

# Защо версиите на .NET са важни!

**Представете си, че правите приложение, в което създавате клас Person.**

Съобразно вашите програмни умения създавате методи, които да описват неговите възможните действия – ОК.

Направили сте го да ходи, да яде, да бяга и т.н. Но **не можете** да го накарате да пътува с автомобил, защото автомобила може да е твърде сложен за вашите умения и няма как да го пресъздадете.

Точно това правят допълнителните външни кодове като цяло – дават ви готова функционалност! В зависимост от типа на външния код биват най-различни, но понякога имат и доста общи неща. Има галерия от външни кодове, които можем да изтеглим през интернет през **вградена функционалност във VisualStudio, която се нарича NuGet**. След като си изтеглим и инсталираме външните кодове, за да добавим такъв в проектът си просто използваме using и неймспейса който е специфичен за добавения код в началото на даден клас.

**Базовият фреймуорк обаче (.NET версията) трябва да я имаме инсталирана предварително на компютърът си, т.е. не се инсталира през NuGet, а или от интернет или от VS меню Tools/Get Tools and Features.**

*\* Може да имаме инсталирани много различни версии едновременно на самия компютър, просто когато създаваме нов проект избираме коя версия от наличните искаме да използваме!*

Защо са всички тези различни версии... Ами просто е.

Ако фреймуорк 1.0 ви е снабдил с колело, то фреймуорк 1.1 ви дава по-модерно колело, 1.2 тротинетка (не електрическа). Във версия 2.0 се добавят и каруци и файтони, 5.0 влакове, 6.0 коли и т.н. Вмъквам и тук таблицата с различните версии

.NET Версия	Номера на версии	Операционни системи	Описание
.NET Framework	1.0, 1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.6, 4.7, 4.8	Windows	Работи само на Windows. Използва се основно за десктоп и уеб приложения с Windows Forms и ASP.NET.
.NET Core	1.0, 1.1, 2.0, 2.1, 3.0, 3.1	Windows, macOS, Linux	Мултиплатформена версия, подходяща за различни ОС, модернизирана платформа.
.NET (унифицирана)	5.0, 6.0, 7.0, 8.0	Windows, macOS, Linux	Новата обединена платформа, която обединява .NET Framework и .NET Core. Поддържа и облачни среди като Azure.

Специално .NET Framework версиите са ориентирани тясно към Уиндоус среда. Докато .NET Core и .NET са мултиплатформени – Win, MacOS, Linux.

Ако предположим че Windows е сухоземната среда, Linux е морската среда, а MacOS – въздушната среда, то досещате се, .NET Framework осигурява само сухоземни транспортни средства, докато в .NET Core и .NET са включени освен част от сухоземните, то и други превозни средства за вода и въздух. За да сме по точни има и фреймуорци и за телефони, за тях като най-нови може да асоциираме превозни средства за космоса.

Тук е мястото да отбележим, че има и Standart версии, където се включват части от трите различни фреймуорка, като идеята е хем да се поддържат различните видове, хем да не се поддържат по-старите за да не се претрупва целия пакет, който ги съдържа (а по-честата причина е наличието на уязвимости в имплементацията на самите версии). Има и някакви разлики в компилацията на крайния ви код, но това са подробности. Който иска може да чете повече на следния адрес:

[.NET Standard - .NET | Microsoft Learn](https://learn.microsoft.com/en-us/dotnet/standard/net-standard?tabs=net-standard-1-3)

<https://learn.microsoft.com/en-us/dotnet/standard/net-standard?tabs=net-standard-1-3>

**Всеки един ПРОЕКТ може да „стъпи” на различен Фреймуорк!**

Всеки Проект който добавяте в Солюшъна си може да стъпва на различен фреймуорк, но когато единия проект е тестващ, а другия – тестван, то най-добре да са в един и същи фреймуорк (като рамка от трите вида) и тестващия проект да не е по-стара версия от базовия проект!

**Проблем може да се получи, когато тестов проект се опитва да направи нещо за което не разполага с необходимата функционалност, т.е. фреймуорка на който стъпва е по-слаб от фреймуорка на основния проект!**

**(\* Но не е изключено и обратното)**

Примерно искам да тествам как Джони кара електрическа тротинетка, само че фреймуорка, който я описва е добавен в основния проект от версия 8.0, но не и в тестващия 6.0 (от времето когато не е имало такива тротинетки). Няма как тестващия да я пресъздаде за да повтори събитието!

**Обратното (обратна несъвместимост) – рядко, но не невъзможно**

Както споменах малко по-горе е заради премахването на уязвими кодове или твърде остарели кодове от по-ниската версия.

Примерно искам да тествам как Джони използва копие, но фреймуорка който описва що това копие е във версия 1.0 (и в базовия проект си работи нормално), а в тестващия проект, който примерно е версия 8.0 копията са премахнати защото вече никой не ги използва.

Когато създаваме тестващ проект предполагам е добре да избираме версия от същия тип какъвто е базовия(тествания) проект. И не би трябвало да е проблем ако версията на тестващия е по-висока, понеже той би разполагал с всички функционалности, така че да може да пресъздаде работата на базовия проект.



Да, когато създаваш тестващ проект, е добра практика да избираш същия тип версия на .NET, какъвто е използвана за базовия (тествания) проект. Това гарантира, че средата, в която се изпълнява тестът, съответства на тази, в която работи самият проект.

Обаче, ако тестващият проект използва по-висока версия на .NET, обикновено не би трябвало да има проблем. По-високите версии на .NET обикновено са съвместими назад и включват всички функционалности на предишните версии. Така тестващият проект ще може да симулира правилно поведението на базовия проект и да използва нови инструменти за тестове, ако е необходимо.

Има само няколко случая, в които може да възникнат проблеми:

1. **Обратна несъвместимост:** В редки случаи някои функционалности може да бъдат променени или премахнати в по-новите версии на .NET, което може да доведе до проблеми, ако базовият проект зависи от тях.
2. **Миграция на проекта:** Ако тестващият проект използва много по-висока версия на .NET, може да се наложи в бъдеще да мигрираш базовия проект към същата или по-висока версия, за да поддържаш съвместимост между тях.

Обикновено, ако базовият проект е на стабилна версия и не планираш миграция в близко бъдеще, е най-сигурно да използваш същата версия и за тестващия проект.

### Как да проверим на коя версия е писан пакет, който не сме създали ние?

През **Solution Explorer** маркираме пакета, после от менюто на студиото използваме **Project/Име на проекта Properties**. Може и директно с десен бутон върху проекта. Като и двете места тези опции се намират най-долу измежду тези които се показват.

