使用dynamic在ASP.NET MVC中应用匿名ViewModel

C# 4.0中的新出现的dynamic关键字允许你做到很多以前很难实现的事情，但是dynamic也很容易让你忘记C#本身还是一个强类型的编程语言，所以，在使用中就会产生误区。最近我发现我自己想要创建测试用的View页面，后面有一个简单的控制器相对应。因为是临时使用，所以我使用了一个复杂匿名对象作为Model，并且使用dynamic视图来显示它。结果我错了….

匿名类型的限制

下面我们先看一段简单的示例来说明问题。下面给出控制器中的操作方法，传递一个匿名类型到前台的View模型中：

```
public ActionResult UsingDynamic()
{
  return View(new
  {
    TestString = "This is a test string"
  });
}
```

下面我们来确认一下，这个匿名类型对象，正确的传递到了View中。
在页面中输入：

```
<%: Model %>
```

我们可以得到如下输出:

```
{ TestString = This is a test string }
```

到现在为止,我们看到一个具有TestString属性和值的期望类型.现在我们尝试着在界面上输出TestString的值：

```
<%: Model.TestString %>
```

这时候ASP.NET抛出一个异常：

“object”不包含“TestString”的定义。

怎么会这样呢？我们清楚地看到传递过来的对象的确包含“TestString”属性和值，当我们打印object的时候抱着个错误。其根本原因是：
匿名类型默认访问修饰符为internal

这意味着他们只可以从其定义的程序集中被访问。一旦你超越了程序集的边界，将会被当做普通的object对象被解析，因此不具备TestString的属性。
既然如此，我们怎么能够按照我们预想的那样，可以直接在View中调用到TestString以使用其值呢?

使用POCO
我知道按照下面的写法一定可以解决问题。
首先预定义一个公开的PocoViewModel类型，代码如下：

```
public class PocoViewModel
{
    public string TestString { get; set; }
}
```

然后定义 控制器 中的 操作方法：

```
public ActionResult UsingPoco()
{
    return View(new PocoViewModel()
    {
        TestString = "This is a test string"
    });
}
```

然后View可以直接从ViewPage继承，接下来的使用结果肯定如我们所期望的那样了。
ExpandoObject类
在.NET 4.0中多了一种类型：ExpandoObject。ExpandoObject类型是一种可以再运行时随意动态添加和删除成员的类型。这种特性正是我们所需要的，不仅可以随意调用类成员，而且可以跨程序集访问。使用了ExpandoObject对象的控制器操作方法如下：

```
public ActionResult UsingExpando()
{
    dynamic viewModel = new ExpandoObject();
    viewModel.TestString = "This is a test string";

    return View(viewModel);
}
```