[七天学会ASP.NET MVC (一)——深入理解ASP.NET MVC](http://www.cnblogs.com/powertoolsteam/p/MVC_one.html)

**[](http://images0.cnblogs.com/blog/139239/201506/121203406767021.jpg)**

**系列文章**

[**七天学会ASP.NET MVC (一)——深入理解ASP.NET MVC**](http://www.cnblogs.com/powertoolsteam/p/MVC_one.html)

[**七天学会ASP.NET MVC (二)——ASP.NET MVC 数据传递**](http://www.cnblogs.com/powertoolsteam/p/MVC_two.html)

[**七天学会ASP.NET MVC (三)——ASP.Net MVC 数据处理**](http://www.cnblogs.com/powertoolsteam/p/MVC_three.html)

[**七天学会ASP.NET MVC (四)——用户授权认证问题**](http://www.cnblogs.com/powertoolsteam/p/MVC_four.html)

[**七天学会ASP.NET MVC (五)——Layout页面使用和用户角色管理**](http://www.cnblogs.com/powertoolsteam/p/MVC_five.html)

[**七天学会ASP.NET MVC (六)——线程问题、异常处理、自定义URL**](http://www.cnblogs.com/powertoolsteam/p/MVC_Six.html)

[**七天学会ASP.NET MVC（七）——创建单页应用**](http://www.cnblogs.com/powertoolsteam/p/MVC_Seven.html)

**ASP.NET vs MVC vs WebForms**

许多ASP.NET开发人员开始接触MVC认为MVC与ASP.NET完全没有关系，是一个全新的Web开发，事实上ASP.NET是创建WEB应用的框架而MVC是能够用更好的方法来组织并管理代码的一种更高级架构体系，所以可以称之为ASP.NET MVC。

我们可将原来的ASP.NET称为 ASP.NET Webforms，新的MVC 称为ASP.NET MVC.

**ASP.NET Web Form**

ASP.NET 在过去的十二年里，已经服务并成功实现Web 应用的开发。我们首先了解一下为什么ASP.NET能够如此流行，并成功应用。

微软编程语言从VB开始就能够成为流行并广泛应用，都源于其提供的强大的Visual studio能够进行可视化的编程，实现快速开发。

使用VS时，开发人员能够通过拖拽UI元素，并在后台自动生成这些界面的代码。称为后台代码。在后台代码中开发人员可以添加操作这些UI元素的逻辑代码。

[](http://images0.cnblogs.com/blog/139239/201506/121203413943635.png)

因此微软的可视化RAD架构体系有两方面组成，一方面是UI，一方面是后台代码。因此ASP.NET Web 窗体，包含ASPX和ASPX.CS，WPF包含XAML/XAML.CS等。

**ASP.NET Web Form存在的问题**

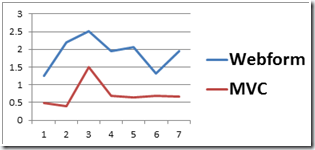
我们不得不考虑的问题是，既然ASP.NET Web Form 如此成功且具有优势，为什么微软还要推出ASP.NET MVC?主要是因为ASP.NET Webform的性能问题。在Web应用程序中从两方面来定义性能：

1. 响应时间： 服务器响应请求的耗时

2. 带宽消耗： 同时可传输多少数据。

**响应时间**

我们可以理解为什么ASP.NET Webform比较慢，如图我们做了一些小的加载测试。分别使用ASP.Net MVC和ASP.Net Webform,发现ASP.Net MVC的响应时间比Webform快了两倍。

**[](http://images0.cnblogs.com/blog/139239/201506/121203419411535.png)**

接下来我们在思考一个问题为什么ASP.NET MVC的性能更好？看看下面这个示例，简单的UI代码和UI的后台代码。

假如一个textbox的ASPX页面：

<asp:TextBox ID="TextBox1" runat="server">

对应的UI后台代码：

1: protected void Page\_Load(object sender, EventArgs e)

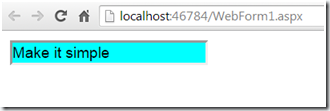
2: {

3: TextBox1.Text = "Make it simple";

4: TextBox1.BackColor = Color.Aqua;

5: }

运行结果：

**[](http://images0.cnblogs.com/blog/139239/201506/121203425829406.png)**

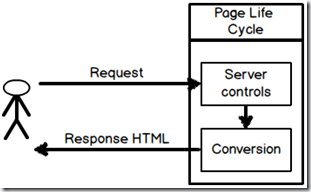
如果查看HTML输出，则会显示如下代码：

<input name="TextBox1" type="text" value="Make it simple" id="TextBox1" style="background-color:Aqua;" />

我们再来思考上面提到的问题

1. 这种HTML生成方式是否很有效？我们是否为了获取如此简单的HTML而长时间的消耗服务器

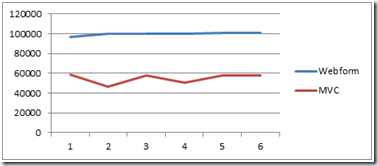
2. 开发人员是否可以直接编写HTML?很难实现吗？

**[](http://images0.cnblogs.com/blog/139239/201506/121203434417276.png)**

通过分析我们可以得知，每一次请求都有转换逻辑，运行并转换服务器控件为HTML输出。如果我们的页面使用表格，树形控件等复杂控件，转换就会变得很糟糕且非常复杂。HTML输出也是非常复杂的。由于这些不必要的转换从而增加了响应时间。该问题的解决方案就是摆脱后台代码，写成纯HTML代码。

**带宽消耗**

ASP.NET开发人员都非常熟悉Viewstates，因为它能够自动保存post返回的状态，减少开发时间。但是这种开发时间的减少会带来巨大的消耗，Viewstate增加了页面的大小。在做的加载测试中，与MVC 对比，我们发现Viewstate增加了两倍的页面存储。以下是测试结果：

**[](http://images0.cnblogs.com/blog/139239/201506/121203441296432.png)**

页面尺寸的增加是因为viewstate产生了额外的字节。下图就是Viewstate的截图。许多人可能会不同意此观点，但是众所周知，开发人员是如何工作的，如果有选择，他们肯定会采取别的选择。

**[](http://images0.cnblogs.com/blog/139239/201506/121203448161290.jpg)**

1. **HTML 消耗**

现在因为我们都是后台代码和ASP.NET web server控件的努力，我们对于怎样得到HTML以及如何使他们更有效没有更好的办法。如下面展示的ASPX 代码，你能确定会生成什么样的HTML代码吗？

* <asp:Label ID="Label1" runat="server" Text="I am label">

* <asp:Literal ID="Literal1" runat="server" Text="I am a literal">

* <asp:Panel ID="Panel1" runat="server">I am a panel

Lable标签会生成DIV标签还是SPAN标签？运行后生成的HTML代码的结果如下：label生成了span标签，Literal生成了转换为了简单的文本，而panel转换为了DIV标签。

 <span id="Label1">I am label</span>I am a literal

* **I am a panel**

因此与其生成HTML代码，还不如直接编写HTML代码，并实现HTML控件。

**所以该问题的解决方案是：不使用服务器控件，直接编写HTML代码。**

直接编写HTML代码的好处在于web设计者可以与开发人员紧密合作及时沟通。设计人员可以使用他们喜爱的设计工具来设计HTMl代码，像dream weaver,前端页面等，设计独立。如果我们使用服务器控件，这些设计者工具可能不会识别。

**2. 后台代码类的重用性**

如果仔细观察一些专业的ASP.NET Webform项目，你会发现后台代码类往往都包含了大量的代码，并且这些代码也是非常复杂的。而现在，后台代码类继承了“System.Web.UI.Page”类。但是这些类并不像普通的类一样能够到处复用和实例化。换句话来讲，在Weform类中永远都不可能执行以下代码中的操作：

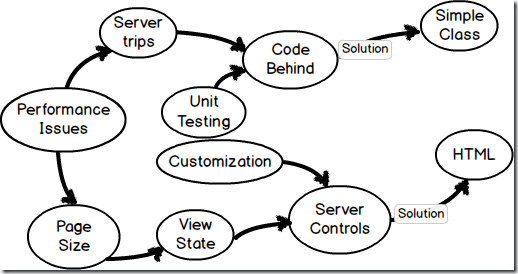
**1: WebForm1 obj = new WebForm1();obj.Button1\_Click();**

* **3. 单元测试**

既然无法实例化后台代码类，单元测试也是非常困难的，也无法执行自动化测试。必须手动测试。

* **解决方案**

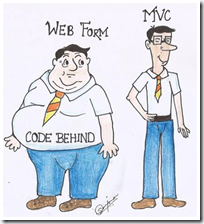
既然讲了ASP.Net Webform存在的两大问题即服务器控件和后台代码，以下是根源图，

[](http://images0.cnblogs.com/blog/139239/201506/121203461914303.png)

那么解决方案是什么？

就是我们需要将后台代码迁移到独立的简单的类库，并且拜托ASP.Net服务器控件，并写一些HTML示例。

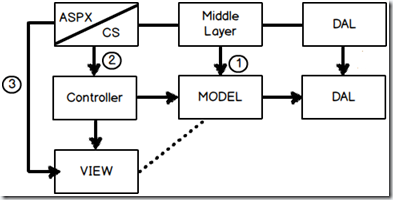
ASP.NET Webform 和MVC 比较，如下图：

[](http://images0.cnblogs.com/blog/139239/201506/121203470353646.png)

**Microsoft Asp.Net MVC 是如何弥补Web Form存在的问题的?**

后台代码和服务器控件是一切问题的根源。所以如果你查看当前的WebForm体系结构，开发者正在使用的包含3层体系结构。三层体系结构是由UI包含ASPX及CS 后台代码。

UI，业务逻辑以及包含数据访问的中间层

[](http://images0.cnblogs.com/blog/139239/201506/121203482234716.png)

Asp.Net MVC 由Model，View，Controller三部分组成。Controller中包含后台代码逻辑，View是ASPX，如纯HTML代码，Model是中间层。通过上图可获得这三部分的关系。

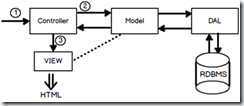
所以会发现MVC的改变有两点，View变成简单的HTML，后台代码移到简单的.NET类中，称为控制器。

以下是ASP.NET MVC 请求流的通用步骤：

Step 1：首先获取控制器。

Step 2：依赖行为控制器创建Model对象，Model通过转换调用数据访问层。

Step 3：数据填充Model之后，传递到View 显示层，实现显示的目的。

[](http://images0.cnblogs.com/blog/139239/201506/121203500192900.png)

到这里我们就已经了解了ASP.Net MVC的各个组件。下面我们做一些小的实验深入了解MVC的各组件。首先我们从Controller 控制器开始，因为Controller是MVC体系架构的核心部分。

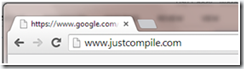
**你是否真的理解Asp.Net MVC的Controller(控制器)？**

为了我们能够更好的理解Controller，我们首先需要理解Controller中涉及的专业术语：用户交互逻辑。

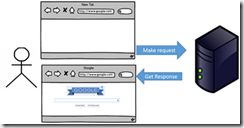
**什么是用户交互逻辑？**

**场景1**

你是否想过当用户输入URL摁下回车键时，会发生什么事情？

[](http://images0.cnblogs.com/blog/139239/201506/121203507235285.png)

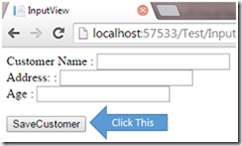
浏览器首先需要给服务器发送请求，服务器再做出响应。

[](http://images0.cnblogs.com/blog/139239/201506/121203517541870.png)

通过这些请求之后，客户端正尝试与服务器交互，服务器能够反馈响应，因为服务器端存在一些判断逻辑来处理这些请求。这些能够处理用户请求以及用户交互行为的业务逻辑称为用户交互逻辑。

场景2

有一种常见的情况，服务器端发送的请求是HTML请求。HTML请求是由一组输入控件和提交按钮组成的。

[](http://images0.cnblogs.com/blog/139239/201506/121203525356699.png)

当用户点击“Save”按钮之后会发生什么？

如果你的回答是有一些事件处理器来处理button点击事件，那么很抱歉回答是错误的。

在Web编程中是没有事件的概念的，Asp.net Web forms 根据我们的行为自动添加了处理代码，所以给我们带来的错觉认为是事件驱动的编程。这只是一种抽象的描述。

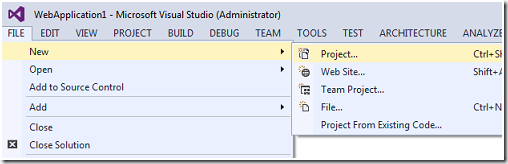
当点击Button时，一个简单的HTTP请求会发送到服务器。差别在于Customer Name,Address以及Age中输入的内容将随着请求一起发送。最终，如果是有个请求，服务器端则有对应的逻辑，使服务器能够更好响应请求。简单来说是将用户交互逻辑写在服务器端。

在Asp.Net MVC中，C代表Controller，就是用来处理用户交互逻辑的。

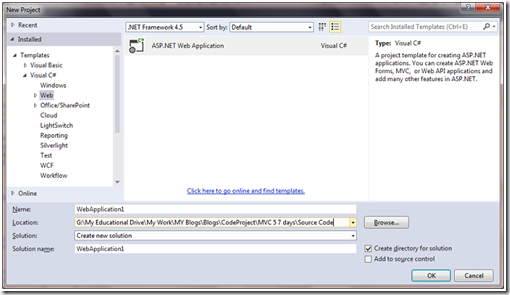
**实验一：简单的MVC Hello world，着重处理Controller。**

* **Step1 创建一个Asp.Net MVC 5项目**

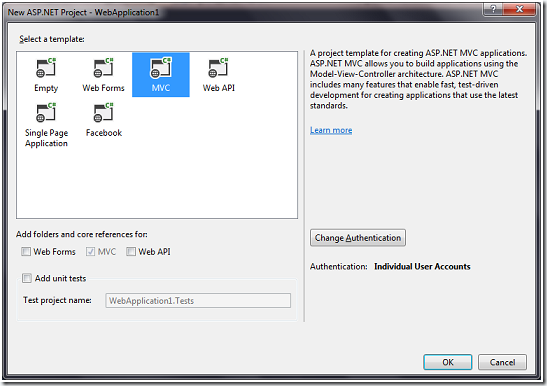
打开Visual studio 2013 点“文件”->新建->项目。

[](http://images0.cnblogs.com/blog/139239/201506/121203534266326.png)

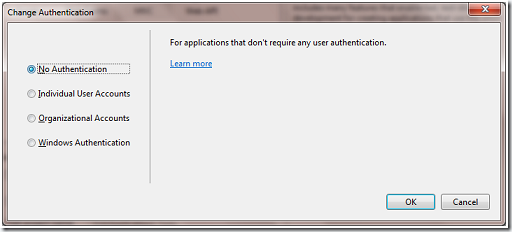
* Step 1.2 选择Web 应用，输入项目名称，选择存放路径，点击确定。

[](http://images0.cnblogs.com/blog/139239/201506/121203542542440.png)

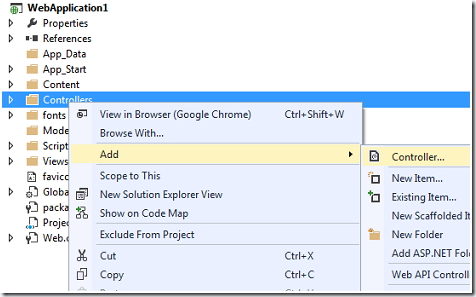
* Step 1.3 选择MVC 模板

[](http://images0.cnblogs.com/blog/139239/201506/121203554106454.png)

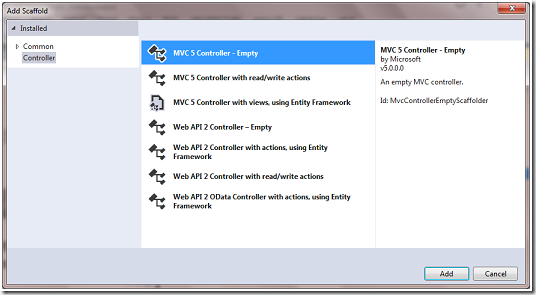
* Step 1.4 选择Change Authentication（改变授权），弹出对话框中选择“No Authentication”，并点击确定。

[](http://images0.cnblogs.com/blog/139239/201506/121203566299280.png)

* Step 2 –创建控制器
* Step 2.1，在资源管理器中，右击controller文件夹，选择添加->Controller(控制器)

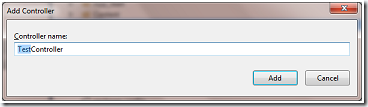
[](http://images0.cnblogs.com/blog/139239/201506/121203575982352.png)

* Step 2.2 选择空 MVC 5 Controller 并点击添加

[](http://images0.cnblogs.com/blog/139239/201506/121203591299849.png)

* Step 2.3 输入控制器的名称”TestController“，点击添加。

在这一步骤中，要特别注意千万不能删除名称中的” Controller”关键字。名称中必须包含Controller关键字。

[](http://images0.cnblogs.com/blog/139239/201506/121204001913893.png)

* Step 3. 创建行为方法

打开新建的TestController 类，可以发现已生成的Index 方法，将该方法删除，并且添加新方法命名为GetString ，代码如下：

1: public class TestController : Controller

2: {

3: public string GetString()

4: { return "Hello World is old now. It’s time for wassup bro ;)";

5:

6: }

7: }

* Step 4. 运行并测试 按 F5 键，在地址栏中以“ControllerName/ActionName”这样的形式输入，需要注意的输入控制器名称时，不能输入”Controller“只输入”Test”。

[](http://images0.cnblogs.com/blog/139239/201506/121204012239477.png)

**实验一：Q&A**

1. TestController 和Test之间的关系是什么？

TestController是类名称，而Test是Controller的名称，请注意，当你在URL中输入controller的名称，不需要输入Controller这个单词。

2. Action（行为） 方法是什么？

Action 方法 简单的来说就是一个Controller内置的public类型的方法，能够接收并处理用户的请求，上例中，GetString 方法返回了一个字符串类型的响应。

注意：在Asp.Net Web Forms中默认的返回请求是HTML的，如果需要返回其他类型的请求，就必须创建HTTP 处理器，重写内容类型。这些操作在Asp.net中是很困难的。在Asp.net MVC中是非常简单的。如果返回类型是”String“直接返回，不需要发送完整的HTML。

3. 如果从Action 方法中返回对象值会出现什么意外情况？

请浏览以下代码

1: namespace WebApplication1.Controllers

2: {

3: public class Customer

4: {

5: public string CustomerName { get; set; }

6: public string Address { get; set; }

7: }

8: public class TestController : Controller

9: {

10: public Customer GetCustomer()

11: {

12: Customer c = new Customer();

13: c.CustomerName = "Customer 1";

14: c.Address = "Address1";

15: return c;

16: }

17: }

18: }

 输出结果如下所示：

[](http://images0.cnblogs.com/blog/139239/201506/121204021446562.png)

当返回类型如“Customer”这样类似的对象时，将调用ToString()方法，返回“NameSpace.ClassName”形式的类名。

4. 如果需要获得上面例子中的属性值，要如何操作？

简单重写类的“ToString”方法，如下：

1: public override string ToString()

2: {

3: return this.CustomerName+"|"+this.Address;

4: }

运行结果：

[](http://images0.cnblogs.com/blog/139239/201506/121204030663648.png)

5. Action 方法是否只能用Public修饰符来修饰？

答案是肯定的，每个公有方法都会自动称为Action 方法。

6. 非public方法是什么？

类的方法都比较简单，并且并不是公共可用的。无法在Web中调用。

7. 如果我们需要其他函数来完成一些特定功能，但不是Action Method要如何实现？

使用NonAction属性修饰，如下：

1: [NonAction]

2: public string SimpleMethod()

3: {

4: return "Hi, I am not action method";

5: }

当尝试给以上Action 方法发送请求时，会获得以下结果：

[](http://images0.cnblogs.com/blog/139239/201506/121204041133462.png)

**View部分**

Controller是处理用户请求，并做出响应，通常情况下响应都是以显示在浏览器中，使用HTML代码，浏览器才可识别。HTML有图像，文本，输入控件等。通常称为用户界面的设计即UI层，在ASP.net MVC称为View。

**实验二——深入理解View**

在实验二中，创建一个简单的MVC应用，仅仅具有Controller和简单的字符串类型的返回值。让我们来了解MVC中的View部分吧。

* Step1 –创建新的Action 方法

在TestController中添加新的Action 方法，如下：

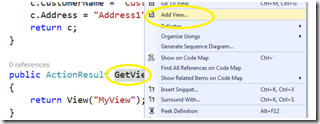
1: public ActionResult GetView()

2: {

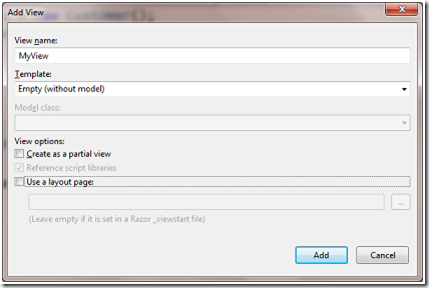
3: return View("MyView");

4: }

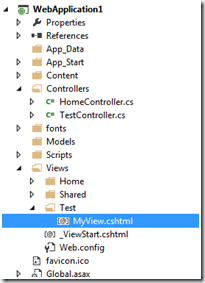
* Step 2 创建View
* Step 2.1 右击上述创建的Action 方法，选择“添加View”

[](http://images0.cnblogs.com/blog/139239/201506/121204050518074.png)

* Step 2.2 在添加View的对话框中输入View名称“MyView”，取消选择“使用布局”的复选框，点击添加。

[](http://images0.cnblogs.com/blog/139239/201506/121204063162589.png)

资源管理器重的Views/Test文件夹中会添加一个新的View文件。

[](http://images0.cnblogs.com/blog/139239/201506/121204075826102.png)

* Step3 在View中添加内容

打开MyView.cshtml 文件，并添加以下内容：

@{Layout = null;}

<!DOCTYPE html>

<html><head><meta name="viewport" content="width=device-width" />

<title>MyView</title>

</head><body>Welcome to MVC 5 Step by Step learning  
</body></html>

* Step 4. 运行 按F5键运行应用

[](http://images0.cnblogs.com/blog/139239/201506/121204083161244.png)

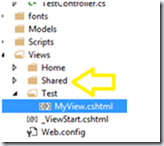
**实验二：Q&A**

1. 为什么View会放在Test的文件夹中？

View是与放置在特定目录下的Controller相关。这个特定文件夹是以”ControllerName”命名的，并且放在View文件夹内

2. 在多个控制器中无法重用View吗？

当然可以，我们需要在将这些文件放在特定的Shared文件夹中。将View 放在Shared文件夹中所有的Controller都可用。

[](http://images0.cnblogs.com/blog/139239/201506/121204091768114.png)

3. 单个Action 方法中可引用多个View吗？

可以，ASP.NET MVC的view和Controller不是严格的匹配的，一个Action Method可以引用多个view，而一个View也可以被一个Action方法使用如下代码所示：

1: public ActionResult GetView()

2: {

3: if(Some\_Condition\_Is\_Matching)

4: {

5: return View("MyView");

6: }

7: else

8: {

9: return View("YourView");

10: }

11: }

4. View函数的功能是什么？

 创建 ViewResult 对象将会渲染成视图来给用户反馈

* ViewResult 创建了ViewPageActivator 对象
* ViewResult 选择了正确的ViewEngine，并且会给ViewEngine的构造函数传ViewPageActivator对象的参数
* ViewEngine 创建View类的对象
* ViewEngine 调用View的RenderView 方法。

5. ActionResult和 ViewResult的关系是什么？

ActionResult是抽象类，而ViewResult是ActionResult的多级孩子节点，多级是因为ViewResult是ViewResultBase的子类，而ViewResultBase是ActionResult的孩子节点。

6. 什么是ContentResult？

ViewResult是HTML响应而ContentResult是标准的文本响应，仅返回字符串类型。区别就在于ContentResult是ActionResult的子类。

经过了本节MVC基础知识的学习，相信大家对MVC已经有了基本的认识。大家在进行ASP.NET MVC的开发时，还可以借助一些开发工具。[ComponentOne Studio ASP.NET MVC](http://www.gcpowertools.com.cn/products/componentonemvc/) 是一款轻量级控件，与Visual Studio无缝集成，完全与MVC6和ASP.NET 5.0兼容，将大幅提高工作效率。