What is the best data type to use for money in C#?

Ask Question

381

39

What is the best data type to use for money in C#?

c# currency

edited Feb 1 at 7:02
Cœur
19.4k 10 116 155

asked Mar 28 '09 at 19:30
NotDan
16.2k 33 105 151

4 You might find answers from this post helpful. – ntombela Mar 28 '09 at 19:41

Here's a mapping for all data types: docs.microsoft.com/en-us/dotnet/framework/data/adonet/… – JohnLBevan Feb 1 '18 at 13:14

Also, if using data annotations, include `using System.ComponentModel.DataAnnotations;` … `[DataType(DataType.Currency)]` msdn.microsoft.com/en-us/library/… – JohnLBevan Feb 1 '18 at 14:14

9 Answers

367

As it is described at decimal as:

> The decimal keyword indicates a 128-bit data type. Compared to floating-point types, the decimal type has more precision and a smaller range, which makes it **appropriate for financial and monetary** calculations.

You can use a decimal as follows:

```
decimal myMoney = 300.5m;
```

edited May 14 '17 at 15:26
Yves
1,652 2 17 38

answered Mar 28 '09 at 19:32
Lee Treveil
4,608 4 25 28

37 You should explain what about that link is important. An answer should be good enough on its own, with a link as additional reference or detail. See stackoverflow.com/help/how-to-answer – TheRubberDuck Apr 13 '15 at 15:32

2 So the minimum-length answer can be fewer characters than the minimum-length comment - interesting! Not that I have a problem with the terse/concise answer, especially when it is also "deep" in that it links to further discussion. – B. Clay Shannon May 1 '15 at 20:02

3 Amazing answer, and I don't feel it needs further explanation since it completely answers the question. The link to MSDN documentation is a bonus as far as I'm concerned. Bravo! – trnelson May 11 '16 at 18:33

@Leee Treveil , what is the money is like (9.0098) means 4 character after Point – SAR Dec 31 '16 at 5:10

5

Most applications I've worked with use `decimal` to represent money. This is based on the assumption that the application will never be concerned with more than one currency.

This assumption may be based on another assumption, that the application will never be used in other countries with different currencies. I've seen cases where that proved to be false.

Now that assumption is being challenged in a new way: New currencies such as Bitcoin are becoming more common, and they aren't specific to any country. It's not unrealistic that an application used in just one country may still need to support multiple currencies.

Some people will say that creating or even using a type just for money is "gold plating," or adding extra complexity beyond the known requirements. I strongly disagree. The more ubiquitous a concept is within your domain, the more important it is to make a reasonable effort to use the correct abstraction up front. If you want to see complexity, try working in an application that used to use `decimal` and now there's an additional `Currency` property next to every `decimal` property.

If you use the wrong abstraction up front, replacing it later will be a hundred times more work. That means potentially introducing defects into existing code, and the best part is that those defects will likely involve amounts of money, transactions with money, or just anything with money.

And it's not that difficult to use something other than decimal. Google "nuget money type" and you'll see that numerous developers have created such abstractions (including me.) It's easy. It's as easy as using `DateTime` instead of storing a date in a `string` .

answered Feb 8 '18 at 1:18

Scott Hannen
13.7k 15 26

Home

PUBLIC

Stack Overflow

Tags

Users

Jobs

Teams
Q&A for work

Learn More

110

System.Decimal

> The Decimal value type represents decimal numbers ranging from positive 79,228,162,514,264,337,593,543,950,335 to negative 79,228,162,514,264,337,593,543,950,335. The Decimal value type is appropriate for financial calculations requiring large numbers of significant integral and fractional digits and no round-off errors. The Decimal type does not eliminate the need for rounding. Rather, it minimizes errors due to rounding.

I'd like to point to this excellent answer by zneak on why double shouldn't be used.

edited May 23 '17 at 12:10
Community ♦
1 1

answered Mar 28 '09 at 19:32
David Walschots
8,333 5 27 48

62

Use the Money pattern from Patterns of Enterprise Application Architecture; specify amount as decimal and the currency as an enum.

edited Mar 12 '13 at 10:42
Gan
3,507 2 27 43

answered Mar 28 '09 at 19:41
lmsasu
3,857 14 68 109

1 I was actually going to suggest this, but I make Currency a class so I can

define an exchange rate (in relation to a "base currency", often the US dollar [which I set to have an exchange rate of 1.00]). – Thomas Owens Mar 28 '09 at 20:25

15 codeproject.com/KB/recipes/MoneyTypeForCLR.aspx – Chris Haines Jan 19 '10 at 9:46

5 For the future visitors of this thread (like me), there is now this: nuget.org/packages/Money and it rocks! – Korijn Nov 4 '14 at 12:07

Wondering if such a type should be a struct or class. A decimal + an (int) enum makes it 20 bytes. My money is on struct still. – nawfal Dec 9 '16 at 10:23

That `Money` nuget has a dead github link for project site so...no docs? – George Mauer Apr 2 '17 at 17:09

8

Another option (especially if you're rolling you own class) is to use an int or a int64, and designate the lower four digits (or possibly even 2) as "right of the decimal point". So "on the edges" you'll need some "* 10000" on the way in and some "/ 10000" on the way out. This is the storage mechanism used by Microsoft's SQL Server, see http://msdn.microsoft.com/en-au/library/ms179882.aspx

The nicity of this is that all your summation can be done using (fast) integer arithmetic.

answered Feb 16 '13 at 22:21

dsz
2,638 22 25

13

Agree with the Money pattern: Handling currencies is just too cumbersome when you use decimals.

If you create a Currency-class, you can then put all the logic relating to money there, including a correct ToString()-method, more control of parsing values and better control of divisions.

Also, with a Currency class, there is no chance of unintentionally mixing money up with other data.

answered Mar 28 '09 at 20:07

Lennaert
2,325 13 15

4

Create your own class. This seems odd, but a .Net type is inadequate to cover different currencies.

answered Mar 28 '09 at 20:03

Noel Kennedy
9,179 3 34 54

15

decimal has a smaller range, but greater precision - so you don't lose all those pennies over time!

Full details here:

http://msdn.microsoft.com/en-us/library/364x0z75.aspx

answered Mar 28 '09 at 19:34

dommer
16.1k 7 58 112

5 Who wants to keep their pennies anyways? :) – JeremyK Jan 30 '14 at 19:43

Someone should skim off those fractions of a penny and deposit them into a bank account. That's never been done before right? – hardba11 Jan 9 '17 at 19:05

@hardba11 It was, and he got rich like millionaire, but then got arrest. Because actually those fractions is for the bank themselves, it part of the bonus the bank got – Thaina Jan 10 '17 at 4:35

20

Decimal. If you choose double you're leaving yourself open to rounding errors

answered Mar 28 '09 at 19:33
SquidScareMe
2,528 2 19 35

How do you not with Decimal? – Beep beep Feb 3 '10 at 19:45

6 @Jess `double` can introduce rounding errors because floating point cannot represent all numbers exactly (e.g. 0.01 has no exact representation in floating point). `Decimal` , on the other hand, *does* represent numbers *exactly*. (The trade-off is `Decimal` has a smaller range than floating point) Floating point can give you * inadvertent* rounding errors (e.g. `0.01+0.01 != 0.02`). `Decimal` can give you rounding errors, but only when you asked for it (e.g. `Math.Round(0.01+0.02)` returns zero) – Ian Boyd Dec 15 '11 at 20:42

2 @IanBoyd: The value "$1.57" can be precisely represented (double)157. If one uses `double` and carefully applies scaling and domain-specific rounding when appropriate, it can be perfectly precise. If one is sloppy in one's rounding, `decimal` may yield results which are semantically incorrect (e.g. if one adds together multiple values which are supposed to be rounded to the nearest penny, but doesn't actually around them first). The only good thing about `decimal` is that scaling is built-in. – supercat Jun 9 '12 at 23:55

1 @supercat, regarding this comment "if one adds together multiple values which are supposed to be rounded to the nearest penny, but doesn't actually around them first", i do not see how a float would solve this. It is a user error and has nothing to do with decimals IMHO. i do get the point but i feel it has been misplaced, mainly because IanBoyd did specify that ...if you ask for it. – sawe Jul 22 '13 at 9:03

protected by Brad Larson ♦ Nov 22 '14 at 21:08

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now