Enum struct? A Value object that behaves like a Enum

Asked 16 years ago Modified 16 years ago Viewed 4k times

4

I am wondering how you would approach this problem

I have two Taxrates that can apply to my products. I specifically want to avoid persisting the Taxrates into the database while still being able to change them in a central place (like Taxrate from 20% to 19% etc).

so I decided it would be great to have them just compiled into my application (It's internal). The problem is that I want to not only to know the Rate but also the Name of the Taxrate.

I could go with an Enum that maps to the value. But then I'd have to create some method that retrieves the German Name of that Taxrate for the English enum-value (I write my code in english, the App is in german).

I thought about just using hardcoded objects to reflect this,

```
public interface Taxrate
{
    string Name { get; }
    decimal Rate { get; }
}

public class NormalTaxRate : Taxrate
{
    public string Name
    { get { return "Regelsteuersatz"; } }

    public decimal Rate
    { get { return 20m; } }
}
```

But then I'd have to create some sort of list that holds two instances of those two objects. Doing it static may work, but still I'd have to keep some sort of list. Also I'd have to find a way to map my POCO Domain Object to this, because I doubt NHibernate can instantiate the right Object depending on a value in a field.

It doesn't really feel right, and I think I'm missing something here. Hope somebody has a better solution, I can't think of one.

greetings, Daniel

Ps: also please retag this question if you find something fitting, I can't think of more meaningful tags right now.

c# enums

Share Improve this question Follow

asked Nov 28, 2008 at 11:12

Tigraine
23.6k ● 12 ● 72 ● 113

5 Answers

Sorted by: Highest score (default)

6

EDIT: Note that the code here could easily be abbreviated by having a private constructor taking the tax rate and the name. I'm assuming that in real life there might be actual behavioral differences between the tax rates.

It sounds like you want something like Java's enums.

C# makes that fairly tricky, but you can do it to some extent using private constructors and nested classes:

```
public abstract class TaxRate
{
    public static readonly TaxRate Normal = new NormalTaxRate();
    public static readonly TaxRate Whatever = new OtherTaxRate();

    // Only allow nested classes to derive from this - and we trust those!
    private TaxRate() {}

    public abstract string Name { get; }
    public abstract decimal Rate { get; }

    private class NormalTaxRate : TaxRate
    {
        public override string Name { get { return "Regelsteuersatz"; } }
        public override decimal Rate { get { return 20m; } }
    }

    private class OtherTaxRate : TaxRate
    {
        public override string Name { get { return "Something else"; } }
        public override decimal Rate { get { return 120m; } }
    }
}
```

You'd probably want some sort of static method in TaxRate to return the right instance based on name or whatever.

I don't know how easily this fits in with NHibernate, but hopefully it will help to some extent...

As noted in the comments, it's pretty ugly - or at least can get pretty ugly when you've got lots of different values. Partial classes can help here:

```
// TaxRate.cs
public partial abstract class TaxRate
{
    // All the stuff apart from the nested classes
}

// TaxRate.Normal.cs
public partial abstract class TaxRate
{
    private class NormalTaxRate : TaxRate
    {
        public override string Name { get { return "Regelsteuersatz"; } }
        public override decimal Rate { get { return 20m; } }
    }
}

// TaxRate.Other.cs
public partial abstract class TaxRate
{
    private class OtherTaxRate : TaxRate
    {
        public override string Name { get { return "Something else"; } }
        public override decimal Rate { get { return 120m; } }
    }
}
```

You can then munge the project file to show the nested classes as children of the outer class, as shown in this SO question.

Share

Improve this answer

Follow

edited May 23, 2017 at 12:24

Community Bot

1 • 1

answered Nov 28, 2008 at 11:25

Jon Skeet

1.5m • 889 • 9.3k • 9.3k

As ugly as that looks, Skeeter is right... that's not a bad solution :P I'd also override the default conversion process so that you can convert from a decimal to one of these instances. – Timothy Khouri Nov 28, 2008 at 11:29

BTW: I only think it's ugly because it's smooshed together. – Timothy Khouri Nov 28, 2008 at 11:30

Edited to note the possibility of putting this in partial classes. – Jon Skeet Nov 28, 2008 at 11:35

I'd do it like this:

3

```
public class TaxRate
{
    public readonly string Name;
    public readonly decimal Rate;

    private TaxRate(string name, decimal rate)
    {
        this.Name = name;
        this.Rate = rate;
    }


    public static readonly TaxRate NormalRate = new TaxRate("Normal rate", 20);
    public static readonly TaxRate HighRate = new TaxRate("High rate", 80);
}
```

This way it would be easy to use it - just access `TaxRate` static members like enum values. To use it with NHibernate you will have to create your own custom NHibernate type class (see documentation for it), but it's not so hard. I've done it once already.

Share Improve this answer Follow

answered Nov 28, 2008 at 12:00

Vilx-

107k ● 90 ● 288 ● 430

1

Why not store the tax rates in application configuration, eg in the web.config or app.config file? These are simple XML files, which have a section called where you can specify custom parameters with a key and a value. For example:

```
<appSettings>
    <add key="BaseTaxRate" value=20"/>
    <add key="HigherTaxRate" value=40"/>
</appSettings>
```

You can then retrieve these in your application simply:

```
string baseTaxRate = ConfigurationSettings.AppSettings["BaseTaxRate"];
```

This way, they are easily changed, and won't require re-compilation and re-deployment of your application.

Obviously, you can also store the names of the tax rates as extra settings in the config file.

Share

Improve this answer

Follow

edited Nov 28, 2008 at 11:29

answered Nov 28, 2008 at 11:22

Winston Smith

21.9k ● 10 ● 62 ● 75

I must say I find the idea of using nested classes somewhat bizarre for what seems to me to be a mainstream requirement.

What's wrong with storing the rates in the database (or other persistence medium such as the application configuration file)? I'd have thought you'd want to have a unique Id for each tax rate, used for the relation between products and tax rates.

So you would have a TaxRate class with Id/Description/Rate (*). You could load a dictionary with all possible values for quick lookup of rate/description by Id.

(*) in a multilingual app you'd need to look up a localized description for each pair Culture/Id - either from a second table in the database, or from resources etc...

In any case, hardwiring rates seems wrong - especially now where governments are playing with tax rates to try to boost their economies.

Share Improve this answer Follow

answered Nov 28, 2008 at 14:33

to StackOverflow

125k ● 33 ● 210 ● 341

> Every product has a distinct taxrate applied to it. Every product needs to display that taxrate in every list. There are lists that contain 1000+ Products and need to display the Taxrate there. If I have to lookup the Taxrate from the DB the server gets much more load. – Tigraine Nov 29, 2008 at 16:46

I know it is against the main premise of your question, but I would like to give some thoughts to this issue.

While it at the moment may seem like a logical decision to avoid persisting VAT and tax rates in the database, time will prove you wrong. First, you should always be able to extract the data as it was at a given time, and to do that you need versioning of the rates.

As we know, the only constant is change. Say the rates changes (they will). If you have lost your access to the source, you can't accommodate the proper maintenance. If the rates were to be split from flat VAT to different rates, updating the system would be even more difficult.

This problem is not something though up. The United Kingdom has reduced VAT from 17.5% to 15% effective as of 1st December. Many people are stuck with old software without ability to update the rates. Please avoid doing this mistake (and splitting the rates into the database would give a lot of other improvements as well).

Share Improve this answer Follow

answered Nov 28, 2008 at 15:01

Troels Thomsen
11.6k ● 4 ● 30 ● 29