# The Right Way to Use Reflection

Yes, we all know reflection is slow, but sometimes it is necessary to use it to satisfy a business requirement in our application. Just like anything, there is a right way and a wrong way to use reflection. Microsoft has made significant improvements in performance over the years for getting and setting properties. Make sure you are using the new ways of using reflection. Before I show you the new way to get and set properties, let's first learn about reflection and the old way of using it.

First off, let's say you have a **Product** class you have created that has a property named **ProductName**. If you wish to set the value of ProductName to a string, you write code like the following:

```
Product entity = new Product();
entity.ProductName = "A New Product";
```

Instead of hard-coding the name of the property to set, you might want to create a generic routine that you can pass a property name to and the value to set that property to. This can be accomplished using reflection as shown in the following code:

```
Product entity = new Product();
typeof(Product).InvokeMember("ProductName",
  BindingFlags.SetProperty,
    Type.DefaultBinder, entity,
     new Object[] { "A New Product" });
```

InvokeMember is a method of the System.Type class. The typeof() method returns an instance of the Type class which contains meta-data about the Product class. You pass 5 parameters to the InvokeMember method. The first parameter is the name of the property to set. The second parameter is the name of the property or method to invoke; in this case it is the Set property. The third parameter specifies to use the default binder. The fourth parameter is the variable with a reference to the instance of the class specified by the type (in this case the Product object). The last parameter is an object array of whatever you need to pass to the method or property you are invoking. For setting the ProductName property you only need a single object array of the string you are setting.

# A Better Way to Set Property Values

While the InvokeMember method works for setting a property, it is quite slow. There is a more efficient way to set a property using reflection. There is a GetProperty method on the Type class you use to retrieve a PropertyInfo object. This PropertyInfo object has a SetValue method that you use to set the value on that property. Below is an example of calling the SetValue method.

```
Product entity = new Product();

typeof(Product).GetProperty("ProductName").
  SetValue(entity, "A New Product", null);

MessageBox.Show(entity.ProductName);
```

The above code is easier to understand than the InvokeMember method and is over 100% faster! That is a big difference, and you should take advantage of it when you need to use reflection to set properties.