Распределенные информационно-аналитические системы

Практическое занятие № 1. «Основы в ASP.NET Core. Часть 1»

Профессор кафедры КБ-2: д.т.н. Шатовкин Р.Р.

Учебные вопросы:

1. **Введение в ASP.NET Core.**
2. **Первое приложение на ASP.NET Core с .NET CLI.**
3. **Первое приложение в Visual Studio.**

1. Введение в ASP.NET Core

Что такое ASP.NET Core

ASP.NET Core представляет технологию для создания веб-приложений на платформе .NET, развиваемую компанией Microsoft. В качестве языков программирования для разработки приложений на ASP.NET Core используются C# и F#.

История ASP.NET фактически началась с выходом первой версии .NET в начале 2002 года и с тех пор ASP.NET и .NET развивались параллельно: выход новой версии .NET знаменовал выход новой версии ASP.NET, поскольку ASP.NET работает поверх .NET. В то же время изначально ASP.NET была нацелена на работу исключительно в Windows на веб-сервере IIS (хотя благодаря проекту Mono приложения на ASP.NET можно было запускать и на Linux).

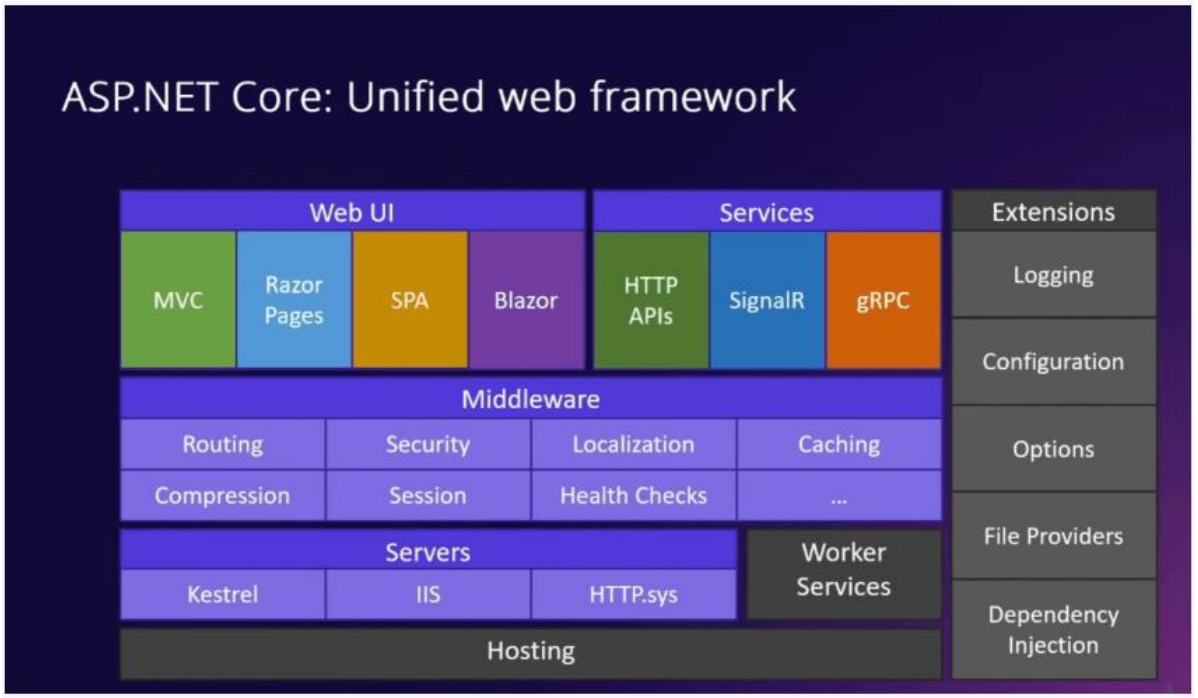
Однако 2014 год ознаменовал большие перемены, фактически революцию в развитии платформы: компания Microsoft взяла курс на развитии ASP.NET как кроссплатформенной технологии, которая развивается как opensource-проект. Данное развитие платформы в дальнейшем получило название ASP.NET Core, собственно как ее официально именуют Microsoft до сих пор. Первый релиз обновленной платформы увидел свет в июне 2016 года. Теперь она стала работать не только на Windows, но и на MacOS и Linux. Она стала более легковесной, модульной, ее стало проще конфигурировать, в общем, она стала больше отвечать требованиям текущего времени.

Текущая версия ASP.NET Core вышла вместе с релизом .NET 7 в ноябре 2022 года.

ASP.NET Core теперь полностью является opensource-фреймворком. Все исходные файлы фреймворка доступны на github в репозитории https://github.com/dotnet/aspnetcore/.

Архитектура и модели разработки

Текущую архитектуру платформы ASP.NET Core можно выразить следующим образом:



На самом верхнем уровне располагаются различные модели взаимодействия с пользователем. Это технологии построения пользовательского интерфейса и обработки ввода пользователя, как MVC, Razor Pages, SPA (Single Page Application - одностраничные приложения с использованием Angular, React, Vue) и Balzor. Кроме того, это сервисы в виде встроенных HTTP API, библиотеки SignalR или сервисов GRPC.

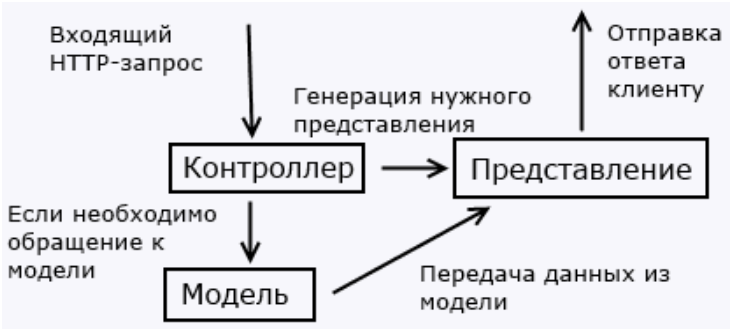
Все эти технологии базируются и/или взаимодействуют с чистым ASP.NET Core, который представлен прежде всего различными встроенными компонентами middleware – компонентами, которые применяются для обработки запроса. Кроме того, технологии высшего уровня также взаимодействуют с различными расширениями, которые не являются непосредственной частью ASP.NET Core, как расширения для логгирования, конфигурации и т.д.

И на самом нижнем уровне приложение ASP.NET Core работает в рамках некоторого веб-сервера, например, Kestrel, IIS, библиотеки HTTP.sys.

Это вкратце архитектура платформы, теперь рассмотрим моделей разработки приложения ASP.NET Core:

Прежде всего это **базовый ASP.NET Core**, который поддерживает все основные моменты, необходимые для работы современного веб-приложения: маршрутизация, конфигурация, логгирования, возможность работы с различными системами баз данных и т.д. В ASP.NET Core 6 в фреймворк был добавлен так называемый **Minimal API** – минимизированная упрощенная модель, который еще упростила процесс разработки и написания кода приложения. Все остальные модели разработки работаю поверх базового функционала ASP.NET Core

ASP.NET Core MVC представляет в общем виде построения приложения вокруг трех основных компонентов – Model (модели), View (представления) и Controller (контроллеры), где модели отвечают за работу с данными, контроллеры представляют логику обработки запросов, а представления определяют визуальную составляющую.



ASP.NET Core MVC в .NET 7

Razor Pages представляет модель, при котором за обработку запроса отвечают специальные сущности – страницы Razor Pages. Каждую отдельную такую сущность можно ассоциировать с отдельной веб-страницей.

ASP.NET Core Web API представляет реализацию паттерна REST, при котором для каждого типа http-запроса (GET, POST, PUT, DELETE) предназначен отдельный ресурс. Подобные ресурсы определяются в виде методов контроллера Web API. Данная модель особенно подходит для одностраничных приложений, но не только.

Blazor представляет фреймворк, который позволяет создавать интерактивные приложения как на стороне сервера, так и на стороне клиента и позволяет задействовать на уровне браузера низкоуровневый код WebAssembly.

Особенности платформы

ASP.NET Core работает поверх платформы .NET и, таким образом, позволяет задействовать весь ее функционал.

В качестве языков разработки применяются языки программирования, поддерживаемые платформой .NET. Официально встроенная поддержка для проектов ASP.NET Core есть у языков C# и F#

ASP.NET Core представляет кросс-платформенный фреймворк, приложения на котором могут быть развернуты на всех основных популярных операционных системах: Windows, Mac OS, Linux. И таким образом, с помощью ASP.NET Core мы можем как создавать кросс-платформенные приложения на Windows, на Linux и Mac OS, так и запускать на этих ОС.

Благодаря модульности фреймворка все необходимые компоненты веб-приложения могут загружаться как отдельные модули через пакетный менеджер Nuget.

Поддержка работы с большинством распространенных систем баз данных: MS SQL Server, MySQL, Postgres, MongoDB

ASP.NET Core характеризуется расширяемостью. Фреймворк построен из набора относительно независимых компонентов. И мы можем либо использовать встроенную реализацию этих компонентов, либо расширить их с помощью механизма наследования, либо вовсе создать и применять свои компоненты со своим функционалом.

Богатый инструментарий для разработки приложений. В качестве инструментария разработки мы можем использовать такую среду разработки с богатым функционалом, как Visual Studio от компании Microsoft.

Также можно использовать для разработки среду Rider от компании JetBrains.

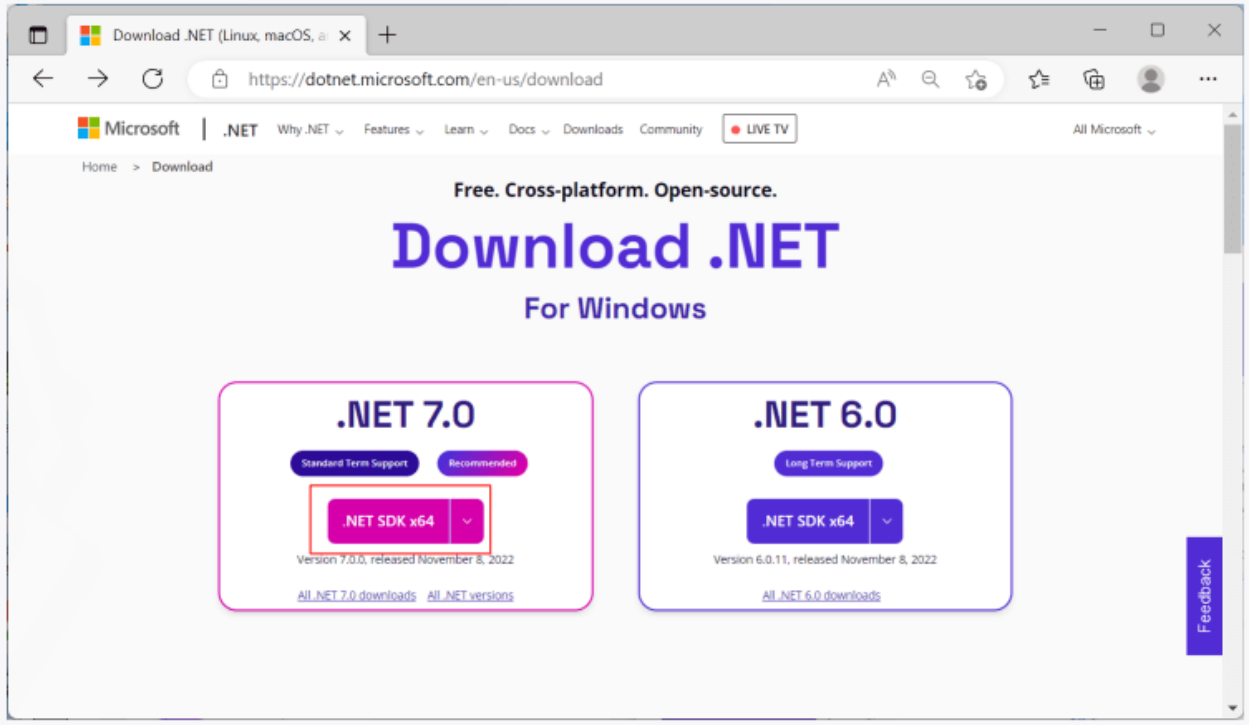
Кроме того, имеющаяся оснастка .NET CLI позволяет создавать и запускать проекты ASP.NET в консоли. И таким образом, для написания кода можно использовать обычных текстовый редактор, например, Visual Studio Code.

2. Первое приложение на ASP.NET Core с .NET CLI

Создадим первую программу на ASP.NET Core. Что нам для этого потребуется? Прежде всего необходим текстовый редактор для написания кода программы. В данном случае я буду использовать в качестве текстового редактора Visual Studio Code.

Также для компиляции и запуска программы нам потребуется .NET SDK.

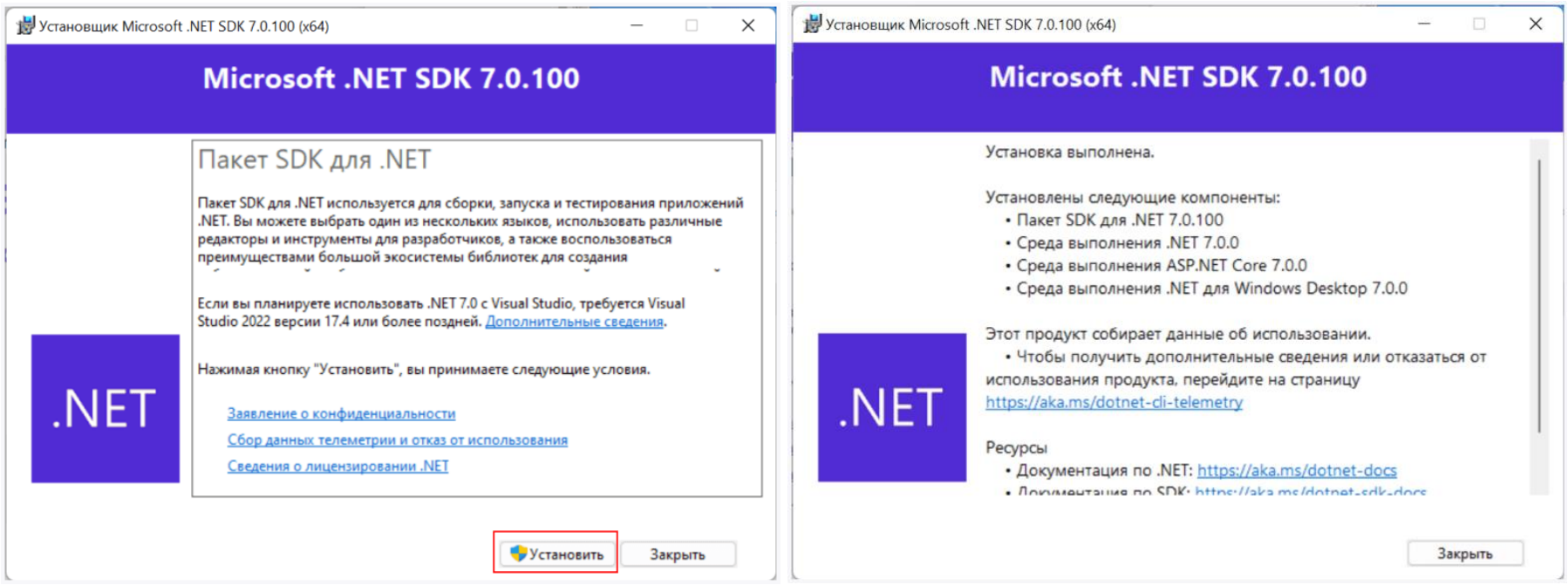
Для его установки перейдем на официальный сайт по ссылке .NET SDK.



Загрузка .NET SDK для ASP.NET Core

Выберем последнюю на данный момент версию – .NET SDK 7 и загрузим его.

Затем запустим программу установки:



Установка .NET SDK для ASP.NET Core Установка .NET SDK для ASP.NET Core и C#

После установки .NET SDK для первого проекта определим какую-нибудь папку.

Например, будет папка C:\dotnet\aspnet\helloapp.

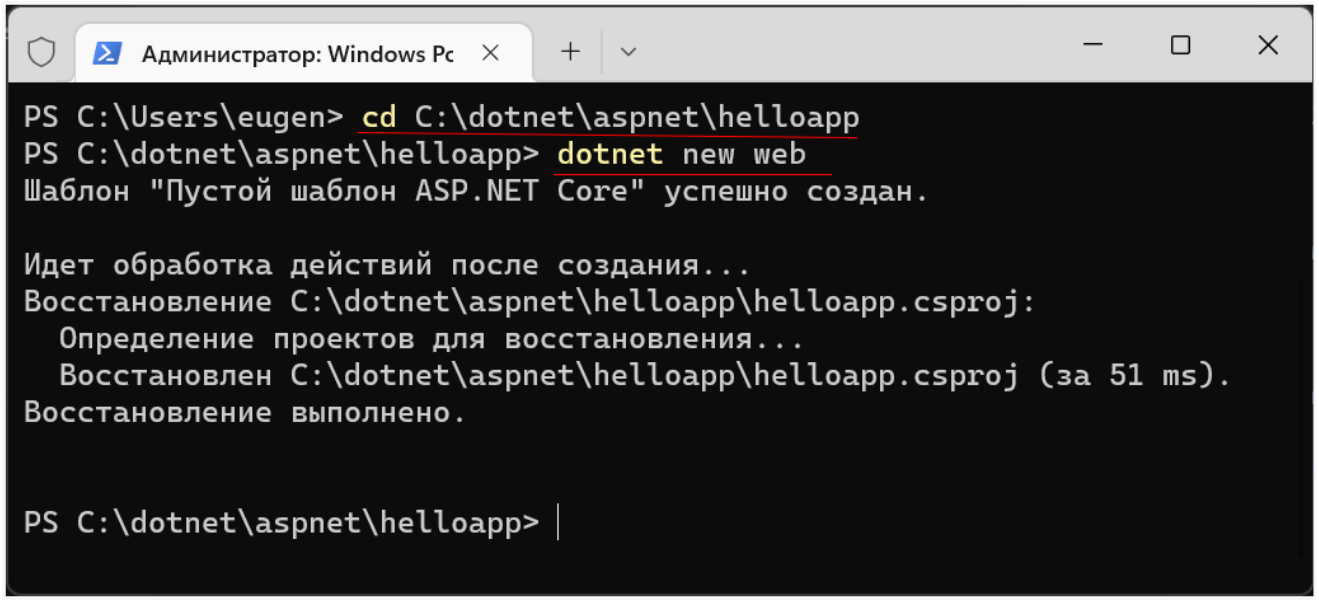
Откроем терминал/командную строку и перейдем к созданной папке проекта с помощью команды cd:

```
cd C:\dotnet\aspnet\helloapp
```

В данном случае мы для создания и запуска проекта мы будем использовать встроенную инфраструктуру .NET CLI, которая устанавливается вместе с .NET SDK.

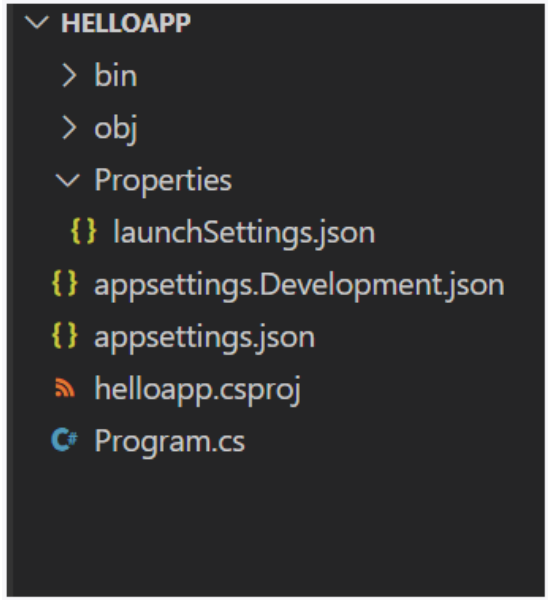
Для создания проекта в .NET CLI применяется команда dotnet new, после которой указывается тип проекта. Для ASP.NET Core есть ряд встроенных типов проектов. В данном случае мы будем использовать самый простейший тип – web. Поэтому введем в терминале команду:

```
dotnet new web
```



Создание первого проекта ASP.NET Core и C# с помощью .NET CLI

После выполнения этой команды у нас будет создан следующий проект:



Первый проект ASP.NET Core на C# в Visual Studio Code

Структура проекта ASP.NET Core

Рассмотрим **базовую структуру простейшего стандартного проекта ASP.NET Core**:

Dependencies: все добавленные в проект пакеты и библиотеки, иначе говоря зависимости.

Properties: узел, который содержит некоторые настройки проекта. В частности, в файле **launchSettings.json** описаны настройки запуска проекта, например, адреса, по которым будет запускаться приложение.

appsettings.json: файл конфигурации приложения в формате json.

appsettings.Development.json: версия файла конфигурации приложения, которая используется в процессе разработки.

helloapp.csproj: стандартный файл проекта C#, который соответствует названию проекта (по умолчанию названию каталога) и описывает все его настройки.

Program.cs: главный файл приложения, с которого и начинается его выполнение. Код этого файла настраивает и запускает веб-приложение

Например, посмотрим на содержимое файла **helloapp.csproj**

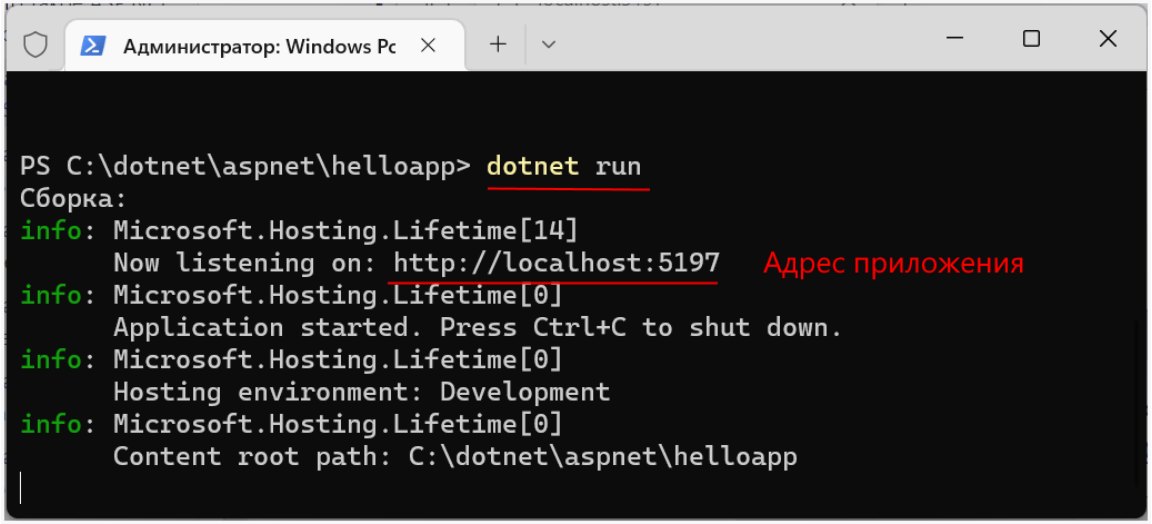
```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net7.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>

</Project>
```

Ключевой компонент здесь – атрибут `Sdk="Microsoft.NET.Sdk.Web"`, который собственно и определяет, что приложение будет использовать SDK "Microsoft.NET.Sdk.Web", который предназначен именно для веб-проектов.

Запуск проекта

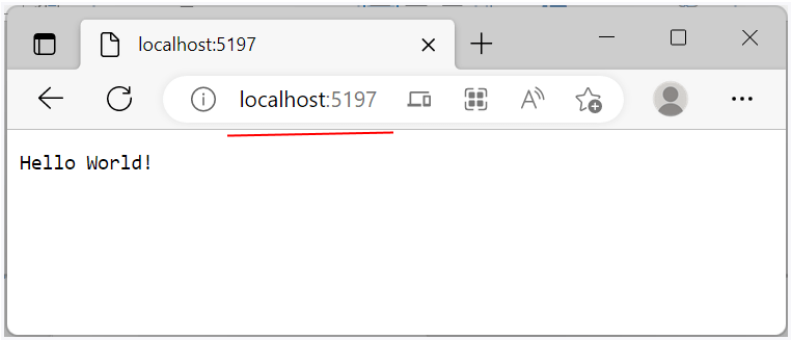
Проект по умолчанию не представляет какой-то грандиозной функциональности, тем не менее этот проект мы уже можем запустить. Итак, запустим проект. Для этого выполним команду:

```
dotnet run
```



Запуск проекта ASP.NET Core и C# с помощью .NET CLI

При запуске в консоли мы можем увидеть адрес, по которому мы можем обращаться к приложению. В моем случае это адрес "http://localhost:5197/". И я могу обратиться по этому адресу к приложению в браузере и увидеть в нем строку "Hello World!" – результат работы кода по умолчанию из файла **Program.cs**:



Первое приложение на ASP.NET Core на C# с .NET CLI

Запуск приложения и файл Program.cs

Рассмотрим код файла **Program.cs**, который создает подобное приложение:

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

app.MapGet("/", () => "Hello World!");

app.Run();

```

Это так называемое **Minimal API** – упрощенная минимизированная модель для запуска веб-приложения в ASP.NET.

Приложение в ASP.NET Core представляет объект **Microsoft.AspNetCore.Builder.WebApplication**. Этот объект настраивает всю конфигурацию приложения, его маршруты, используемые зависимости и т.д.

Исходный код класса на github: https://github.com/dotnet/aspnetcore/blob/main/src/DefaultBuilder/src/WebApplication.cs

Для создания объекта **WebApplication** необходим специальный класс-строитель – **WebApplicationBuilder**. И в файле **Program.cs** вначале создается данный объект с помощью статического метода **WebApplication.CreateBuilder**:

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

В качестве параметра в метод передаются аргументы, которые передаются приложению при запуске.

Получив объект **WebApplicationBuilder**, у него вызывается метод **Build()**, который собственно и создает объект **WebApplication**:

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

С помощью объекта **WebApplication** можно настроить всю инфраструктуру приложения – его конфигурацию, маршруты и так далее. В файле **Program.cs** по умолчанию для приложения определяется один маршрут:

```
app.MapGet("/", () => "Hello World!");
```

Метод **MapGet()** в качестве первого параметра принимает путь, по которому можно обратиться к приложению. В данном случае это путь "/", то есть по сути корень веб-приложения – имя домена и порта, после которых может идти слеш, например, https://localhost:7256/

В качестве второго параметра в метод **MapGet()** передаются обработчик запроса по этому маршруту в виде функции. Здесь это лямбда-выражение, которое возвращает строку "Hello World!". Именно поэтому при обращении к приложению мы увидим данную строку в браузере.

И в конце необходимо запустить приложение. Для этого у класса **WebApplication** вызывается метод **Run()**:

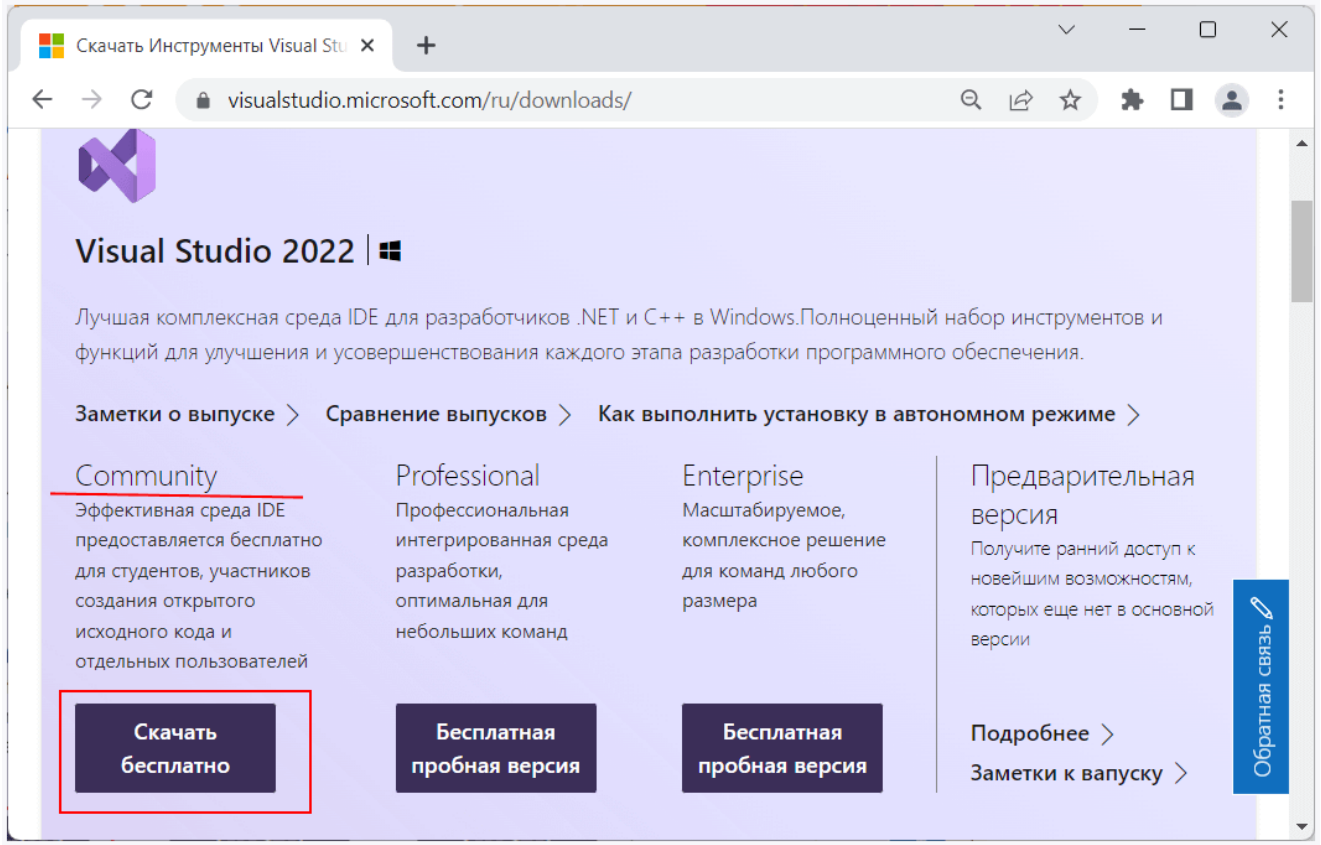
```
app.Run();
```

В итоге запустится приложение в виде консоли, и мы сможем обращаться к приложению из различных браузеров.

3. Первое приложение в Visual Studio

В прошлом учебном вопросе было рассмотрено создание первого проекта ASP.NET Core с помощью .NET CLI с компиляцией и запуском проекта в терминале. Это самый простой способ для создания приложений ASP.NET. Однако также мы можем использовать среду разработки Visual Studio, которая упрощает многие аспекты по работе с проектом ASP.NET. Рассмотрим, как создавать проект в Visual Studio.

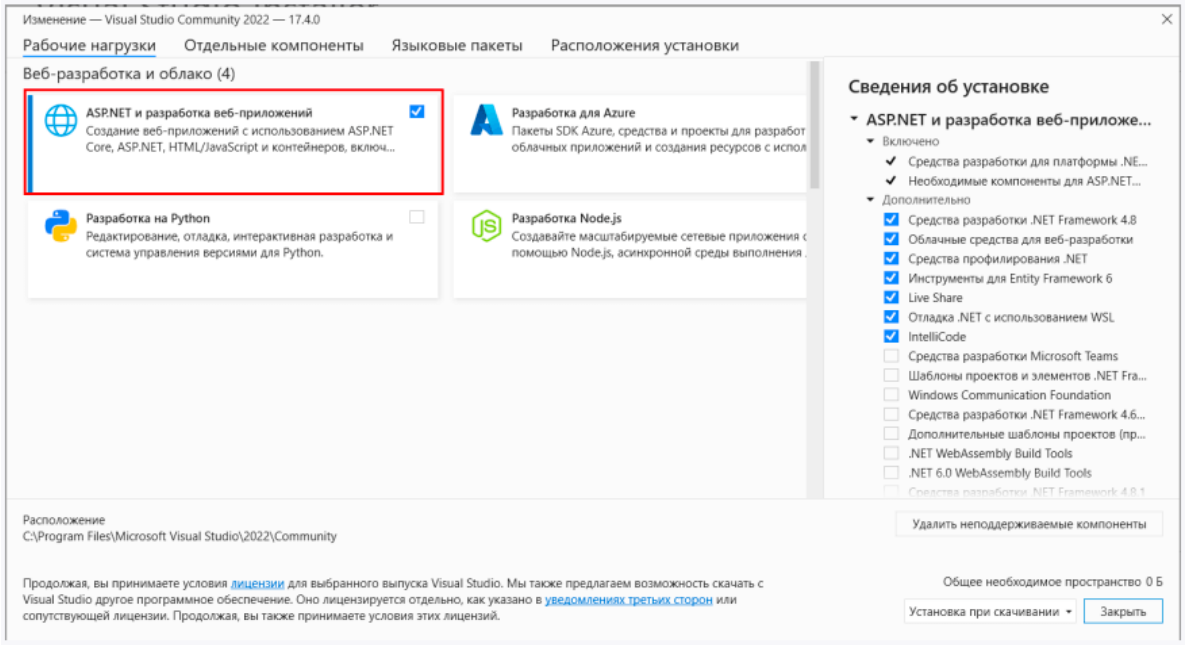
Итак, в начале загрузим бесплатный выпуск среды – Visual Studio Community 2022 по следующему адресу: [Microsoft Visual Studio 2022]



Установка Visual Studio 2022

После загрузки запустим программу установщика. В открывшемся окне нам будет предложено выбрать те компоненты, которые мы хотим установить вместе Visual Studio. Стоит отметить, что Visual Studio - очень функциональная среда разработки и позволяет разрабатывать приложения с помощью множества языков и платформ. В нашем случае нам будет интересовать прежде всего C# и .NET.

Чтобы добавить в Visual Studio поддержку проектов для ASP.NET Core, в программе установки среди рабочих нагрузок можно выбрать только пункт **ASP.NET и разработка веб-приложений**. Можно выбрать и больше опций или вообще все опции, однако стоит учитывать свободный размер на жестком диске - чем больше опций будет выбрано, соответственно тем больше места на диске будет занято.

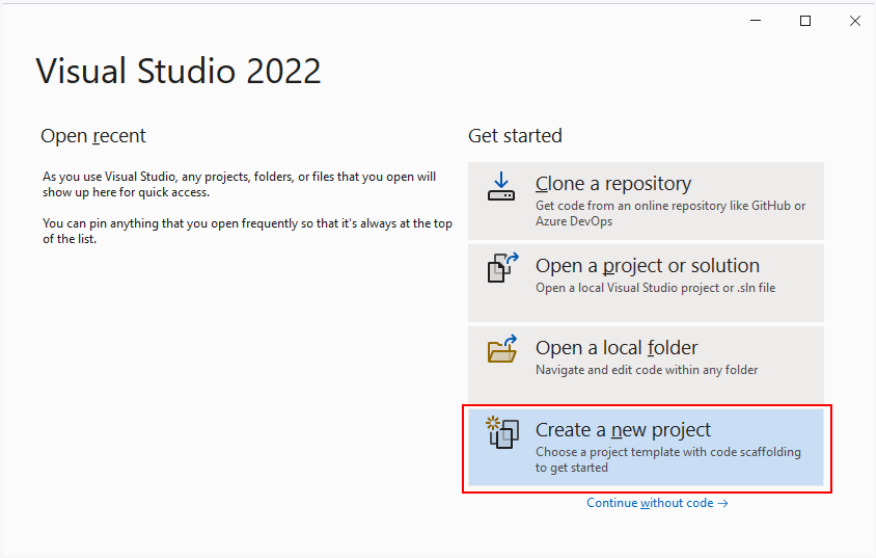


Установка Visual Studio 2022

И при инсталляции Visual Studio на ваш компьютер будут установлены все необходимые инструменты для разработки программ, в том числе фреймворк .NET.

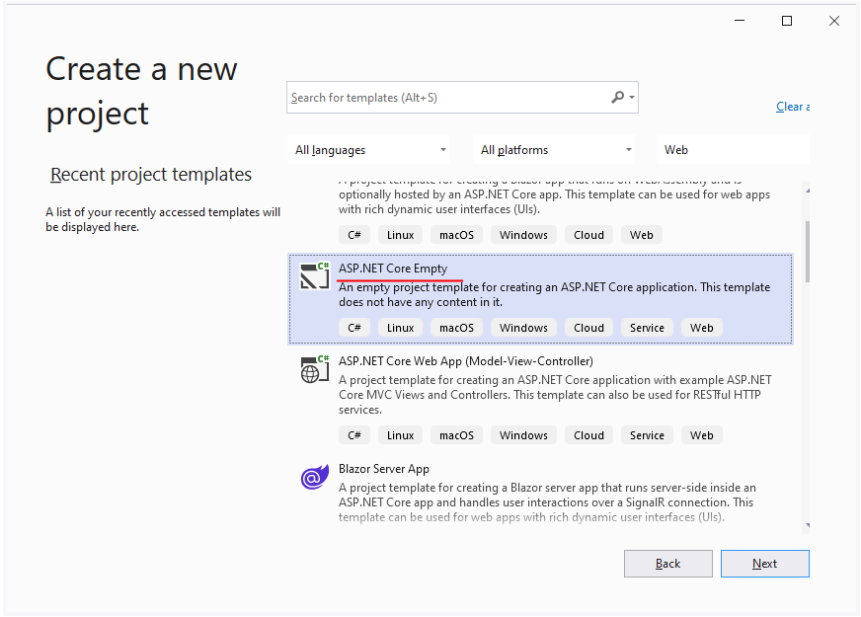
После завершения установки создадим первый проект на ASP.NET Core.

Вначале откроем Visual Studio. На стартовом экране выберем **Create a new project** (Создать новый проект)



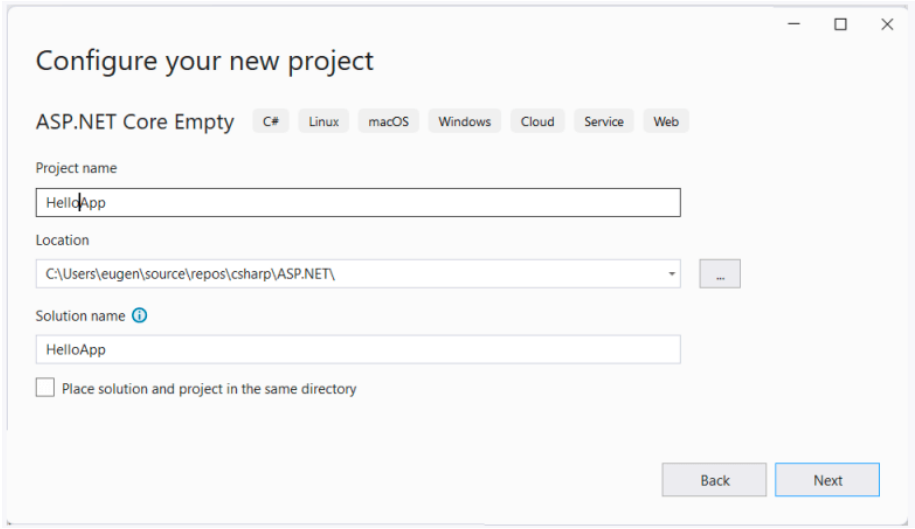
Создание первого проекта в Visual Studio 2022

На следующем окне в качестве типа проекта выберем **ASP.NET Core Empty**



Проект консольного приложения на C# и .NET в Visual Studio 2022

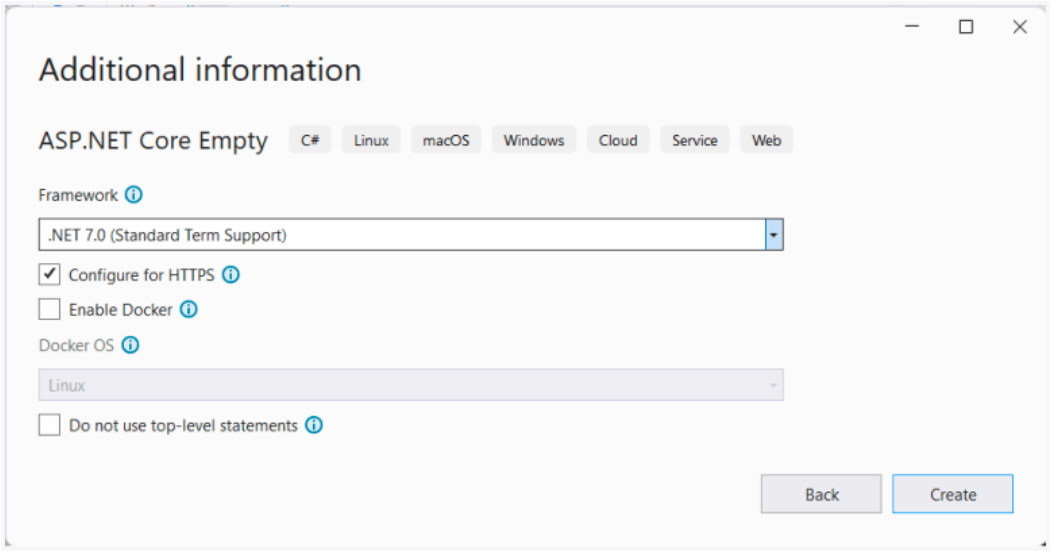
Далее на следующем этапе нам будет предложено указать имя проекта и каталог, где будет располагаться проект.



Создание первого приложения на C#

В поле **Project Name** дадим проекту какое-либо название. В примере это **HelloApp**.

На следующем окне Visual Studio предложит нам выбрать версию .NET, которая будет использоваться для проекта. Выберем здесь последнюю на данный момент версию - .NET 7.0.



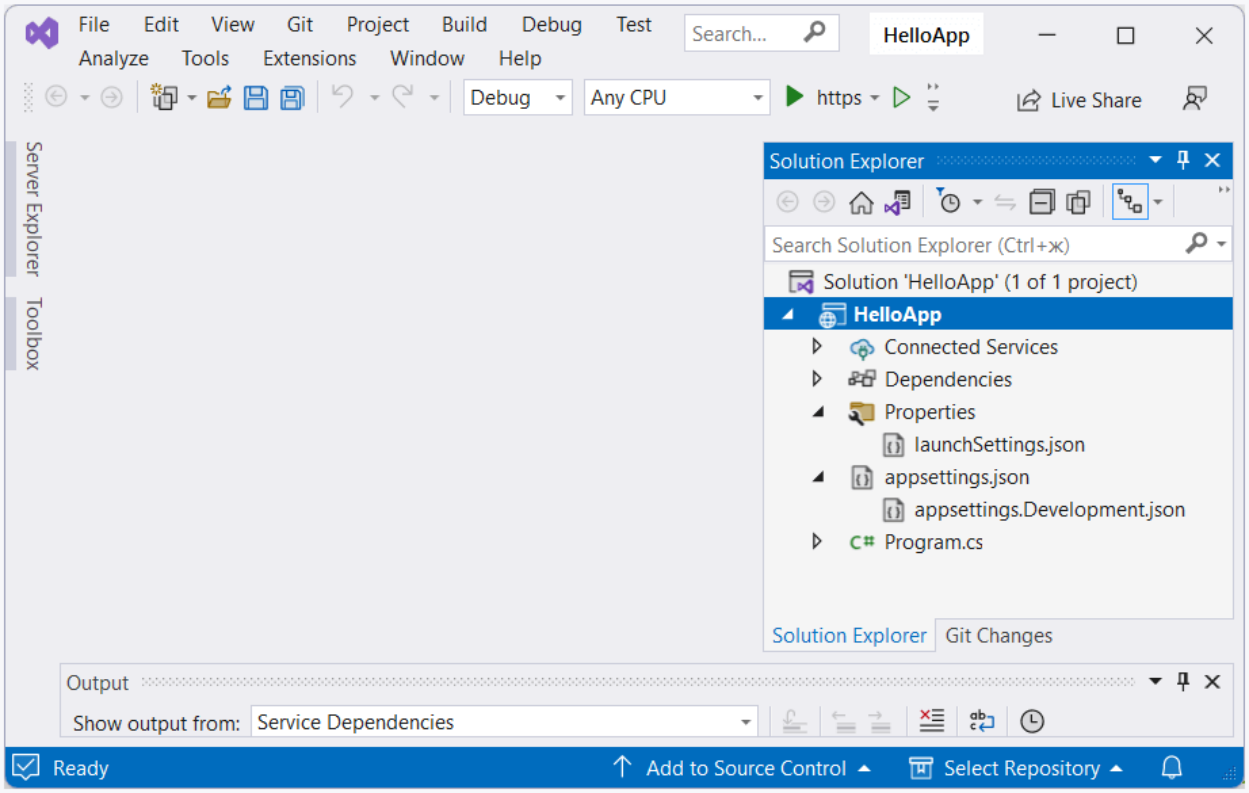
Установка C# и .NET в Visual Studio

Кроме того, здесь с помощью флажка **Configure for HTTPS** можно установить использование протокола https. По умолчанию этот флажок отмечен, это значит, что проект по умолчанию будет использовать протокол https.

Другой флажок – **Enable Docker** позволяет задействовать Docker. Если этот флажок установлен, то в поле ниже можно будет выбрать ОС, которая будет использоваться под Docker. Но в данном случае оставим этот флажок неотмеченным.

Третий флажок – **Do not use level statements** позволяет, как и в консольных проектах, отключить поддержку выражений верхнего уровня. Этот флажок также оставим неотмеченным.

Оставим все остальные настройки по умолчанию и нажмем на кнопку Create (Создать) для создания проекта. После этого Visual Studio создаст и откроет нам проект:



Первый проект ASP.NET Core на C#

Структура проекта ASP.NET Core

Рассмотрим базовую структуру стандартного проекта ASP.NET Core в Visual Studio.

Проект **ASP.NET Core Empty** содержит очень простую структуру – **необходимый минимум для запуска приложения**:

Connected Services: подключенные сервисы из Azure.

Dependencies: все добавленные в проект пакеты и библиотеки, иначе говоря зависимости.

Properties: узел, который содержит некоторые настройки проекта. В частности, в файле **launchSettings.json** описаны настройки запуска проекта, например, адреса, по которым будет запускаться приложение.

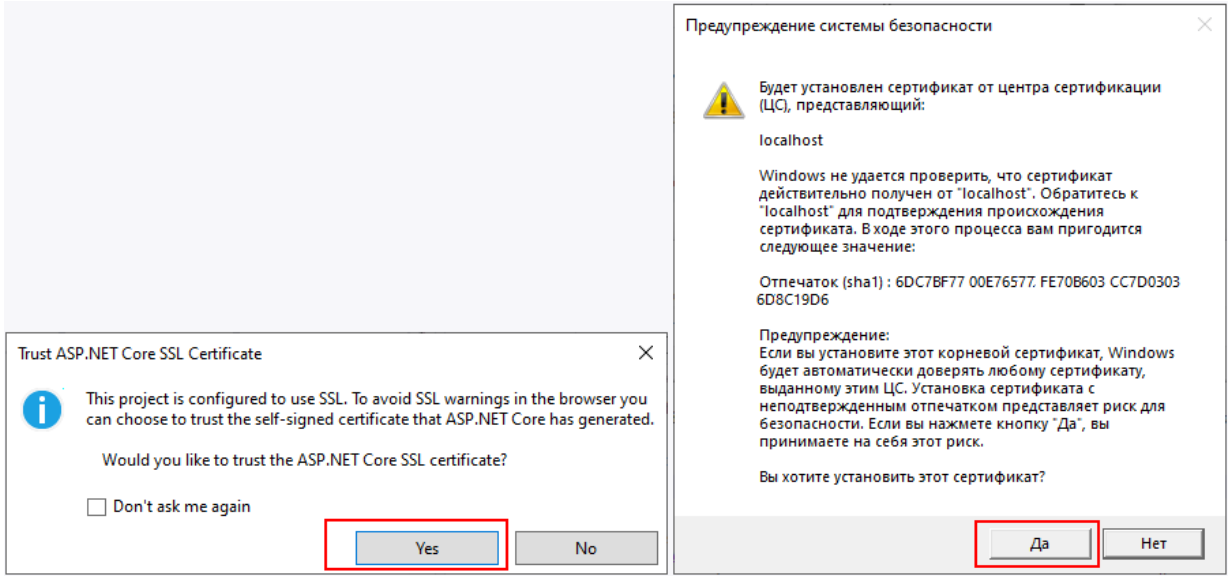
appsettings.json: файл конфигурации проекта в формате json.

appsettings.Development.json: версия файла конфигурации приложения, которая используется в процессе разработки

Program.cs: главный файл приложения, с которого и начинается его выполнение. Код этого файла настраивает и запускает веб-приложение.

Запуск проекта

Проект по умолчанию не представляет какой-то грандиозной функциональности, тем не менее этот проект мы уже можем запустить. Итак, запустим проект. При запуске нам может отобразиться окно, где надо подтвердить доверие для серфиката SSL, а также его установку.

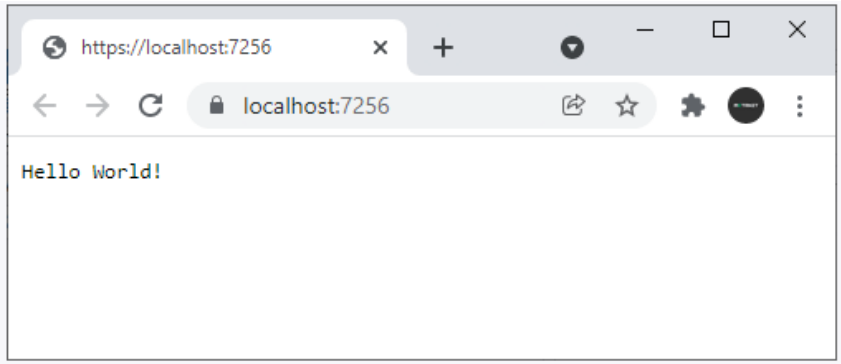


Подтверждение сертификата SSL при запуске приложения ASP.NET Core

Установка сертификата SSL при запуске приложения ASP.NET Core

И после подтверждения и установки сертификата отобразиться консоль, где выводится некоторая базовая информация о приложении:

И, кроме того, будет запущен браузер, где мы сможем лицезреть строку "Hello World!" – результат работы кода по умолчанию из файла **Program.cs**:



Первое приложение на ASP.NET Core