

[КАК СТАТЬ АВТОРОМ](#)[Неделя мобильной разработки](#)[\[Подкаст\] Мониторим...](#)

dkshibekov 3 февраля в 23:02

# AutoMapper: добавление и использование в проекте ASP.Net Core

.NET\*, ASP\*, C#\*

[Из песочницы](#)[Tutorial](#)

При работе с данными (и не только) нам часто приходится сталкиваться с необходимостью копирования (маппинга) значений свойств одного объекта в новый объект другого типа.

Например предположим, что запрос к базе данных возвращает нам запись в виде объекта, представленного классом `Person`:

```
public class Person
{
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public DateTime BirthDate { get; set; }
}
```

Далее нам необходимо создать новый объект, представленный классом `Student`:

```
public class Student
{
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public DateTime BirthDate { get; set; }
    public DateTime AdmissionDate { get; set; }
}
```

и скопировать в его свойства данные из свойств полученного из БД объекта.

Без помощи сторонней библиотеки нам пришлось бы сделать это самим:



0



12K



36



```
// получаем запись из БД
var person = _dbRepository.GetPerson(1);

// копируем значения свойств (осуществляем маппинг)
var student = new Student
{
    FirstName = person.FirstName,
    LastName = person.LastName,
    BirthDate = person.BirthDate
};
```

А с использованием библиотеки AutoMapper, маппинг производится всего одной строкой кода:

```
var student = _mapper.Map<Student>(person);
```

## 1. Подключение библиотеки AutoMapper к проекту и ее использование

### Шаг 1

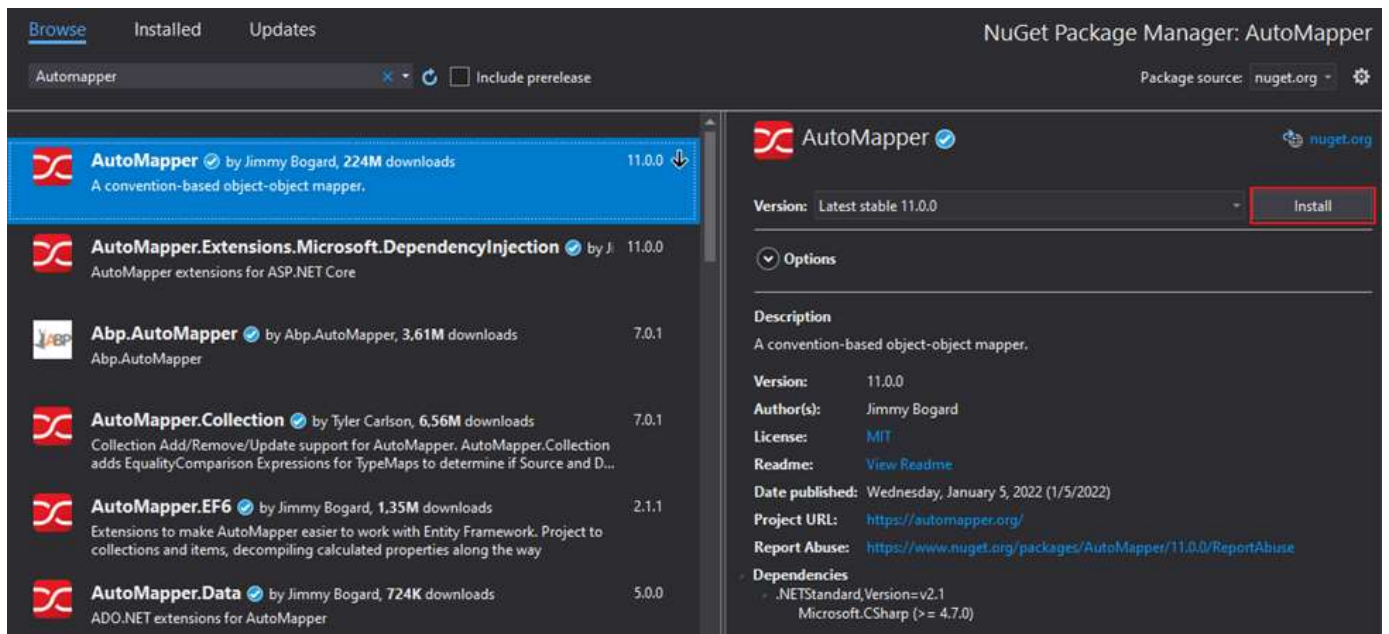
Добавление в проект NuGet пакетов:

- AutoMapper
- AutoMapper.Extensions.Microsoft.DependencyInjection

Для этого в SolutionExplorer (Обозреватель решений) правой кнопкой мыши ждем по названию рабочего проекта и выбираем *Manage NuGet Packages...* (*Управление пакетами Nuget*).

Далее переходим на крайнюю левую вкладку *Browse*, и в строку поиска вводим название устанавливаемого пакета NuGet.

В левом окне выбираем нужный нам пакет, а в правом ждем кнопку *Install*.



Дожидаемся окончания установки.

Продельываем эти действия для обоих пакетов.

## Шаг 2

Добавляем в проект класс AppMappingProfile:

```
public class AppMappingProfile : Profile
{
    public AppMappingProfile()
    {
        CreateMap<Person, Student>();
    }
}
```

В generics метода CreateMap первым передаем тип-источник значений, вторым — тип-приемник.

Т.е. в данном примере мы задаем маппинг из объекта Person в объект Student.

Если мы хотим, чтобы маппинг работал в обоих направлениях, добавляем вызов метода-расширения ReverseMap():

```
CreateMap<Person, Student>().ReverseMap();
```

Теперь мы можем также маппить объект `student` в объект `person`:

```
var person = _mapper.Map<Person>(student);
```

### Шаг 3

Добавляем AutoMapper в DI контейнер. Для этого в метод `ConfigureServices` класса `Startup.cs` добавляем строку:

```
public void ConfigureServices(IServiceCollection services)
{
    // другой код
    services.AddAutoMapper(typeof(AppMappingProfile));
    // другой код
}
```

Классов маппинг-профайлов может быть создано несколько. В таком случае, передаем их в параметры метода `AddAutoMapper` через запятую:

```
services.AddAutoMapper(typeof(AppMappingProfile), typeof(MappingProfile2));
```

### Шаг 4

Используем маппер.

Теперь маппинг нашего объекта `Person` в `Student` происходит в одну строку кода:

```
var student = _mapper.Map<Student>(person);
```

Ниже приведен полный код класса, использующего маппинг:

```
using AutoMapper;
using AutoMapperInAspNetCore.Db;
using AutoMapperInAspNetCore.Models;
```

```
namespace AutoMapperInAspNetCore.Mapping
{
    public class MappingHelper
    {
        private readonly IMapper _mapper;
        private readonly IRepository _dbRepository;

        public MappingHelper(IMapper mapper, IRepository dbRepository)
        {
            _mapper = mapper;
            _dbRepository = dbRepository;
        }

        public void DoSomething()
        {
            // получаем запись из БД
            var person = _dbRepository.GetPerson(1);

            // Создаем новый объект типа Student и копируем в его свойства
            // значения свойств объекта person (осуществляем маппинг)
            var student = _mapper.Map<Student>(person);
        }
    }
}
```

Здесь в качестве generic в метод Map объекта \_mapper передаем тип-приемник (Student), а в параметре передаем объект-источник (person).

Теперь мы имеем объект student типа Student, со значениями полей из объекта person.

При этом мапятся только те поля, названия которых полностью совпадают у обоих типов.

В данном случае – это поля:

```
public string FirstName { get; set; }
public string LastName { get; set; }
public DateTime BirthDate { get; set; }
```

## 2. Маппинг объектов с не совпадающими наименованиями свойств

Для наглядности немного видоизменим класс Student, переименовав его свойства:

```
public class Student
{
    public string Fio { get; set; }
    public DateTime Birthday { get; set; }
    public DateTime AdmissionDate { get; set; }
}
```

Теперь наименования свойств объектов person и student не совпадают.

Для того, чтобы маппинг заработал, нам придется дописать в маппинг-профайле явные правила:

```
public AppMappingProfile()
{
    CreateMap<Person, Student>()
        .ForMember(dest => dest.Fio, opt => opt.MapFrom(src => $"{src.FirstName} {src.LastName}")
        .ForMember(dest => dest.Birthday, opt => opt.MapFrom(src => src.BirthDate));
}
```

где параметр dest представляет собой объект-приемник, а src – объект-источник.

В методе MapFrom для поля Fio объекта student мы применили интерполяцию строки –

```
"${src.FirstName} {src.LastName}"
```

результатом которой будет строка вида «Имя Отчество», которая и присвоится свойству Fio объекта student.

PS: Данный tutorial рассчитан в первую очередь на уже знакомую с предназначением описываемой технологии аудиторию и представляет из себя краткое руководство по быстрому применению базового функционала этой технологии в своих проектах. Tutorial не претендует на полное и исчерпывающее пособие по затронутой тематике и не является рекламой. Идея написания этой, и серии подобных статей, была подсказана студентами, не владеющими в должной мере английским языком и испытывающими определенные трудности при поиске такого рода информации на просторах рунета.


**Теги:** automapper, asp.net core

Хабы: .NET, ASP, C#

### Редакторский дайджест

Присылаем лучшие статьи раз в месяц

Электронпочта



4

Карма

0

Рейтинг

@dkshibekov

Пользователь

Комментарии 36

#### ПОХОЖИЕ ПУБЛИКАЦИИ

- 9 августа в 16:16

Добавляем поддержку Markdown в ASP.NET Core приложение. Часть 1

+3

1.5K

22

0
- 24 июля в 18:51

Создание и запуск веб-приложения в ASP.NET Core

+4

4.6K

55

4 +4
- 24 июня в 12:16

Особенности применения LRU кэша в ASP NET Core приложениях

+9

2.9K

32

0

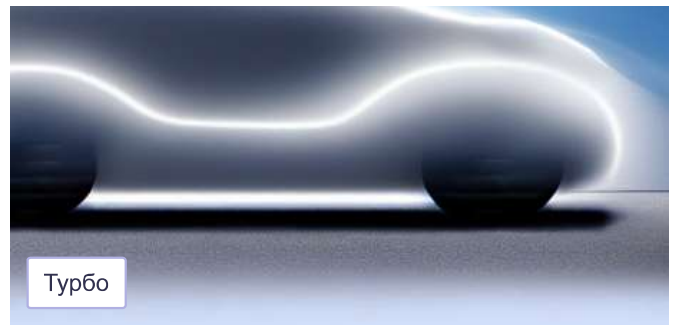
МИНУТОЧКУ ВНИМАНИЯ

Разместить





Как приготовить лучший кейс: пробуем статьи на вкус

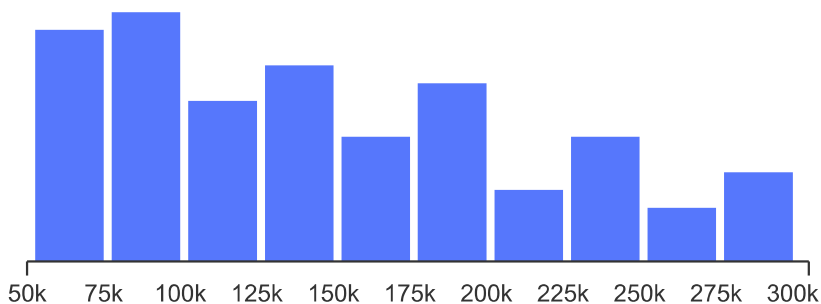


Переезжаем из одного облака в другое

## СРЕДНЯЯ ЗАРПЛАТА В IT

**158 393** ₽/мес.

— средняя зарплата во всех IT-специализациях по данным из 4 838 анкет, за 2-ое пол. 2022 года. Проверьте «в рынке» ли ваша зарплата или нет!



Проверить свою зарплату

## ЛУЧШИЕ ПУБЛИКАЦИИ ЗА СУТКИ

вчера в 16:29

### Осторожно, следующая остановка столбняк

◆ +37

👁 8.7K

📖 33

💬 37 +37

вчера в 13:00

### Российские компьютерные игры 90-х годов. Часть 3: рождение «русского квеста»

◆ +33

👁 5K

📖 32

💬 13 +13

вчера в 13:12

### Краткая (очень) история боёв роботов



 +31 3.6K 14 17 +17

вчера в 10:08

## Импортозамещение без глянца. Том 2 — операционные системы

 +18 9.5K 17 28 +28

вчера в 18:49

## Образование в РФ избыточно

 +14 12K 41 124 +124

## Кто и зачем поставил Open Source в мониторинг

Подкаст 

### ЧИТАЮТ СЕЙЧАС

## Блеск и нищета IT в Германии

 69K  330 +330

## Образование в РФ избыточно

 12K  124 +124

## Почему я не хочу продолжать работу в биотехе

 2.6K  10 +10

## Mikrotik, Telegram и не только...

 697  0

## Осторожно, следующая остановка столбняк

 8.7K  37 +37

## DAST ist fantastisch: отечественный динамический анализатор к взлету готов

Меропост

РАБОТА

.NET разработчик  
71 вакансии

Программист C# удаленно  
94 вакансии

Все вакансии

Ваш аккаунт	Разделы	Информация	Услуги
Войти	Публикации	Устройство сайта	Корпоративный блог
Регистрация	Новости	Для авторов	Медийная реклама
	Хабы	Для компаний	Нативные проекты
	Компании	Документы	Образовательные программы
	Авторы	Соглашение	Стартапам
	Песочница	Конфиденциальность	Мегапроекты



Настройка языка

Техническая поддержка

Вернуться на старую версию