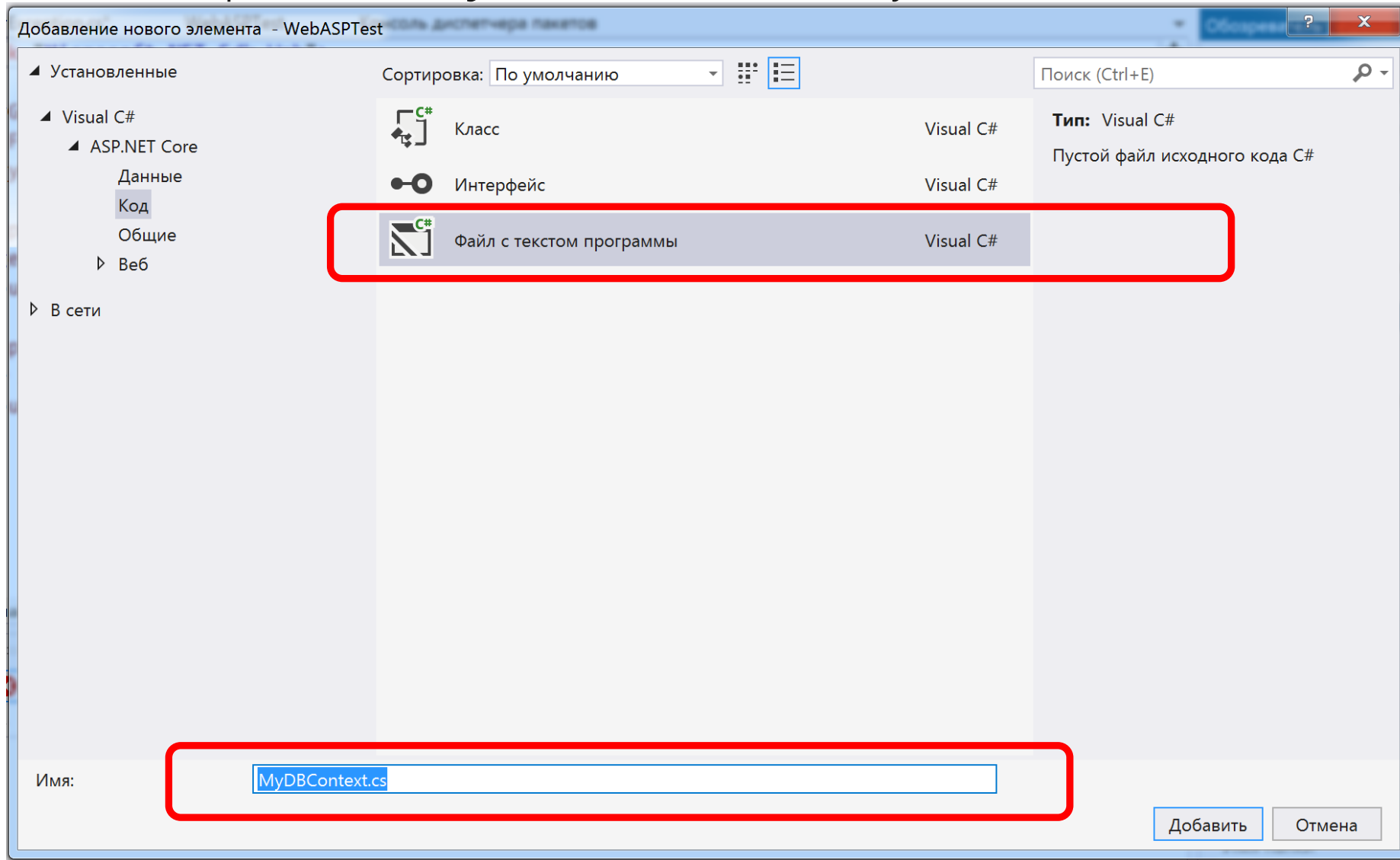
```
19 }
```

```
20 }
```

Добавьте пространство имен **System.ComponentModel.DataAnnotations**, а также в класс **UserException** следующие СВОЙСТВА:

## Шаг 2. Создание класса контекста для базы данных

- Создайте папку **Data**.
- Добавьте файл **Data/MyDBContext.cs** со следующим кодом:





# Создание класса контекста для базы данных

---

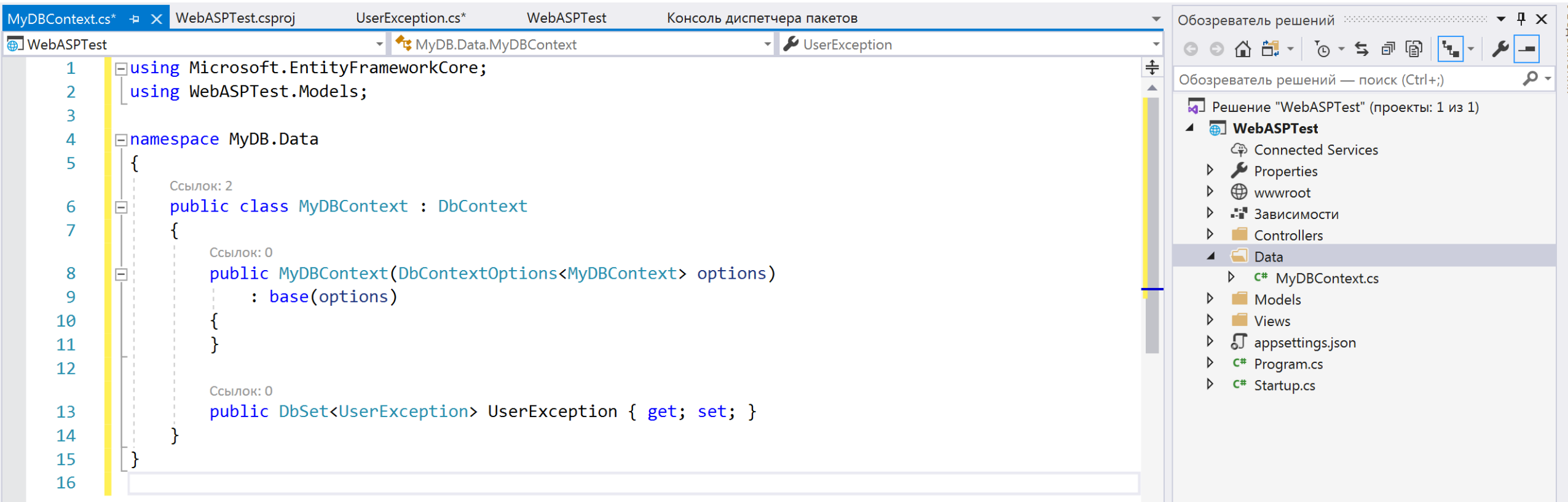
```
using Microsoft.EntityFrameworkCore;
using WebASPTest.Models;

namespace MyDB.Data
{
    public class MyDBContext : DbContext
    {
        public MyDBContext(DbContextOptions<MyDBContext> options)
            : base(options)
        {
        }

        public DbSet<UserException> UserException { get; set; }
    }
}
```

# Создание класса контекста для базы данных

Объект **MyDbContext** обрабатывает задачу подключения к базе данных и сопоставления объектов **UserException** с записями базы данных.



The screenshot displays the Visual Studio IDE with the following components:

- Top Tab Bar:** Shows open files: `MyDbContext.cs*`, `WebASPTTest.csproj`, `UserException.cs*`, `WebASPTTest`, and `Консоль диспетчера пакетов`.
- Left Panel (Solution Explorer):** Shows the project structure for `WebASPTTest`. The `MyDB.Data` folder is expanded, showing the `MyDbContext` class.
- Editor:** Displays the code for `MyDbContext` in `MyDB.Data`. The code is as follows:

```
1 using Microsoft.EntityFrameworkCore;
2 using WebASPTTest.Models;
3
4 namespace MyDB.Data
5 {
6     // Ссылка: 2
7     public class MyDbContext : DbContext
8     {
9         // Ссылка: 0
10        public MyDbContext(DbContextOptions<MyDbContext> options)
11            : base(options)
12        {
13        }
14
15        // Ссылка: 0
16        public DbSet<UserException> UserException { get; set; }
```
- Right Panel (Solution Explorer):** Shows the project structure for `WebASPTTest`. The `Data` folder is expanded, showing the `MyDbContext.cs` file.

**Контекст базы данных** регистрируется с помощью контейнера внедрения зависимостей в методе **ConfigureServices** в файле **Startup.cs**:

## Шаг 3. Регистрация контекста базы данных

---

Добавьте следующие инструкции **using** в начало файла **Startup.cs**.

```
using MyDB.Data;  
using Microsoft.EntityFrameworkCore;
```

Добавьте выделенный ниже код в Startup.ConfigureServices:

```
public void ConfigureServices(IServiceCollection services)  
{  
    services.AddControllersWithViews();  
    services.AddDbContext<MyDBContext>(options =>  
  
options.UseSqlServer(Configuration.GetConnectionString("MyDBContext")));  
}
```

Имя **строки подключения** передается **в контекст** путем вызова метода для объекта **DbContextOptions**.

При локальной разработке система конфигурации ASP.NET Core считывает строку подключения из файла **appsettings.json**.

## Шаг 4. Добавление строки подключения базы данных

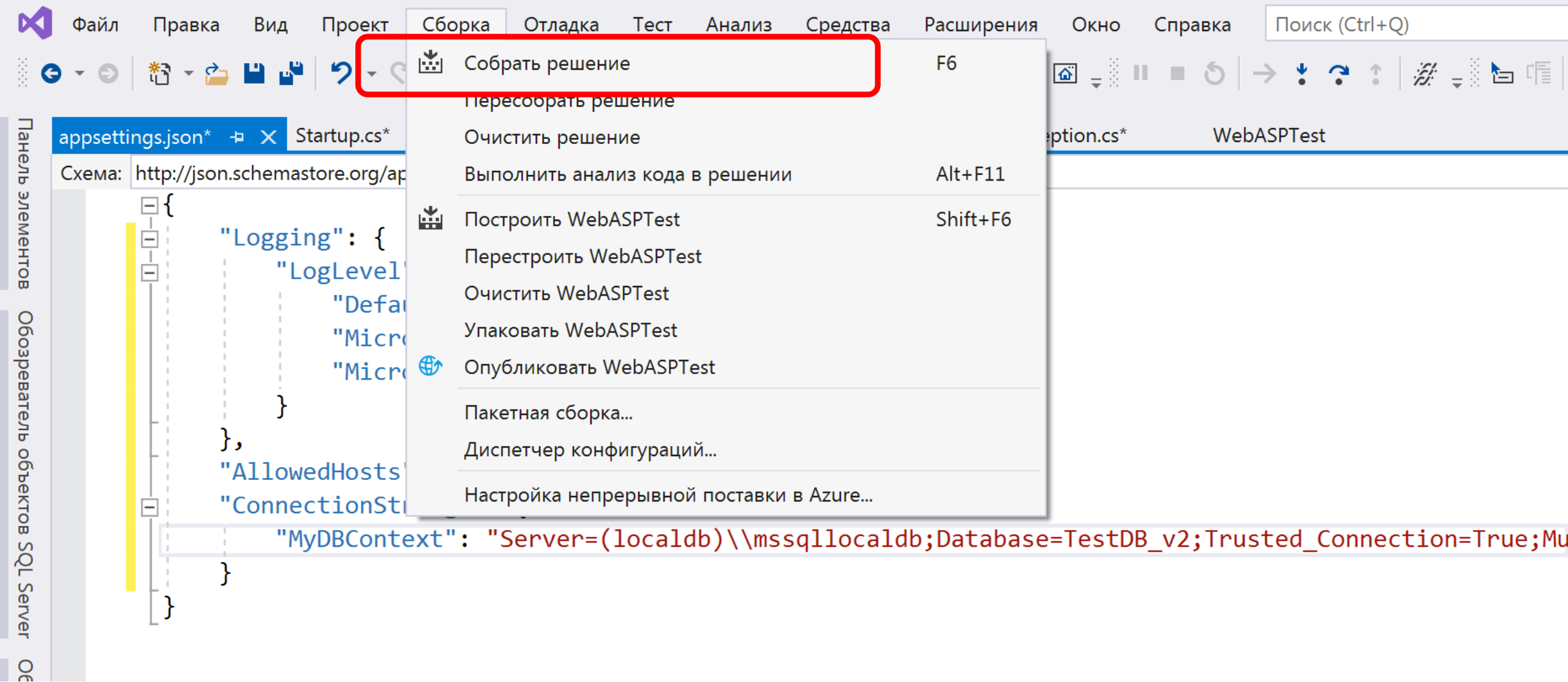
---

Добавьте строку подключения в файл **appsettings.json**:

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "MyExceptionContext":
"Server=(localdb)\\mssqllocaldb;Database=TestDB_v2;Trusted_Connection=True;MultipleActive
ResultSets=true"
  }
}
```

## Шаг 5. Сборка проекта

Выполните **сборку проекта**, чтобы проверить его на ошибки компиляции.



## Шаг 6. Формирования шаблонов страниц исключений

---

Используйте средство формирования шаблонов, чтобы создать страницы для операций **создания, чтения, обновления и удаления (CRUD)** для модели логгера исключений

В **Обозревателе решений** щелкните правой кнопкой мыши папку *Контроллеры* и выберите **Добавить > Создать шаблонный элемент**.

В диалоговом окне **Добавление шаблона** выберите **Контроллер MVC с представлениями, использующий Entity Framework > Добавить**.

## ▲ Установлено

Макет

## ▲ Общие

API

▸ MVC

Страницы Razor

Удостоверение



Область MVC



Контроллер MVC с действиями чтения и записи



Контроллер MVC с представлениями, использующий Entity Framework



Контроллер MVC — пустой



Контроллер API — пустой



Контроллер API с действиями чтения и записи



Контроллер API с действиями, использующий Entity Framework



Представление MVC



Razor Pages на основе Entity Framework (CRUD)



Страница Razor



Страница Razor, использующая Entity Framework

**Контроллер API с действиями, использующий Entity Framework**кем Майкрософт  
v1.0.0.0

Контроллер API с действиями REST для создания, чтения, изменения, удаления и вывода списка сущностей из контекста данных Entity Framework.

Идентификатор: ApiControllerWithContext  
Scaffolder

[Щелкните здесь для поиска расширений для формирования шаблонов в Интернете.](#)

Добавить

Отмена

Выполните необходимые действия в диалоговом окне **Добавление контроллера**:

- **Класс модели:** *UserException*

- **Класс контекста данных:** *MyDBContext*

Добавить Контроллер MVC с представлениями, использующий Entity Framework

Класс модели: UserException (WebASPTTest.Models)

Класс контекста данных: MyDBContext (MyDB.Data) +

Представления:

- ☒ Создать представления
- ☒ Справочные библиотеки сценариев
- ☒ Использовать страницу макета:

...

(Оставить пустым, если значение задано в файле Razor \_viewstart)

Имя контроллера: UserExceptionsController

Добавить Отмена

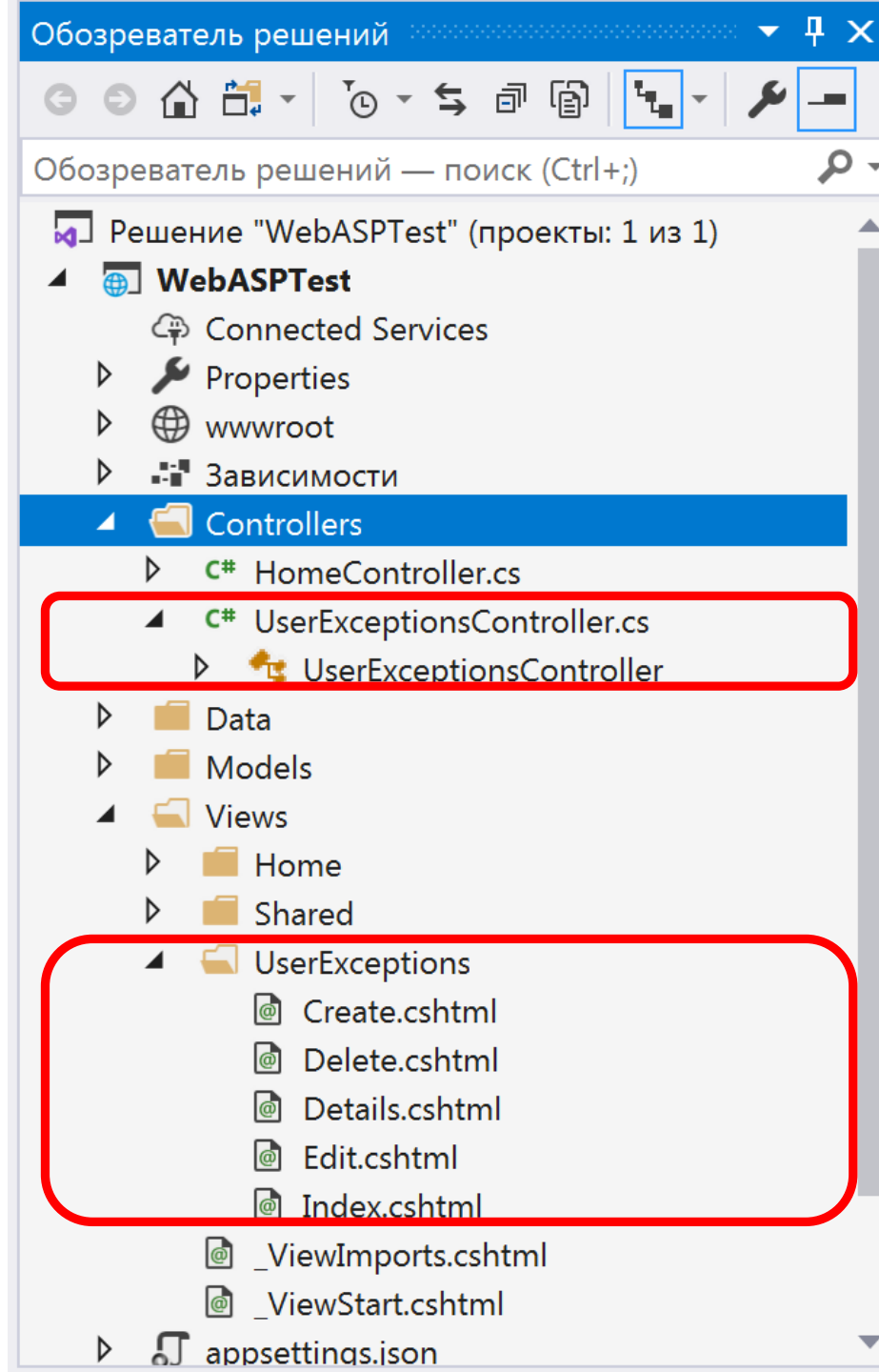


Visual Studio создаст следующие  
компоненты:

- **контроллер Исключений**  
(*Controllers/UserExceptionHandler.cs*);

- **файлы представления Razor** для  
страниц Create, Delete, Details, Edit и  
Index  
(*Views/UserExceptions/\*.cshtml*).

Автоматическое создание этих файлов  
называется **формированием  
шаблонов**.



## Шаг 7. Подключение контроллера исключений исключений

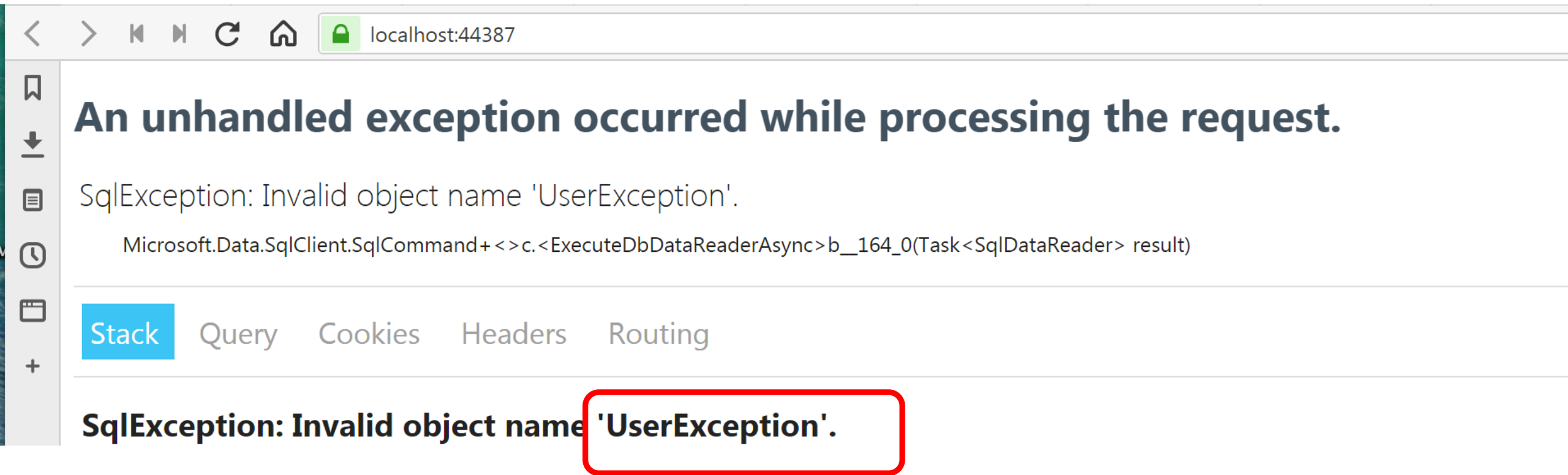
---

В файле **Startup.cs** в карте маршрутизации укажите имя **контроллера UserExceptions**:

```
app.UseEndpoints(endpoints =>
{
    endpoints.MapControllerRoute(
        name: "default",
        pattern: "{controller=UserExceptions}/{action=Index}/{id?}");
});
```

## Шаг 8. Тестирование проекта

При запуске проекта на выполнение (F5), появляется ошибка



Объект POCO модели с именем **UserException** ОБЯЗАН иметь точно такое же имя, как и таблица базы данных, которую он представляет в проекте. В нашей базе данных TestDB\_v2 таблица с исключениями называется **UserExceptions**

## Шаг 8. Тестирование проекта

Поэтому в папке **Models** в файле **UserException.cs** это несоответствие устраняем, внося следующие изменения:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Threading.Tasks;  
using System.ComponentModel.DataAnnotations;  
using Microsoft.EntityFrameworkCore;  
using System.ComponentModel.DataAnnotations.Schema;
```

```
namespace WebASPTest.Models  
{  
    [Table ("UserExceptions")]  
    public class UserException
```

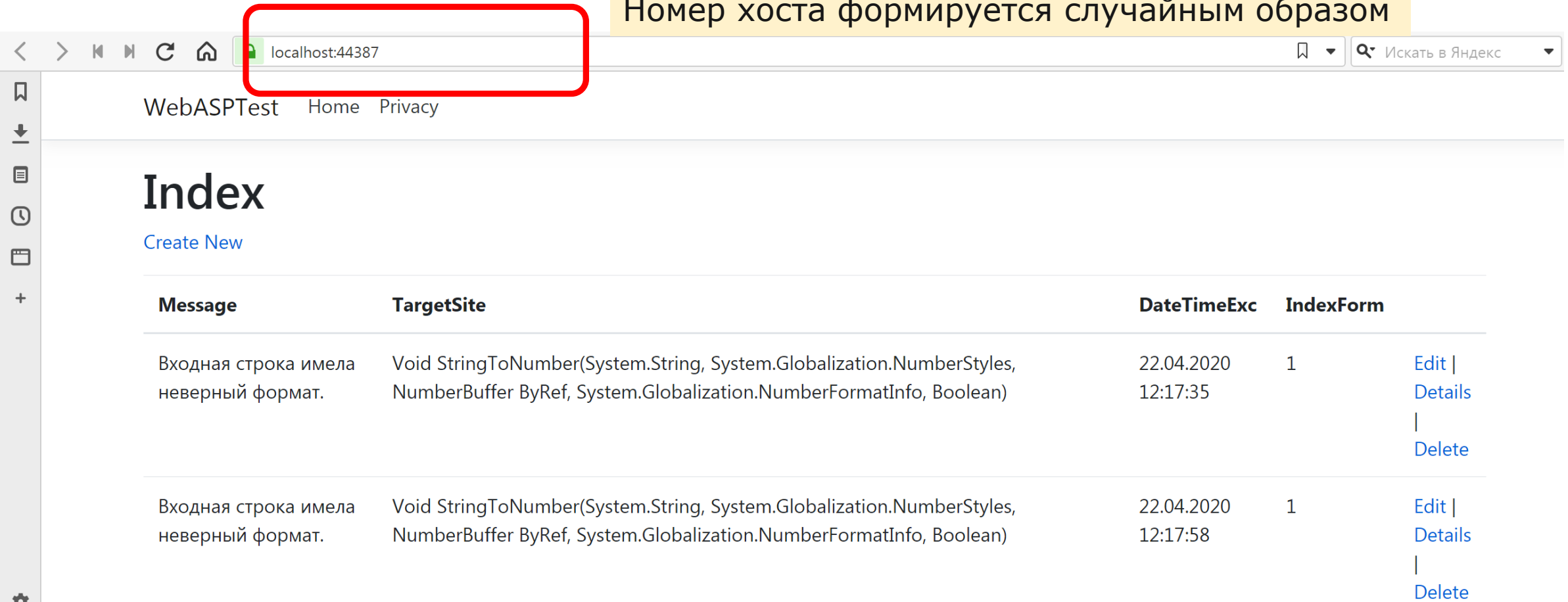
ССЫЛОК: 18

```
{
```

ССЫЛОК: 11

## Шаг 8. Тестирование проекта

Номер хоста формируется случайным образом



WebASPTest Home Privacy

### Index

[Create New](#)

Message	TargetSite	DateTimeExc	IndexForm
Входная строка имела неверный формат.	Void StringToNumber(System.String, System.Globalization.NumberStyles, NumberBuffer ByRef, System.Globalization.NumberFormatInfo, Boolean)	22.04.2020 12:17:35	1 <a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Входная строка имела неверный формат.	Void StringToNumber(System.String, System.Globalization.NumberStyles, NumberBuffer ByRef, System.Globalization.NumberFormatInfo, Boolean)	22.04.2020 12:17:58	1 <a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>

Теперь наш тонкий клиент работает, выполняя требуемые операции с базой данных. При этом **за внешний вид** отображаемых данных «отвечают» **представления**, за **операции с набором данных** отвечает **контроллер**, за **соответствие структур данных** отвечает **модель**, **сервер БД только хранит базу данных**



# Edit

## UserException

Message

except\_new

TargetSite

TargetSite\_new

DateTimeExc

29.04.2020 12:12

IndexForm

3



Save

---



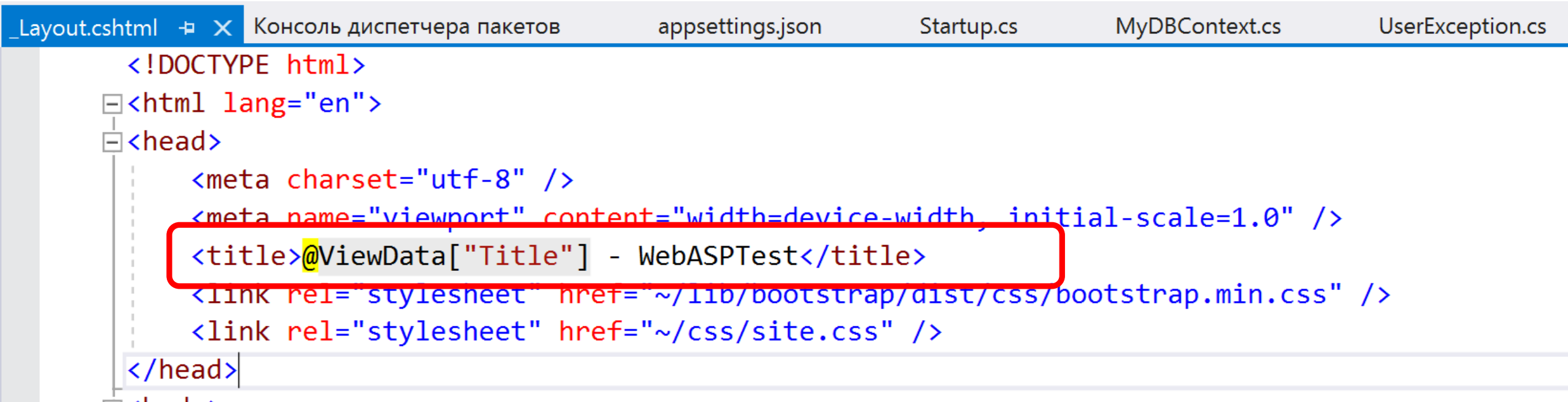
## Шаг 9. Изменение представлений и страниц макета

Меню на каждой странице имеют одинаковый макет.

Макет меню реализован в файле **Views/Shared/\_Layout.cshtml**.

Откройте файл **Views/Shared/\_Layout.cshtml**.

С помощью шаблонов макета можно в одном месте задать макет контейнера HTML для всего сайта и затем использовать его на разных страницах сайта.



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>@ViewData["Title"] - WebASPTest</title>
  <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
  <link rel="stylesheet" href="~/css/site.css" />
</head>
```

Замените в этом файле в `<head>` и `<body>` имя нашего проекта **WebASPTest** на **Логгер исключений** и мы получим требуемый заказчиков интерфейс тонкого клиента



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device width, initial-scale=1.0" />
  <title>@ViewData["Title"] - WebASPTest</title>
  <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
  <link rel="stylesheet" href="~/css/site.css" />
</head>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device width, initial-scale=1.0" />
  <title>@ViewData["Title"] - Логгер исключений</title>
  <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
  <link rel="stylesheet" href="~/css/site.css" />
</head>
```

# Index

Create New

Message	TargetSite	DateTimeExc	IndexForm	
Входная строка имела неверный формат.	Void StringToNumber(System.String, System.Globalization.NumberStyles, NumberBuffer ByRef, System.Globalization.NumberFormatInfo, Boolean)	22.04.2020 12:17:35	1	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Входная строка имела неверный формат.	Void StringToNumber(System.String, System.Globalization.NumberStyles, NumberBuffer ByRef, System.Globalization.NumberFormatInfo, Boolean)	22.04.2020 12:17:58	1	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
except_new	TargetSite_new	29.04.2020 12:12:00	3	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Входная строка имела неверный формат.	Void StringToNumber(System.String, System.Globalization.NumberStyles, NumberBuffer ByRef, System.Globalization.NumberFormatInfo, Boolean)	22.04.2020 07:16:38	3	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Входная строка имела неверный формат.	Void StringToNumber(System.String, System.Globalization.NumberStyles, NumberBuffer ByRef, System.Globalization.NumberFormatInfo, Boolean)	22.04.2020 07:16:42	3	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>

## Результаты фазы 5

---

На текущий момент мы разработали **приложение доступа к данным**, являющееся **desktop-клиентом** («толстым»), а также «тонкий» **ASP.Net Web-клиент** - «Логгер-консоль», предназначенный для мониторинга исключений, но он не выполняет регистрацию инцидентов.

Чтобы регистрировать инциденты должен быть полностью реализован функционал серверов приложений и сервера БД.

Это нам предстоит сделать далее 😊

Но лабораторную работу 5 уже можно делать 😊