[**如何在ASP.NET MVC5不用EF框架完成对数据库的连接操作**](http://blog.csdn.net/qq_37301611/article/details/58606384)

标签： [asp.net](http://www.csdn.net/tag/asp.net)[数据库](http://www.csdn.net/tag/%e6%95%b0%e6%8d%ae%e5%ba%93)[mvc](http://www.csdn.net/tag/mvc)[框架](http://www.csdn.net/tag/%e6%a1%86%e6%9e%b6)[ado.net](http://www.csdn.net/tag/ado.net)

2017-02-28 17:18 272人阅读 [评论](http://blog.csdn.net/qq_37301611/article/details/58606384?locationNum=14&fps=1#comments)(1) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/qq_37301611/article/details/58606384?locationNum=14&fps=1#report)

版权声明：本文为博主原创文章，未经博主允许不得转载。

/\*

Author: VincentW

Time:2017-2-27

TODO:Documentation For ADO[**.NET**](http://lib.csdn.net/base/dotnet) ConnetionDataBase

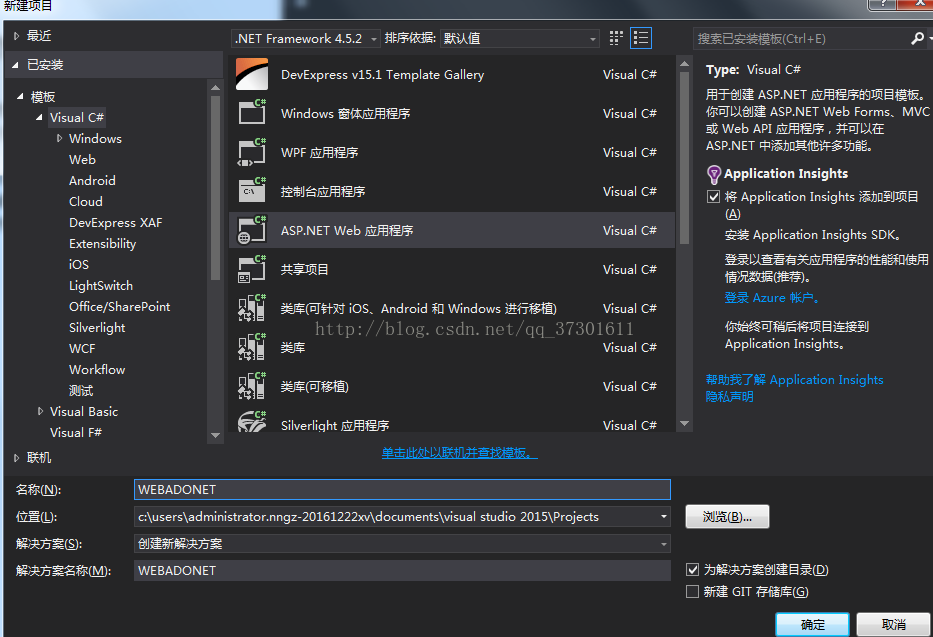
\*/

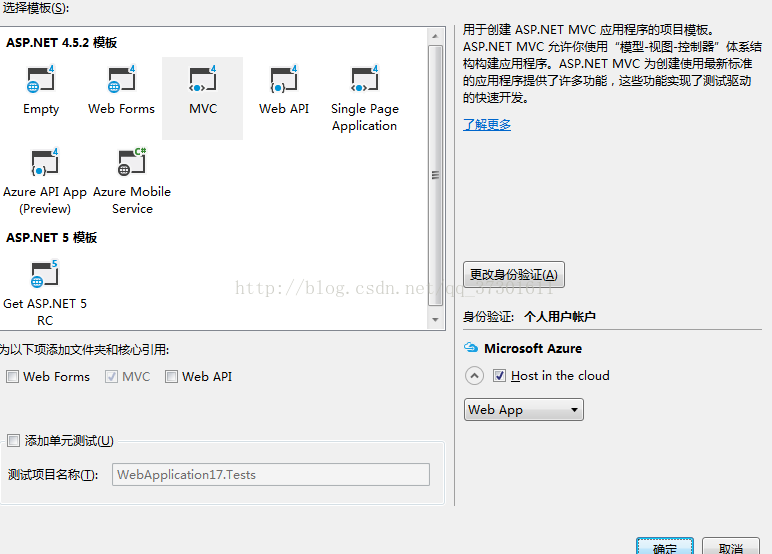
用多了EF框架，今天我们就来讨论一下如何不用EF框架来实现.net MVC对**[数据库](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "_blank)**的操作和连接

首先感谢一下http://www.cnblogs.com/darrenji/p/4625088.html，对我最开始小白时候的ADO[**.Net**](http://lib.csdn.net/base/dotnet)启蒙，并沿用了它的绝大部分代码。如有侵权，请联系删除

**在MVC模式下实现非EF框架的ADO.NET连接数据库并实现增删改查文档。**

**1.  新建一个.NET项目，选择ASP.NET WEB应用程序，输入项目名称为WEBADONET，在弹出的子菜单点选MVC模式，注意不要勾选下面的WebFrom模式，身份验证不需要更改，如果后期有需要再将本Demo复制并进行重构项目集成，点确定，未登录用户会弹出身份验证界面，选Cancel即可。**





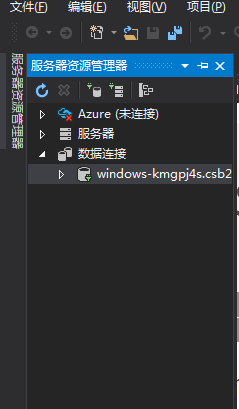
**2.我们看到VS2015界面，这个时候我们会发现项目解决方案下多了很多文件夹，这是VS自带生成的MVC模板，可以照着它的书写方式进行修改，我们先不管这些，我们现在先连接数据库，让本范例的数据可以保存其中。**

**点击工具栏上的工具=》连接数据库，出现以下界面。**



**输入服务器地址，身份验证点选SQL Server验证，输入用户名称和密码，选择数据库的名称，点确定即可。**

**连接成功后会出现如下界面：**



**不同的数据库会有不同的名称，看自己数据库名称而定。点击该数据库，点选属性，复制属性中的连接字符串去修改解决方案下的Web.Configs的连接字符串。**

**步骤如下：**

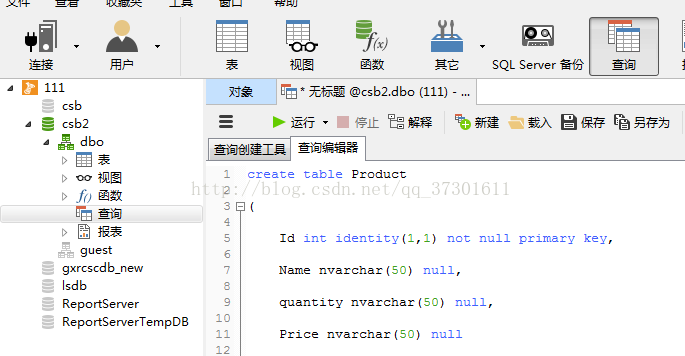
**新增一个连接字符串在ConnectionString节点下，将名称命名为TestConnection,在里面的ConnectionString属性将刚刚复制到的连接字符串复制进去，这里密码要改一下，因为默认的密码是\*\*\*\*\*\*\*\*，修改为可视化密码。**

<addname="TestConnection"connectionString="Data Source=ip地址;InitialCatalog=csb2;User ID=账户;Password=密码"providerName="System.Data.SqlClient" />

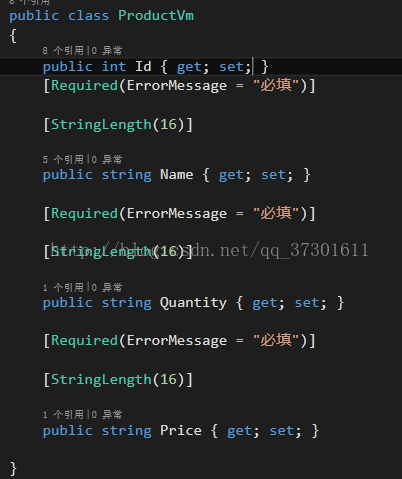
这里需要提醒的是，这不是一个优秀的编写方式，一般来说连接字符串里面不应该有敏感字符，应该不用明文来写，具体的办法在网上可以有详细的解答。在这里我们是为了演示。

**3.在Models文件夹中创建Model，在项目默认出来，需要注意的是，如果希望操作的是数据库中的表，那么在Models中的Models创建就必须类型和数据库中的表一致，不然会导致UpdataDBbaseExcetion.我们先在数据库里面建表，我用的可视化工具是Navicat**

**在数据库中点选查询，新建查询，输入如图所示SQL语句，点运行，那么我们所需要的表就建成了。**



**然后去VS，Models 文件夹下建立与数据库对应的数据模型,新建一个类，命名为ProductVM,如下面的图所示：**



这里模型要和表一一对应，适当的加一些规则，防止一些不可预期的错误。

**4. 这个时候我们数据模型有了，数据库也有了对应的表，那我们就可以开始进行数据库操作了。**

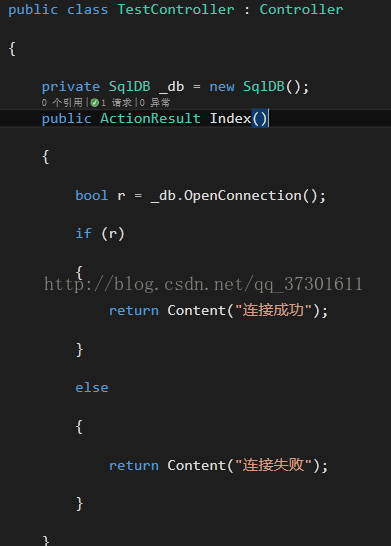
**一般来说我们要封装一个工具类，因为SQL操作是很繁琐的，也很容易出错，对于代码的复用性要求非常高，所以我们现在解决方案的大项目下建立一个Tool文件夹，并在里面建立一个叫SqlDB.cs的类，当然名字可以单独取。在里面输入数据库的开关连接，增删改查等基础工作。**

**public class SqlDB  
  
    {  
  
        protected SqlConnection conn;  
  
        /\*打开连接\*/  
  
        public bool OpenConnection()  
  
        {  
  
            conn = new SqlConnection(ConfigurationManager.ConnectionStrings["DefaultConnection"].ConnectionString);  
  
            try  
  
            {  
  
                bool result = true;  
  
                if (conn.State.ToString() != "Open")  
  
                {  
  
                    conn.Open();  
  
                }  
  
                return result;  
  
            }  
  
            catch (SqlException ex)  
  
            {  
  
                return false;  
  
            }  
  
        }  
  
        /\*关闭连接\*/  
  
        public bool CloseConnection()  
  
        {  
  
            try  
  
            {  
  
                conn.Close();  
  
                return true;  
  
            }  
  
            catch (Exception ex)  
  
            {  
  
                return false;  
  
            }  
  
        }  
  
        /\*数据库操作类\*/  
    
  
        public int InsertData(string sql)  
  
        {  
  
            int lastId = 0;  
  
            //string query = sql + ";SELECT @@Identity;";  
  
            try  
  
            {  
  
                if (conn.State.ToString() == "Open")  
  
                {  
  
                    SqlCommand cmd = new SqlCommand(sql, conn);  
  
                    //cmd.ExecuteNonQuery();  
  
                    lastId = ToInt(cmd.ExecuteScalar());//返回第一行的第一列  
  
                }  
  
                return ToInt(lastId);  
  
            }  
  
            catch (Exception ex)  
  
            {  
  
                return 0;  
  
            }  
  
        }  
        
        /\*转换成整型\*/  
  
        private int ToInt(object o)  
  
        {  
  
            try  
  
            {  
  
                return int.Parse(o.ToString());  
  
            }  
  
            catch (Exception ex)  
  
            {  
  
                return 0;  
  
            }  
  
        }  
  
    }**

**其实这里面代码主要就是四个方法：对于数据库的开关，数据库操作以及返回值变成整形数的四个方法而已，在Demo文档内有更清晰的解释。**

**5.**

**创建一个[测试](http://lib.csdn.net/base/softwaretest" \o "软件测试知识库" \t "_blank)控制器，在Controller文件夹下创建一个TestController,点选空的MVC控制器，确定后输入以下代码：**



**到了这一步，按下快捷键F5运行项目，路由地址是“~/Test/Index”，如果出现测试成功，那说明没有问题，如果测试失败，请重新检查步骤。**

**6．测试成功后开始进行基础的增删改查操作，这里举一个范例，新增来说明：**

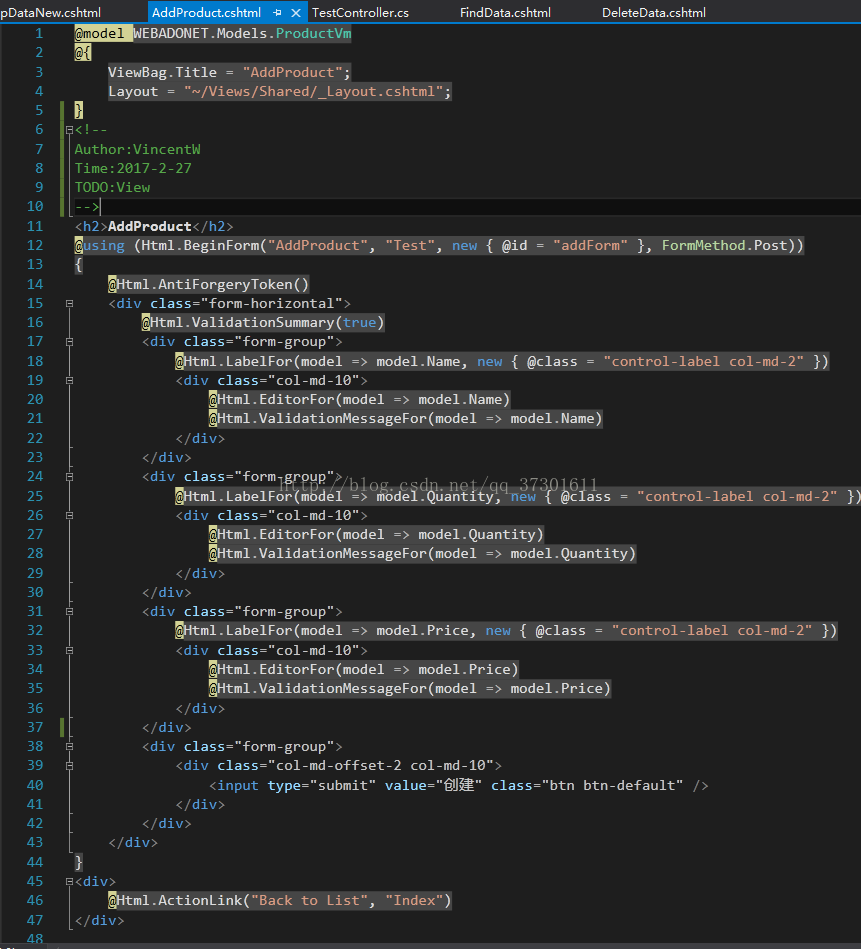
**在TestControllers的文档下，这时候已经有了一个Index方法，我们这时候新建两个方法，返回值依然是ActionResult,名字就取名叫AddProduct,注意，形参的不同，一个是要传入实例化的Models对象的。**



**[HttpPost]，指的是需要POST请求才可以访问，这也是一种.NET自带的辨识思维，默认的浏览器都是GET请求，这里用到了POST那么浏览器进入AddProduct这个方法的时候，会识别出是哪一个方法。**

**[ValidateAntiForgeryToken],防止不合理的请求，攻击数据库。可加可不加，为了安全性考虑建议添加。**

**7．有了ACTION代码，自然需要View视图，右键点击代码中的方法名AddProduct，添加视图，输入如下代码：**



**最上方的model表示传入的model数据是我们定义的模型，这样下面的读取对应的数据的时候才能读取得到**

**值得一说的是model=>model.Price类似语句其实是兰布达表达式的，上方引用布局页是因为Share文件夹中的东西是全局的，ValidationSummary是验证规则，保证避免models错误。**

**其实都是一些**[**HTML5**](http://lib.csdn.net/base/html5)**和CSS的知识，笔者[Html5](http://lib.csdn.net/base/html5" \o "HTML5知识库" \t "_blank)/CSS知识不好，界面也是相对简单粗暴。**

8.

**有了新建数据这个模板，那么自然要有删除**

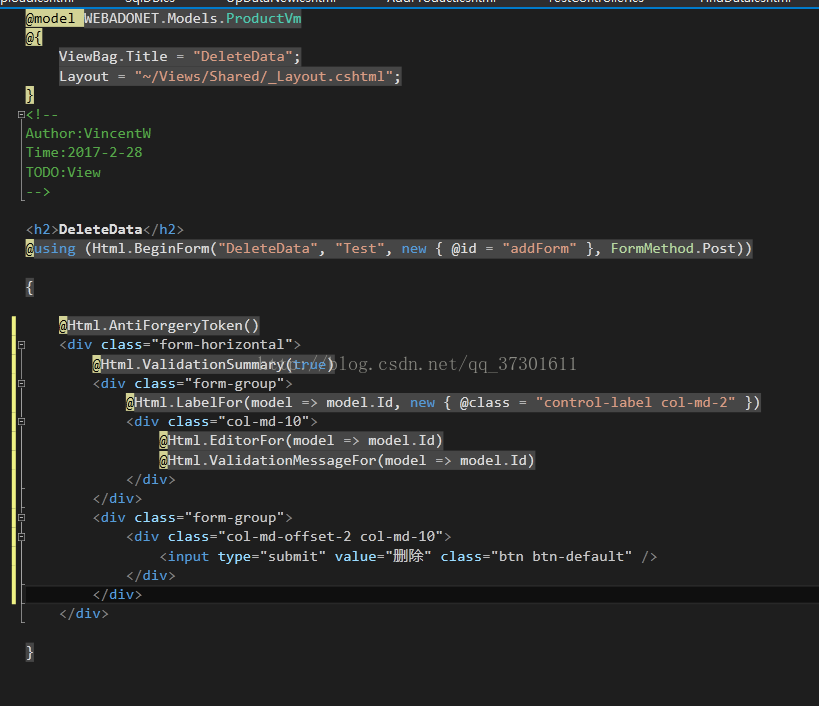
**在TestController中添加两个同名方法DeleteData,其中一个要有传入数据模型的，仿照创建的方法，两个方法的异同只是在SQL语句的不同而已**



**这里需要注意的是，关闭数据库应该在if条件里或者if条件之前，这样才能有效释放资源。**

**接下来我们添加视图：**

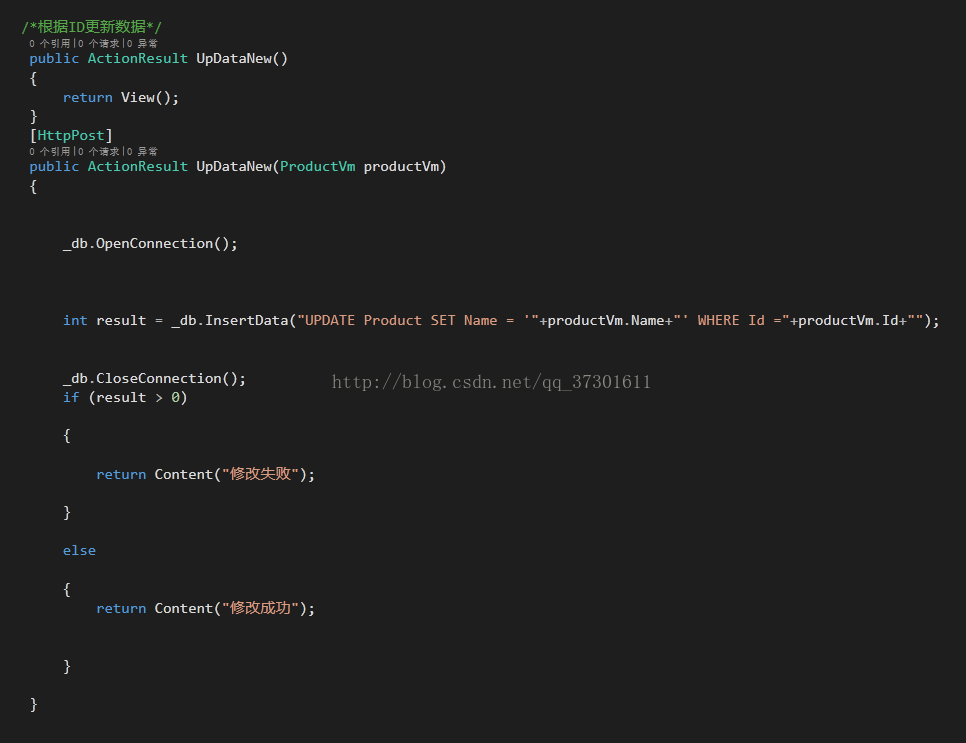
**右键点击方法名，点添加视图，注意选择的是Empty，没有视图模型，使用布局页，如图所示：**

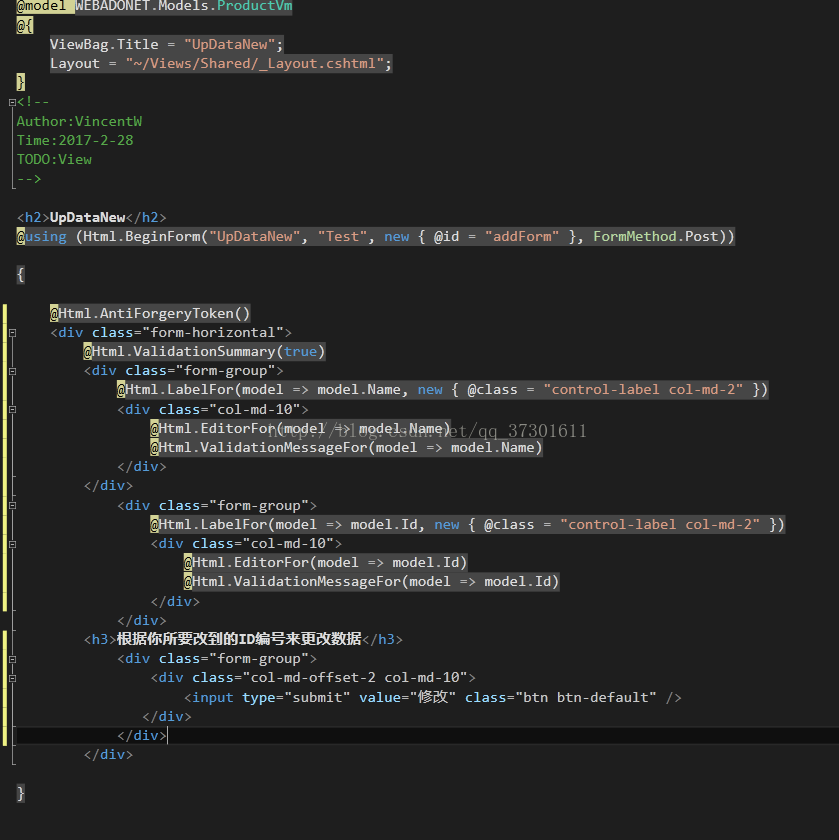
****

**9**

**布局结构都是一样的，最顶上的是模型引用，Html.BeginForm是用于写入响应，四个参数分别是：方法名，控制器名，路由值，指定的HTTP方法，这里指的是响应以后用到的值**

**10.更新数据和删除数据大同小异，也是一个SQL语句不同而已，也在TestControllers新建两个方法如图所示**

  
紧接着填入视图：

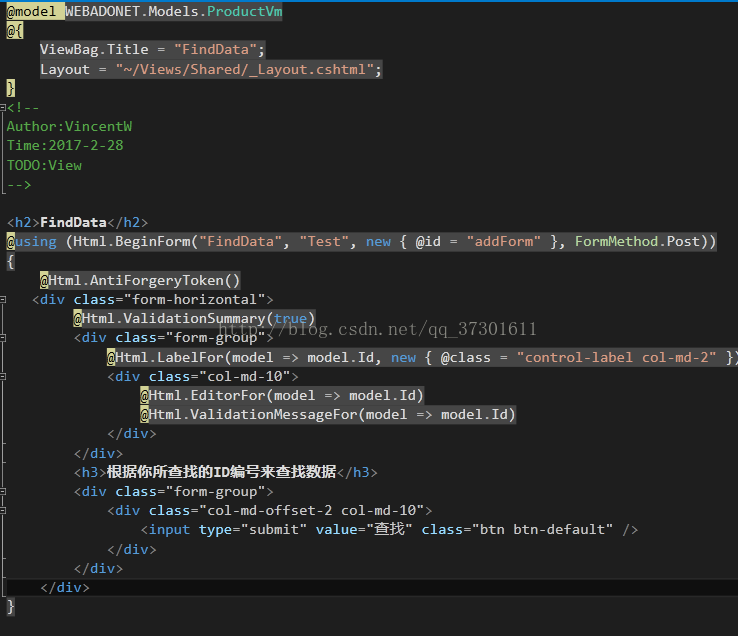


11.

**查找方法，查找方法和其他方法不太一样，因为我们需要查看到方法是否已经查到，那么我们按理说应该返回一个json或者xml，但是这里为了减少代码范例的复杂度，我们只返回字符串，因为返回字符串，那就不能在工具类里面调用数据库操作方法，否则就会被迫使用全局变量的指针来返回。这里是在方法中写入对SQL的操作。注释也对里面的SQL方法进行了解释。**

****

**添加视图：**

****

**.万里长征最后一步，按下F5运行，路由方面就是“~/test/addproduct”,如图所示，输入数据后点击创建，然后去数据库看看吧！**

