Enumerated types as constants across web services?

Asked 16 years, 2 months ago Modified 16 years, 2 months ago

Viewed 6k times

3

I'm working on a project where I'm trying to avoid hard-coding DB IDs in a .NET service-oriented project. There are some instances where I **need** to set ID values through code but I don't want to just hard code the IDs since I've done that before and it lead to DB alignment nightmares when the auto-incrementing IDs were changed when the DB was dumped to a new system.

What I want to do is create an enumerated constants that store the IDs as so that at the worst, only 1 file has to be updated if the DB is ever changed instead of trying to go through thousands upon thousands of lines of code to replace any ID in the system.

This will work on a single system, but in my company's service oriented environment, enumerations don't serialize with their values, just their names.

What is the best way to share IDs across a web service? I'd like to use either enumerations (the ideal situation) or constants in some way, but I can't seem to get this to work. I could make a web method that returns the IDs, but sending a web request for every ID and then

serializing the response and deserializing on the client machine just sounds like a bad idea.

EDIT
I wasn't entirely clear about what I was asking, so I'll elaborate.

I want to have a group of constants. The enum would only be used because it groups constants together appropriately. I'm mainly interested in see if there is a way to share constants across a web service. I need the values the enum represent, not the enum itself. The enum is never sent between the service and the client except as an integer. Internally everything is stored as an ID, not an enum.

Having a separate shared library doesn't sound like the ideal solution since I'm almost at the completion point for this project and I'd only be storing 1 enum/class in the library. It seems like a bit of a waste to make for just one class.

c# .net web-services

Share

Improve this question

Follow

edited Oct 2, 2008 at 1:36

asked Oct 2, 2008 at 0:06

Dan Herbert

103k ● 51 ● 192 ● 221

Is the reason for using enums/constants to prevent other code from changing the values? And, is the reason for sharing these IDs because more than one service accesses the same database? Can you give more information about your data tier? Is there not a central service for that? – hurst Oct 2, 2008 at 3:11

There is a central service for the DB, however there are situations where I need to set IDs for status, and I don't want to just hard code numbers because obviously that's bad. I just need a way to share constant integers across a web service. – Dan Herbert Oct 2, 2008 at 11:57

Have you tried or considered using a data contract? – hurst Oct 3, 2008 at 1:25

7 Answers

Sorted by: Highest score (default)

1

I've always created a separate assembly that contains the enumerations and any interfaces the client/server need to share. You can then reference it from both the client and the server without leaking any functionality.

Share Improve this answer Follow

answered Oct 2, 2008 at 1:18

user22467

1

Enums are inherently serializable as a native data type, so it should not be a problem sharing them across services. But **you have to use a shared data contract**. We use enums for small lookup lists associating tokens to

IDs in the database, but then we also share the data contract between services (we use WCF). This allows us to use the enum tokens to refer to the associated integer value in the code of any service. If the values in the database change, we will have to update the enum manually, but only in one place - the data contract.

Another possible solution is to create a cache in each service that needs the IDs. During startup of each service, have it fetch the values from the central data service and store it in an appropriate manner. This could be a custom cache object or maybe a static dictionary. When you experience the renumbering issue, just restart the services.

I work on a project where this is done for certain user entities where we need the actual IDs and want to avoid constantly calling the data service for something that doesn't change much if ever.

Share Improve this answer

Follow

answered Oct 2, 2008 at 12:40

hurst

1,740 ● 12 ● 8

0

You can define the enums in common library and use it on client as well as server side. When you pass enum through a web service - it gets converted to string. Write a simple conversion extension method that converts it to appropriate enum. example:

```
DayOfWeek ConvertToDayOfWeek(this String str)
{
   return (DayOfWeek)Enum.Parse(typeof(DayOfWeek), str
}
```

(Note: I'm assuming you have rich client/desktop app. consuming the web service)

Share Improve this answer

Follow

answered Oct 2, 2008 at 1:02

Vivek

16.5k ● 5 ● 32 ● 37

> That's not what I'm trying to do though. I'm trying to use an enumeration to store constant integers. The names don't matter (except for to be descriptive). It's the values they represent that matter. – Dan Herbert Oct 2, 2008 at 1:08

0

You can write enum on a shared assembly and use it in your applications.

But I think that serving enum on webservice should work. I have some c# webservices with enum and at client side (php) we pass enum value name and I get value normally at c#.

Share Improve this answer

Follow

answered Oct 2, 2008 at 1:21

Zote

5,379 ● 5 ● 43 ● 43

0

Just create a public enum in your webservice project, and create at least one public WebMethod that uses the enum as its return type. After updating the web reference in your client project, you can then use that enum in the client code as well.

This basically lets you define an enum on the server and use it on both the server and the client, without futzing around with a shared library.

Share Improve this answer

Follow

answered Oct 2, 2008 at 2:16

MusiGenesis

75.3k ● 41 ● 197 ● 338

> That doesn't work for me because the enum doesn't store the constants tied to it, only the names if you publish it on a web service. – Dan Herbert Oct 2, 2008 at 2:28

0

Is the code on the other side of the webservice c# as well or do you actually need to pass the integer across the web?

If you have control over the code on both sides you can just pass the enum and convert it back to an int once it has crossed the wire.

```
(Int32)objectThatWasPassed.EnumerationValue;
```

If you don't have access to the code on the other side and need to pass it as an int you can create an integer property on whatever you are passing and just call;

```
objectbeingPassed.ConstantProperty = (Int32)Whatever.C
```

Share Improve this answer

Follow

answered Oct 2, 2008 at 20:10

Shaun Bowe

9,998 ● 11 ● 52 ● 71

0

There are a few options for what you would like to do. You could define a set of constants defined within a single class that represents the IDs and can help you translate between the ID and something that is more useful to you. This is moderately flexible and if you want tot get really fancy you can even look up some of your magic ids from the database (as per hurst's suggestion). Pick the type you want to send the values and then just wrap/ignore the fact it is a readonly constant.

As mentioned previously you can use WCF to send enumerations across WCF but they are extremely brittle. Any time you change the enum values you will be forced to recompile the service and update your client references. To exposed enums in WCF add the [DataContract] attribute to the class and the [EnumMember] attribute to each member. You have been warned.

A few of the previous suggestions mentioned to use a shared set of values. This is highly recommended so that you only have to manage it an update it once. With such a scenario it is very very very (did I say very?) important to make sure that everyone uses the shared set of values and never the values directly.

Good luck.

Share Improve this answer

Follow

answered Oct 3, 2008 at 2:27

smaclell

4,648 ● 8 ● 43 ● 49