Оглавление

[Что такое: 1](#_Toc85244844)

[Коротко: разница web api MVC и Core: 1](#_Toc85244845)

[Как работать с web api: 2](#_Toc85244846)

[Пример 2](#_Toc85244847)

[Тип ответа: 2](#_Toc85244848)

[Маршрутизация или как понимает, что вернуть: 3](#_Toc85244849)

[Пример 3](#_Toc85244850)

[Как понимает, где просто класс программы, а где контроллер: 3](#_Toc85244851)

[Как понимает, где метод в контроллере – операция для api, а где просто служебный: 4](#_Toc85244852)

[Пример 4](#_Toc85244853)

[Убрать форматтер 5](#_Toc85244854)

[Core в Configuration файла Startup 5](#_Toc85244855)

[Фреймфорк в Register файла WebApiConfig 6](#_Toc85244856)

[Авторизация через куки 6](#_Toc85244857)

[Пример авторизации для Core 6](#_Toc85244858)

[Пример авторизации для MVC 7](#_Toc85244859)

[ПО для тестирования api: 7](#_Toc85244860)

# Что такое:

Web API представляет собой веб-службу, которая может взаимодействовать с различными приложениями. При этом приложение может быть веб-приложением ASP.NET, либо может быть мобильным или обычным десктопным приложением.

Web API представляет способ построения приложения ASP.NET, который специально заточен для работы в стиле REST (Representation State Transfer или "передача состояния представления"). REST-архитектура предполагает применение следующих методов или типов запросов HTTP для взаимодействия с сервером:

GET

POST

PUT

DELETE

## Коротко: разница web api MVC и Core:

Core - кросс платформа, переработанная версия MVC, все еще развивается

Различается файловая структура (Core – разные конфигурации приложения, MVC – web.config)

MVC – состоит из библиотек, Core – nuget и т.д.

# Как работать с web api:

Web api на финальном этапе представляет из себя консольное приложение, которое равёртывается на сервере и находится в запущенном состоянии все время. Мы обращаемся к нему через http json запросами (xml тоже можно, но это уже ближе к soap сервису) по адресу в виде:

[протокол]://[ip хоста/localhost для локалки]:[порт, по умолчанию - 5000]/[контроллер]/[метод]

[протокол]://[домен сервера]/[контроллер]/[метод]

[[протокол]://[домен]/[контроллер]/[метод]/[параметр](http://[домен]/%5bконтроллер%5d/%5bметод%5d/%5bпараметр)]

[[протокол]://[домен]/[контроллер]/[метод]?param=[параметр](http://[домен]/%5bконтроллер%5d/%5bметод%5d?param=%5bпараметр)]

[протокол]://[домен]/[контроллер]/[метод]?param1=[параметр1]&param2=[параметр2]

Исключительно для get запроса без параметров

[протокол]://[ip хоста/localhost для локалки]:[порт, по умолчанию - 5000]/[контроллер]

[протокол]://[домен сервера]/[контроллер]

### Пример

<https://metanit.com/sharp/aspnet_webapi/1.1.php>

<https://vk.com/im?peers=c72_177132299>

- и в ответ он возвращает нам результат в сообщение в виде шапки с метаинформацией и кодом и тела (опционально).

HTTP ответ

https://ru.wikipedia.org/wiki/HTTP

Статус коды (знать основные)

<https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%BA%D0%BE%D0%B4%D0%BE%D0%B2_%D1%81%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D1%8F_HTTP>

# Тип ответа:

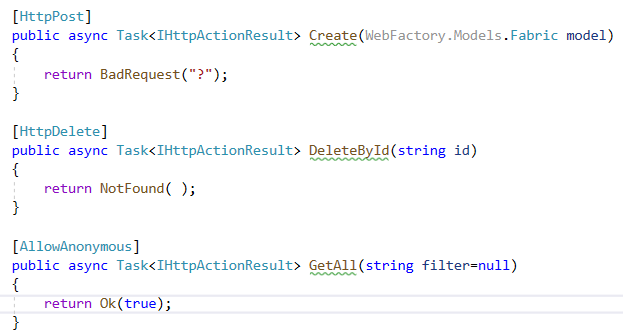
Когда метод возвращает какой-то объект, то код по умолчанию 200

Можно устанавливать код самостоятельно, если метод возвращает IHttpActionResult(MVC)/IActionResult(Core)

BadRequest – 500, Ok – 200 и т.д.

У каждого есть свои переопределения, т.к. могут передавать в теле объекты

Могут быть различия по наличию переопределений в Core/MVC (см справку!)



# Маршрутизация или как понимает, что вернуть:

В адресной ссылке обязательно прописывается контроллер.

При получении запроса система маршрутизации выбирает для обработки запроса нужный контроллер и передает ему данные запроса. Контроллер обрабатывает эти данные и посылает обратно результат обработки.

У маршрутизации есть атрибуты:

Название контроллера строится по принципу [Название]Controller, далее при маршрутизации слово контроллер убирается автоматически, если название не следует правилу, то придется вводить полное название.

### Пример

WeatherForecastController

http://localhost:1547/weatherforecast

WeatherForecastController1

<http://localhost:1547/weatherforecastcontroller1>

Но с помощью атрибута Route можно задавать конкретное название. Тоже самое с методами.

Если в контроллере только по 1 методу на каждую rest операцию и название метода совпадает с операцией (на get запрос – название метода Get), то он будет переходить на нее автоматически, иначе – каждому методу прописывается свой путь (Route) и вид операции (HttpGet/HttpPut/HttpPost/HttpDelete) в атрибуте.

## Как понимает, где просто класс программы, а где контроллер:

Контроллер api наследует класс ApiController – в MVC, ControllerBase – в Core, и обязательно имеет модификатор public.

## Как понимает, где метод в контроллере – операция для api, а где просто служебный:

По умолчанию, все public методы в контроллере считаются за get операцию.

Чтобы скрыть метод из api надо: а) убрать public, б) поставить атрибут NonAction

### Пример

По операциям post, put, delete выполнятся одноименные методы.

У путей могут быть префиксы, которые настраиваются в шаблоне пути. (В Core их атрибутов нет, файл – Startup, в MVC – аттрибут RoutePrefix и в файле WebApiConfig, можно задавать его прямо в Route)

По get запросам будет проходить маршрутизация:

Get без параметра откроется по дефолтному пути префикс/контроллер, т.к. название соответствует операции

Get1 и Get2 будут зависеть от наличия в запросе параметра, т.к. их пути одинаковы

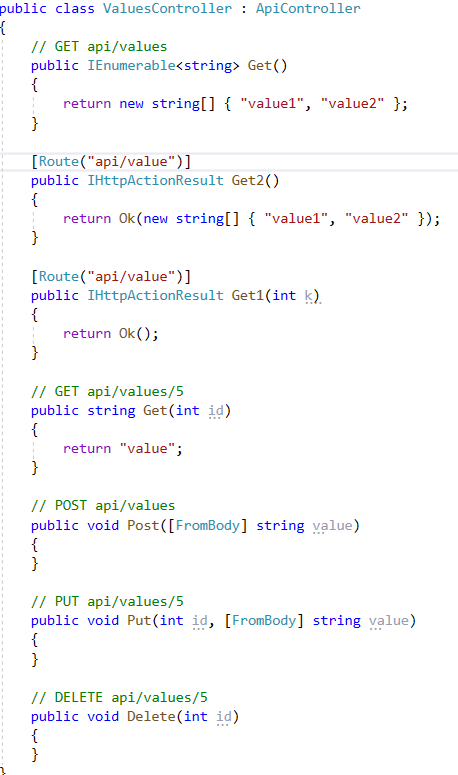
Get с параметром откроется только по пути префикс/контроллер/параметр, т.к. он обязателен

\*при обращении к методу с параметром в виде [[протокол]://[домен]/[контроллер]/[метод]?param=[параметр](http://[домен]/%5bконтроллер%5d/%5bметод%5d?param=%5bпараметр)] название параметра обязательно прописывается как в методе, т.е. для get1 - value?k=4, get - values?id=7

\*\*привязка параметров с атрибутами

FromUri – берется из ссылки, по умолчания

FromBody – из тела



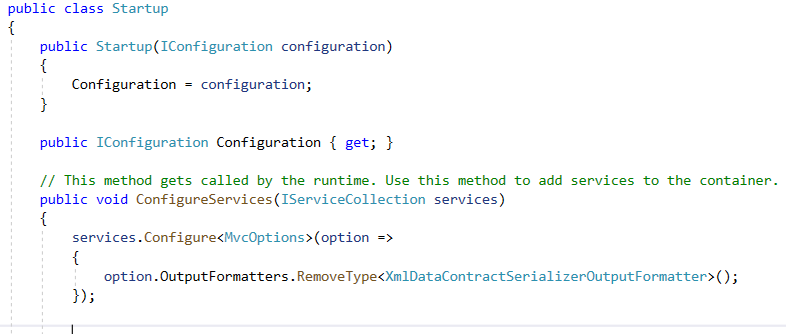
# Убрать форматтер

### Core в Configuration файла Startup

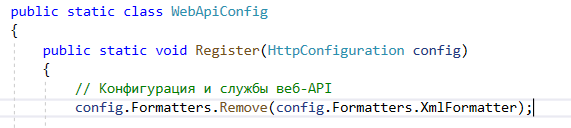
Требуется ссылка

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Formatters;



### Фреймфорк в Register файла WebApiConfig



# Авторизация через куки

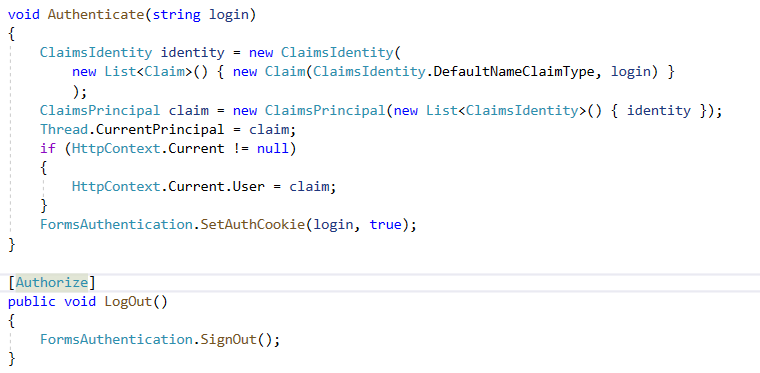
Можно ограничивать доступ к контроллерам/методам с помощью авторизации.

Атрибут Authorize – выполнится только для авторизованного, AllowAnonymous – для любого

### Пример авторизации для Core



### Пример авторизации для MVC



# ПО для тестирования api:

Postman – используется на вср

Fiddler

\*в MVC встроено описание api

Аналоги для Core – swagger, openapi