

# Reflection in 1.1 and 2.0

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.clr/2006-09/msg00024.html>

---

- *From:* Eric <EricS@xxxxxxxxxxxxxxxxxxxx>
  - *Date:* Tue, 12 Sep 2006 12:50:01 -0700
- 

I have here a little sample which does reflection on a class.  
I struck me that 1.1 is nearly two times faster.  
Could me somebody give a hint why?

For the sample: Compile it via "csc.exe /optimize+ Program.cs" with the two compiler for 1.1 and 2.0.

Thanks  
Eric

```
namespace ComponentProfiling
{
    using System;
    using System.ComponentModel;
    using System.Reflection;

    public sealed class Person
    {
        private string _firstName;
        private string _lastName;
        private int _age;

        public string FirstName
        {
            get { return _firstName; }
            set { _firstName = value; }
        }

        public string LastName
        {
            get { return _lastName; }
            set { _lastName = value; }
        }

        public int Age
        {
            get { return _age; }
            set { _age = value; }
        }
    }
}
```

## Reflection in 1.1 and 2.0

```
}  
}  
  
class Program  
{  
    static void Main(string[] args)  
    {  
        Console.WriteLine("Runtime version = {0}", Environment.Version);  
  
        int iterations = args.Length > 0 ? int.Parse(args[0]) : 1000000;  
        Console.WriteLine("Iterations = {0}", iterations.ToString("N0"));  
  
        Person person = new Person();  
  
        PropertyDescriptorCollection properties =  
        TypeDescriptor.GetProperties(typeof(Person));  
  
        string test1 = typeof(TypeDescriptor).Name;  
  
        TimeSpan span1 = Measure(iterations, person,  
        properties["FirstName"],  
        properties["LastName"],  
        properties["Age"]);  
  
        Console.WriteLine("{0} = {1}", test1, span1);  
  
        string test2 = typeof(MyPropertyDescriptor).Name;  
  
        TimeSpan span2 = Measure(iterations, person,  
        new  
        MyPropertyDescriptor(typeof(Person).GetProperty("FirstName")),  
        new MyPropertyDescriptor(typeof(Person).GetProperty("LastName")),  
        new MyPropertyDescriptor(typeof(Person).GetProperty("Age")));  
  
        Console.WriteLine("{0} = {1}", test2, span2);  
  
        Console.WriteLine("{0}/{1} = {2:P}", test1, test2,  
        span1.TotalMilliseconds / span2.TotalMilliseconds);  
  
        Console.WriteLine("Press ENTER to end.");  
        Console.ReadLine();  
    }  
  
    static TimeSpan Measure(int iterations, Person person, params  
    PropertyDescriptor[] properties)  
    {  
        DateTime start;  
        start = DateTime.Now;  
  
        for (int i = 0; i < iterations; i++)  
        {
```

## Reflection in 1.1 and 2.0

```
properties[0].SetValue(person, "John");
properties[1].SetValue(person, "Doe");
properties[2].SetValue(person, 22);
}

return DateTime.Now - start;
}

public sealed class MyPropertyDescriptor : PropertyDescriptor
{
    private readonly PropertyInfo _property;

    public MyPropertyDescriptor(PropertyInfo property)
    :
    base(property.Name, null)
    {
        _property = property;
    }

    public override bool CanResetValue(object component)
    {
        return false;
    }

    public override object GetValue(object component)
    {
        return _property.GetValue(component, null);
    }

    public override void ResetValue(object component)
    {
        throw new NotSupportedException();
    }

    public override void SetValue(object component, object value)
    {
        _property.SetValue(component, value, null);
    }

    public override bool ShouldSerializeValue(object component)
    {
        return false;
    }

    public override Type ComponentType
    {
        get { return _property.ReflectedType; }
    }

    public override bool IsReadOnly
    {
```

## Reflection in 1.1 and 2.0

```
get { return !_property.CanWrite; }  
}
```

```
public override Type PropertyType  
{  
    get { return _property.PropertyType; }  
}  
}  
}
```

.