What's the use of value types in .Net?

Asked 16 years, 3 months ago Modified 11 years, 4 months ago

Viewed 1k times

3

The official guidelines suggest that there can be very few practical uses for these. Does anyone have examples of where they've put them to good use?

.net

Share

Improve this question

Follow

asked Sep 1, 2008 at 13:22

Luke Halliwell

7,352 ● 6 ● 32 ● 31

8 Answers

Sorted by: Highest score (default)

1

Au Contrare... you'll find C/C++ people flocking to structs a.k.a. value types.

An example would be data packets. If you have a large number of data packets to transfer/transmit, you'd use value structs to model your data packets.

reason: Turning something into a class adds an overhead of (approx 8-16 Bytes I forget) of **overhead in the object header** in addition to the instance data. In scenarios

where this is unacceptable, value types are your safest bet
Another use would be situations where you need **value type semantics** - once you create-initialize a object, it is readonly/immutable and can be passed around to n clients.

Share Improve this answer Follow

answered Sep 1, 2008 at 13:43
Gishu
137k ● 47 ● 226 ● 311

1

For the most part, it's good to emulate the behaviour of the framework. Many elementary data types such as `int`s are value types. If you have types that have similar properties, use value types. For example, when writing a `Complex` data type or a `BigInteger`, value types are the logical solution. The same goes for the other cases where the framework used value types: `DateTime`, `Point`, etc.

When in doubt, use a reference type instead.

Share Improve this answer Follow

answered Sep 1, 2008 at 14:00
Konrad Rudolph
545k ● 139 ● 956 ● 1.2k

0

Enums are first class citizens of .NET world. As for structures I found that in most cases classes can be used, however for memory-intense scenarios consider using structures. As a practical example I used structures as data structures for OSCAR (ICQ) protocols primitives.

Share Improve this answer

Follow

answered Sep 1, 2008 at 13:26

aku
124k ● 33 ● 176 ● 203

0

I tend to use enum for avoiding magic numbers, this can be overcome by const I guess, but enum allows you to group them up.

i.e

```
enum MyWeirdType {
TypeA, TypeB, TypeC};

switch(value){
case MyWeirdType.TypeA:
...
```

Share Improve this answer

Follow

answered Sep 1, 2008 at 13:28

Chris James
11.7k ● 12 ● 63 ● 90

0

Exactly what most other people use them for.. Fast and light data/value access. As well as being ideal for making grouping properties (where it makes sense of course) into an object.

For example:

- Display/Data value differences, such as String pairs of image names and a path for a control (or whatever). You want the path for the work under the hood, but the name to be visible to the user.

- Obvious grouping of values for the metrics of objects. We all know Size etc but there may be plenty of situations where the base "metric" types are not enough for you.
- "Typing" of enum values, being more than a fixed enum, but less that a full blown class (already has been mentioned, just want to advocate).

Its important to remember the differences between value and reference types. Used properly, they can really improve efficiency of your code as well as make the object model more robust.

Share Improve this answer Follow

answered Sep 1, 2008 at 13:49

Rob Cooper
28.9k ● 26 ● 105 ● 142

0

Value types, specifically, structs and enums, and have proper uses in object-oriented programming.

Enums are, as aku said, first class citizens in .NET, which can be used from all sorts of things from Colors to DialogBox options to various types of flags.

Structs, as far as my experience goes, are great as Data Transfer Objects; logicless containers of data especially when they comprise mostly of primitive types.

And of course, primitive types are all value types, which resolve to System.Object (unlike in Java where primitive types aren't related to structs and need some sort of wrapper).

Share Improve this answer Follow

answered Sep 1, 2008 at 13:50

Jon Limjap
95.3k ● 15 ● 103 ● 153

0

Actually prior to .net 3.5 SP1 there has been a performance issue with the intensive use of value types as mentioned here in Vance Morrison's blog.

As far as I can see the vast majority of the time you should be using classes and the JITter should guarantee a good level of performance.

structs have 'value type semantics', so will pass by value rather than by reference. We can see this difference in behaviour in the following example:-

```
using System;

namespace StructClassTest {

  struct A {
    public string Foobar { get; set; }
  }

  class B {
    public string Foobar { get; set; }
  }

  class Program {
    static void Main() {
      A a = new A();
      a.Foobar = "hi";
      B b = new B();
      b.Foobar = "hi";

      StructTest(a);
      ClassTest(b);

      Console.WriteLine("a.Foobar={0}, b.Foobar={1}",

      Console.ReadKey(true);
    }

    static void StructTest(A a) {
      a.Foobar = "hello";
    }

    static void ClassTest(B b) {
      b.Foobar = "hello";
    }
  }
}
```

The struct will be passed by value so StructTest() will get it's own A struct and when it changes a.Foobar will change the Foobar of its new type. ClassTest() will receive a reference to b and thus the .Foobar property of b will be changed. Thus we'd obtain the following output:-

```
a.Foobar=hi, b.Foobar=hello
```

So if you desire value type semantics then that would be another reason to declare something as a struct. Note interestingly that the DateTime type in .net is a value type, so the .net architects decided that it was appropriate to assign it as such, it'd be interesting to determine why they did that :-)

Share Improve this answer Follow

answered Sep 1, 2008 at 13:53

ljs
37.8k ● 36 ● 109 ● 124

0

You should use a value type whenever:

- The use of a class isn't necessary (no need for inheritance)
- You want to make sure there's no need to initialize the type.
- You have a reason to want the type to be allocated in stack space
- You want the type to be a complete independent entity on assigment instead of a "link" to the instance as it is in reference types.

Share Improve this answer Follow

edited Aug 1, 2013 at 16:49

Daniel Daranas
22.6k ● 9 ● 65 ● 121

answered Sep 1, 2008 at 13:46

Jorge Córdoba
52.1k ● 11 ● 82 ● 130

I might add : when you don't need identity for your entity. Such as a 2D point or so on. – Andrei Rînea Oct 8, 2008 at 12:51