初识MVC

目标：了解MVC原理，构建带 模板的MVC项目。

重点：MVC架构模式、请求响应过程

难点：适应页面请求转控制器请求模式变化和语法表达

# ASP.NET与MVC

扩展阅读：<https://docs.microsoft.com/zh-cn/aspnet/overview>

ASP.NET 是一个用于构建网站和 web 应用程序web 框架。

MVC是组织和管理代码的一种架构体

MVC不专属于ASP.NET，ASP.NET MVC专指基于ASP.NET的MVC

## ASP.NET的历史主流架构—webform

扩展阅读：

https://docs.microsoft.com/zh-cn/aspnet/web-forms/index

## ASP.NET的替代方案—ASP.NET CORE

扩展阅读：

<https://docs.microsoft.com/zh-cn/aspnet/core/?view=aspnetcore-2.2>

## ASP.NET的当前应用架构—MVC

### MVC扩展阅读

<https://docs.microsoft.com/zh-cn/aspnet/mvc/overview/index>

### MVC项目文件组织结构

（与WEBForm比较）

|  |  |
| --- | --- |
| MVC |  |
|  | 控制器：Controllers  视图：Views  配置文件：除web.config之外，各种配置文件进App\_Star文件夹 |

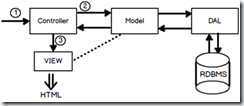
# ASP.NET MVC 基本概念

## 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)

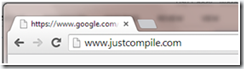
## 控制器Controller

在Asp.Net MVC中，C代表Controller，用于**处理用户交互逻辑**的。

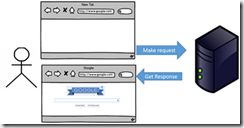
### 什么是用户交互逻辑？

**场景1**

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)

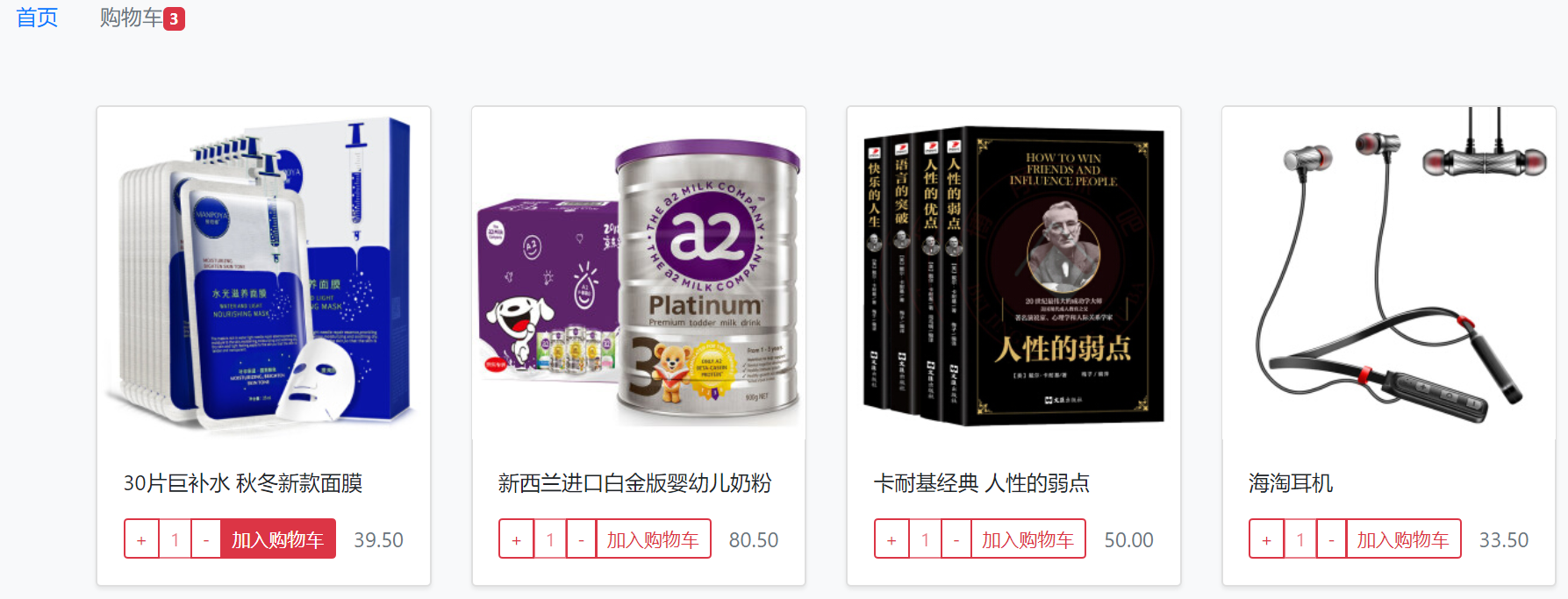
浏览器首先需要给服务器发送请求，服务器再做出响应。

[](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。



当用户点击“加入购物车”按钮之后会发生什么？

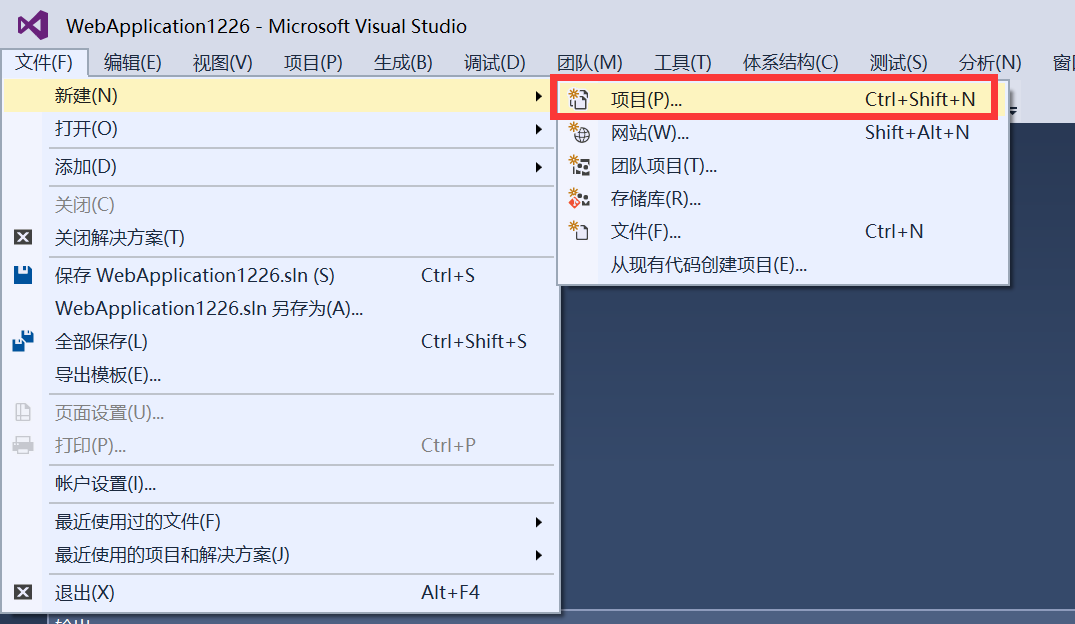
客户端编程—触发单击事件

服务器端编程—向服务器端发送一个简单的HTTP请求。服务器端用相应的逻辑，响应请求。简单来说是将用户交互逻辑写在服务器端。

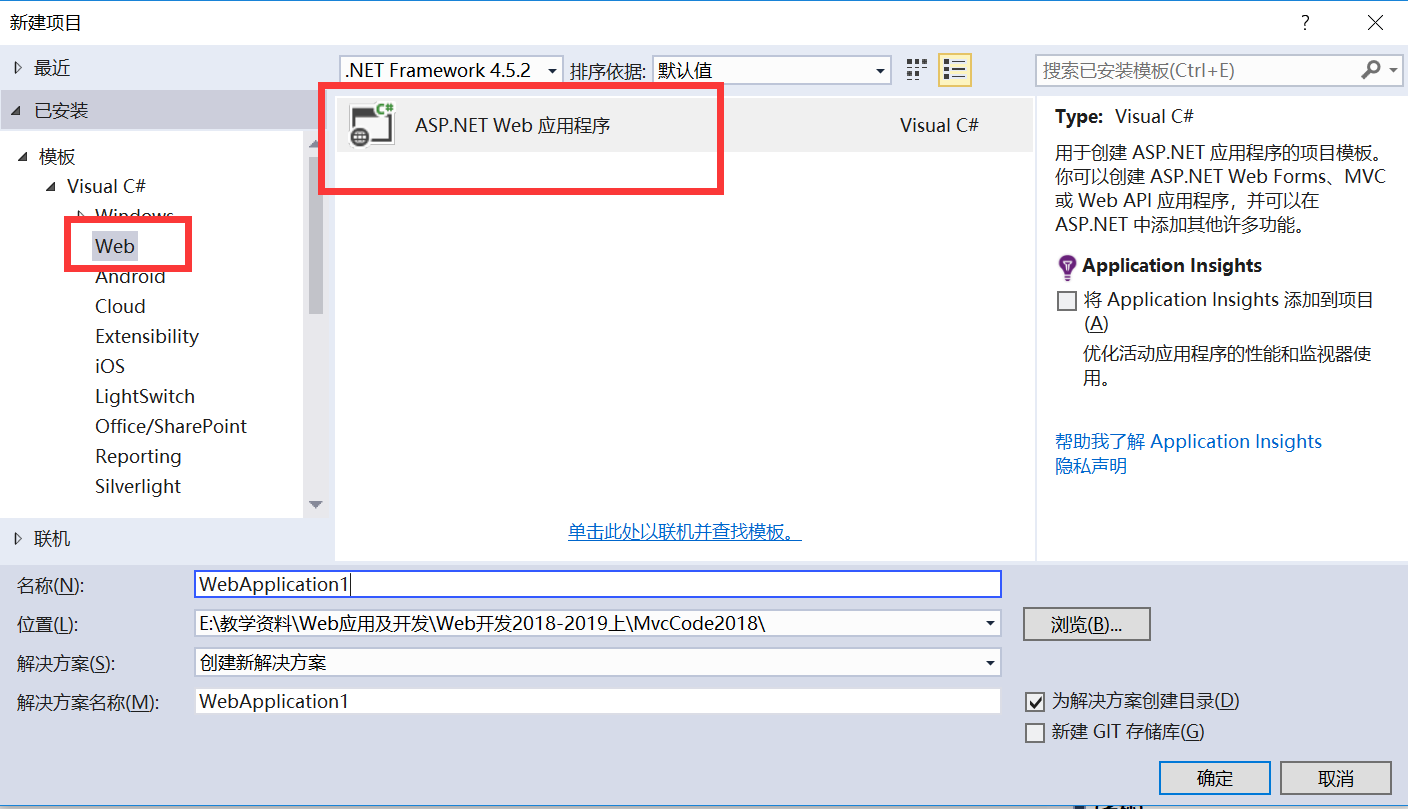
### 实验1 MVC Hello world—学习控制器

* **Step1 创建一个Asp.Net MVC 5项目**

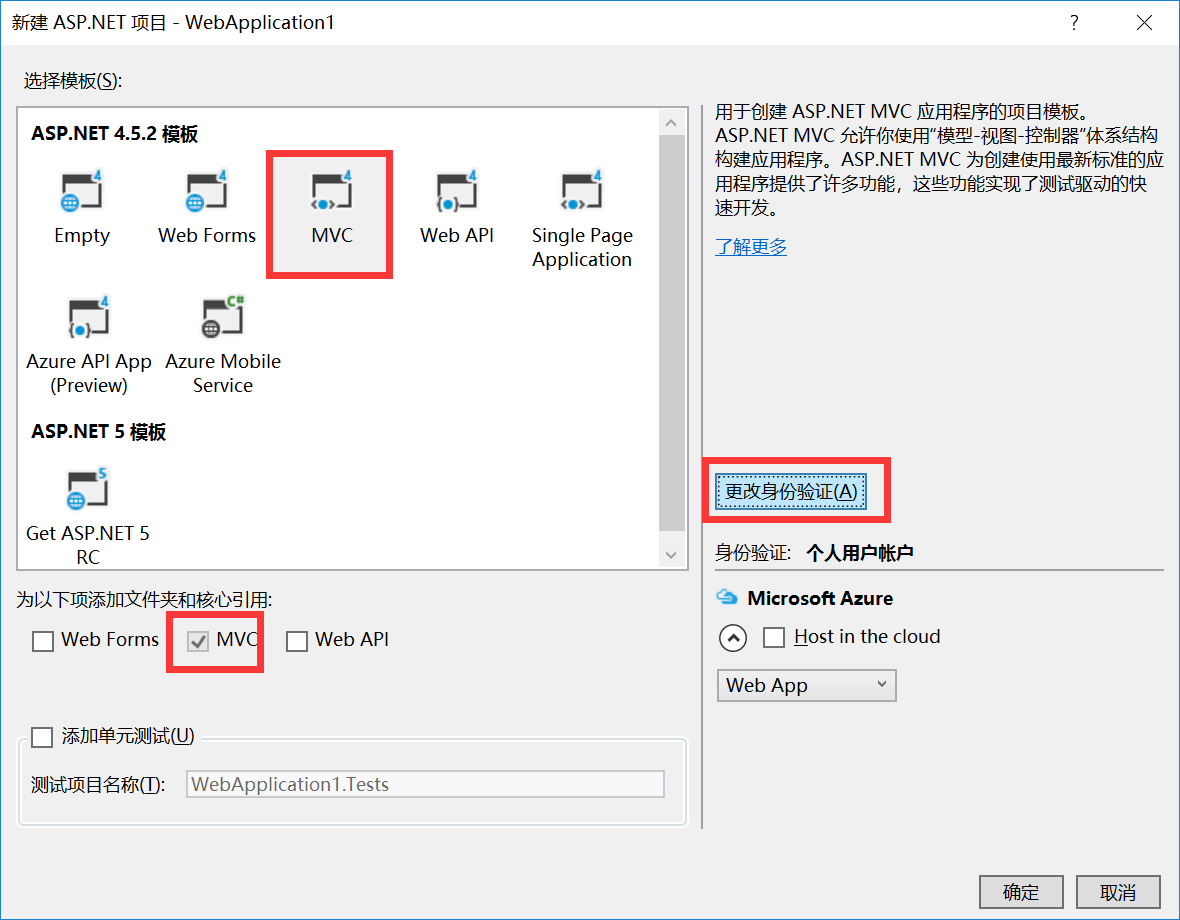
打开Visual studio 点“文件”->新建->项目。



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



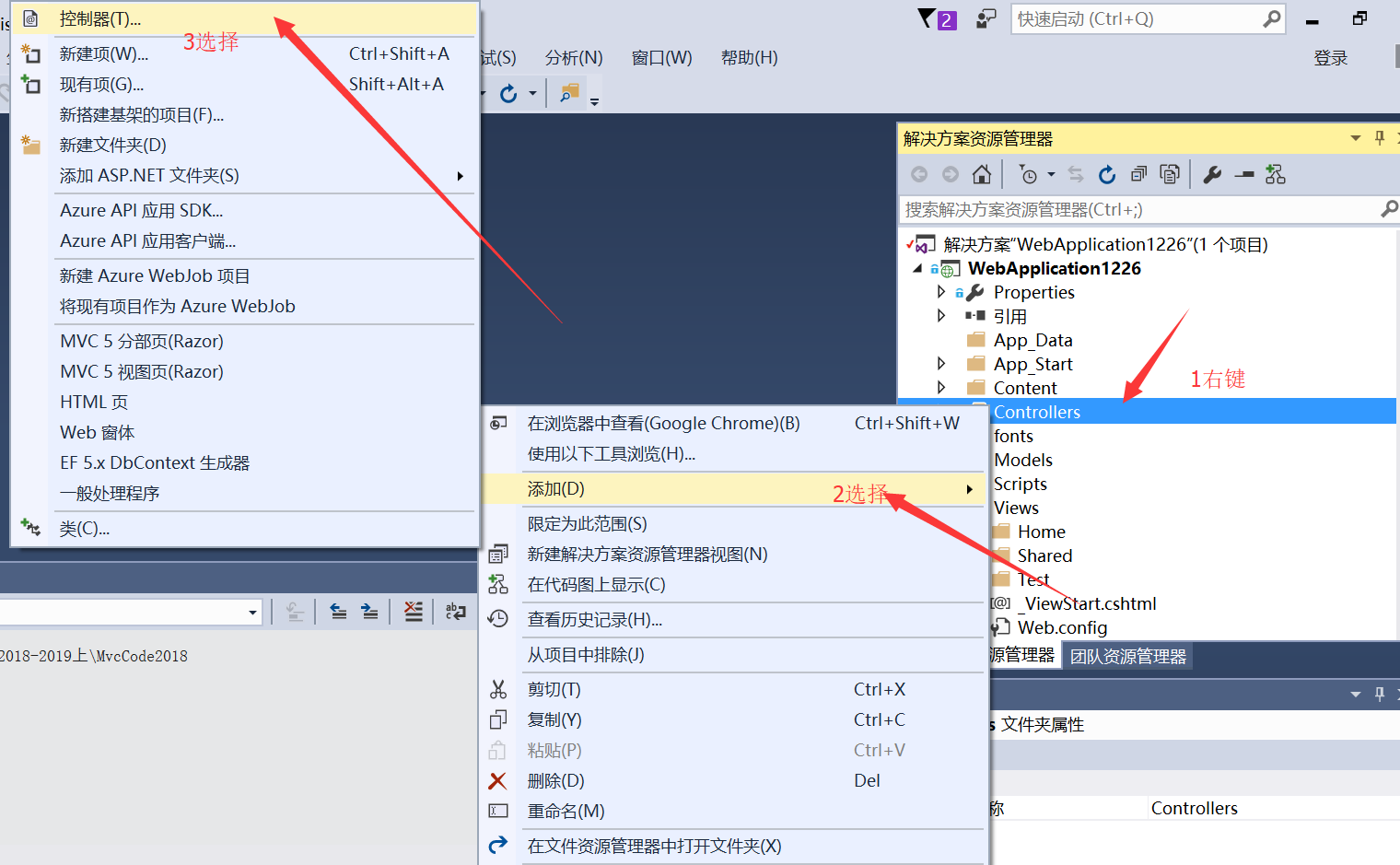
* Step 1.3 选择MVC 模板



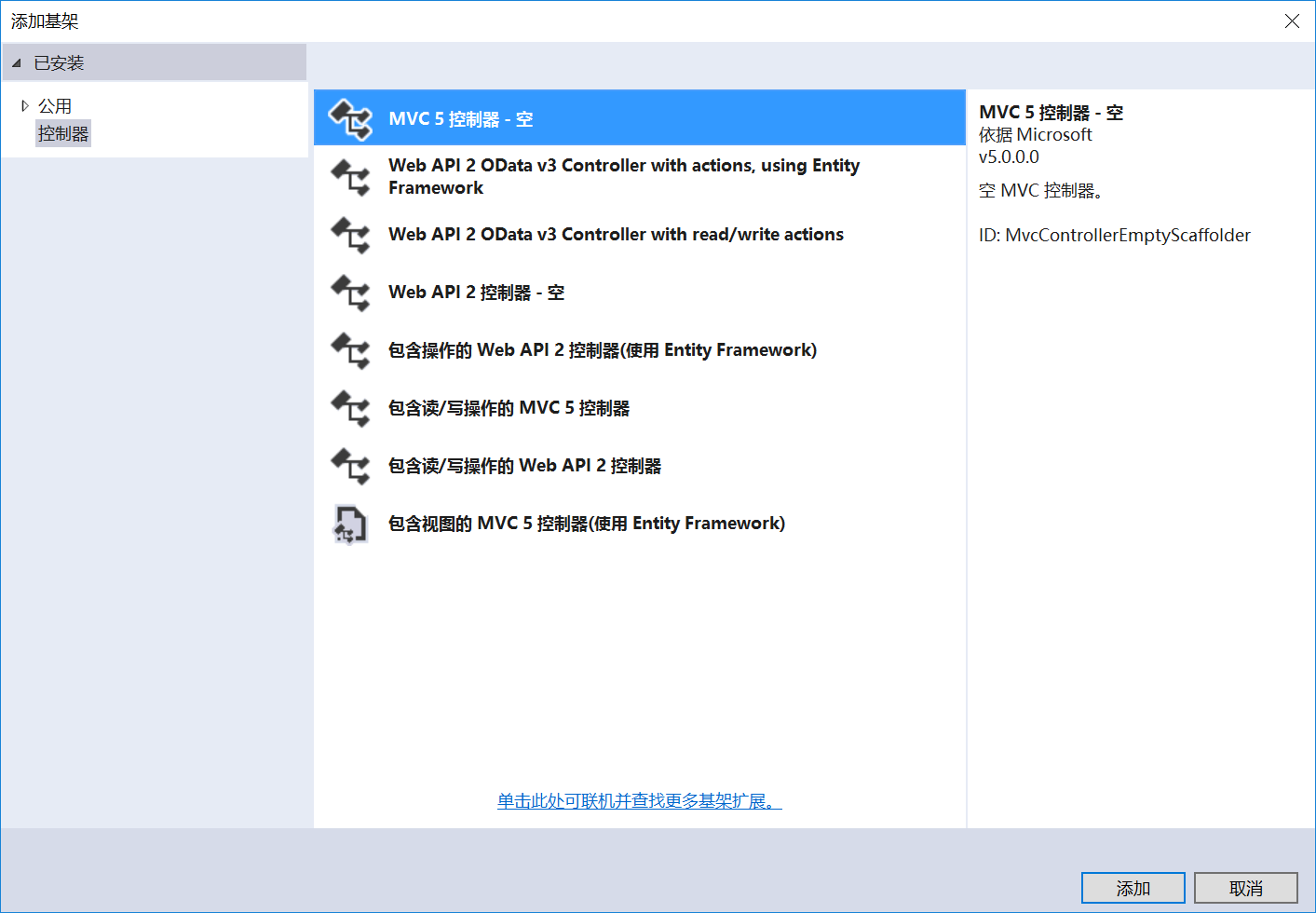
* Step 1.4 选择更改身份验证，弹出对话框中选择“不进行身份验证”，并点击确定。



* 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)

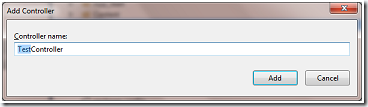


* Step 2.2 选择空 MVC 5 Controller 并点击添加



* 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。

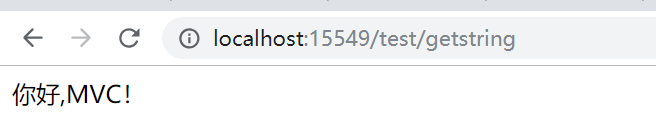
[](http://images0.cnblogs.com/blog/139239/201506/121204001913893.png)

* 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：

|  |
| --- |
| public class TestController : Controller  { public string GetString()  {  return "你好,MVC！";  }  } |

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”。



### 实验1 Q&A

1. TestController 和Test之间的关系是什么？

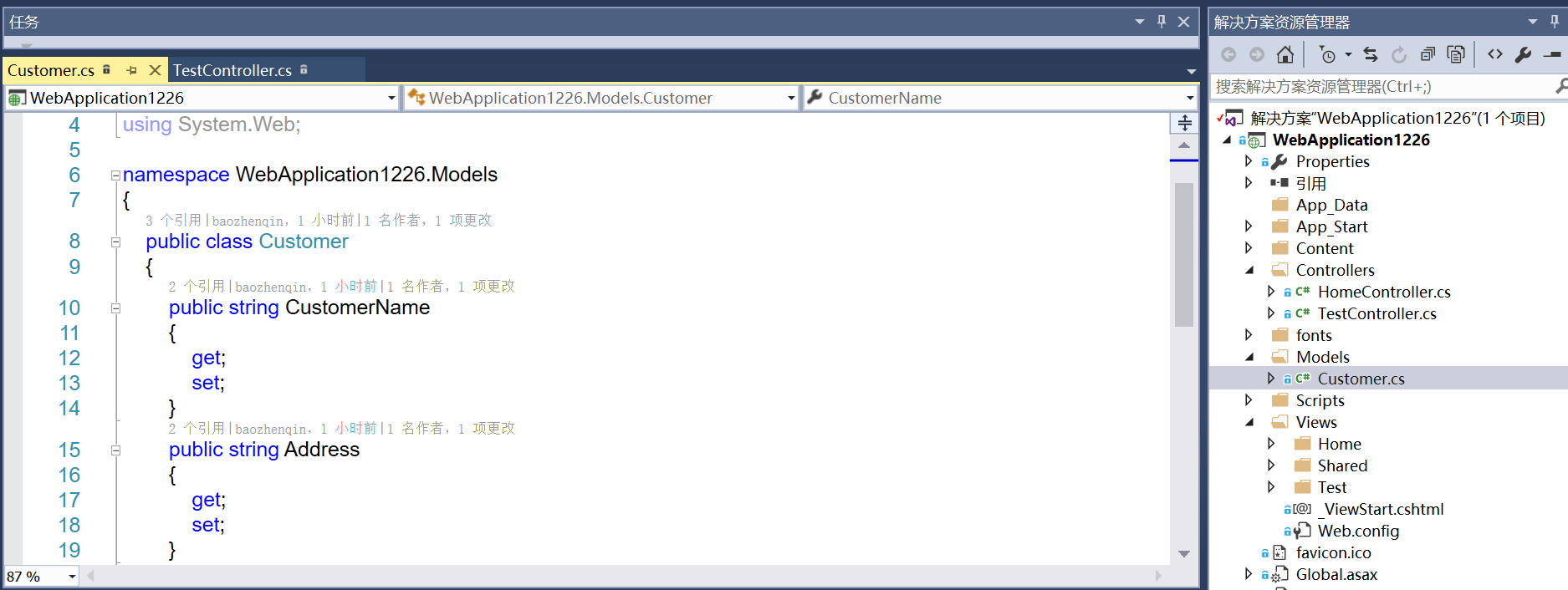
TestController是类名称，而Test是Controller的名称，请注意，当你在URL中输入controller的名称，不需要输入Controller这个单词。

2. Action（行为） 方法是什么？

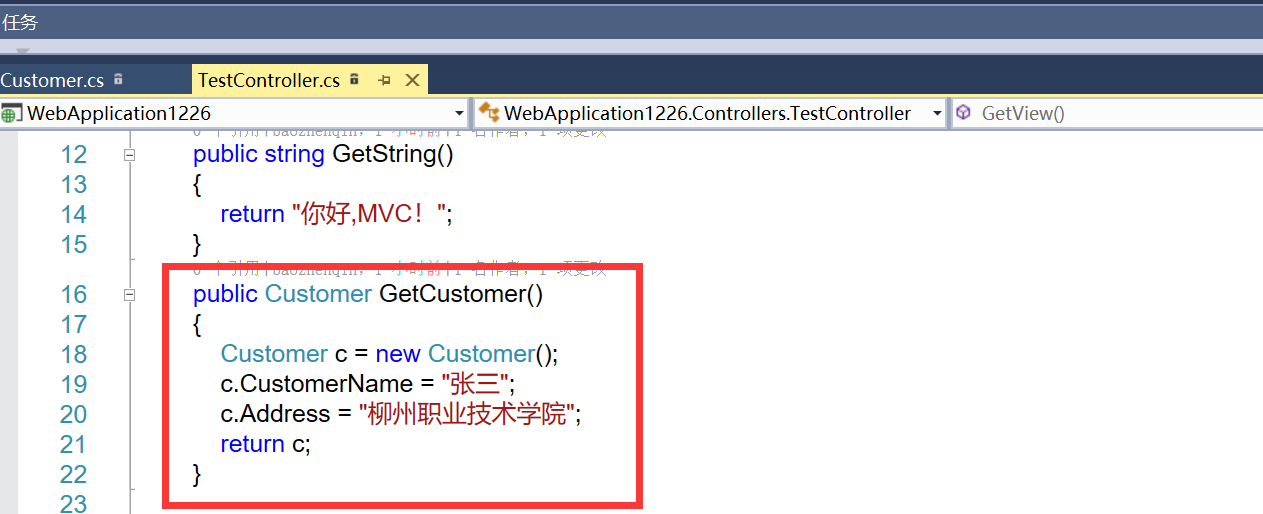
Action 方法 简单的来说就是一个Controller内置的public类型的方法，能够接收并处理用户的请求，上例中，GetString 方法返回了一个字符串类型的响应。

3. 如果从Action 方法中返回对象值会出现什么意外情况？

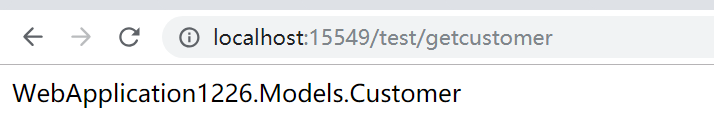
（1）在Models文件夹中添加一个类Customer,并设置两个属性 CustomerName和Address



（2）在Test控制器中增加一个Action



（3）运行程序并定位到http://localhost:15549/test/getcustomer



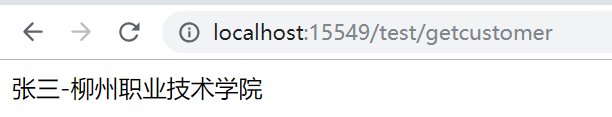
当返回类型如“Customer”这样类似的对象时，将调用ToString()方法，返回“NameSpace.ClassName”形式的类名。

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：

|  |
| --- |
| public override string ToString()  {  return this.CustomerName + "-" + this.Address;  } |

运行结果：



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：

[NonAction]

public string SimpleMethod()

{

return "Hi, I am not action method";

}

当尝试给以上Action 方法发送请求时，会获得以下结果：



## 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2——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。

Step1 –创建新的Action 方法

在TestController中添加新的Action 方法，如下：

public ActionResult GetView()

{

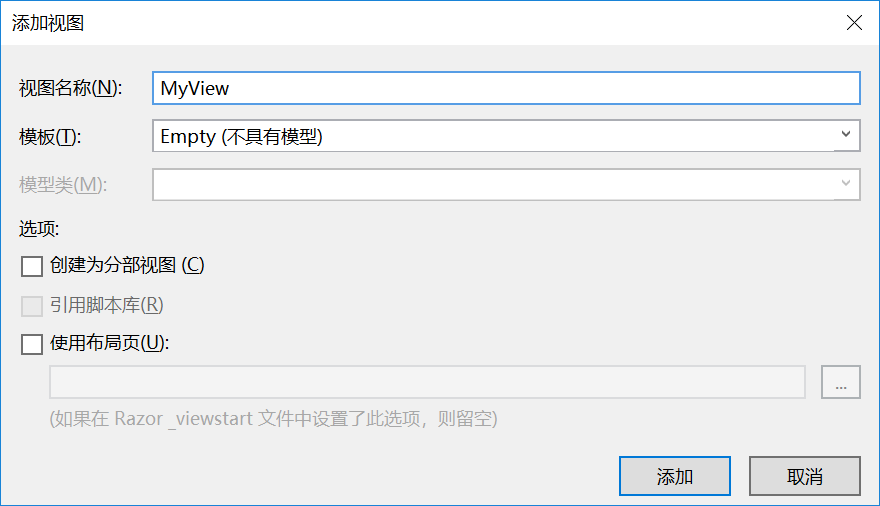
return View("MyView");

}

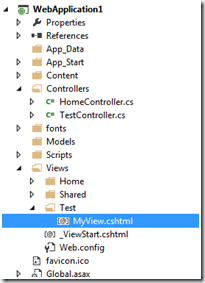
* Step 2 创建View
* Step 2.1 右击上述创建的Action 方法，选择“添加View”



* 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。



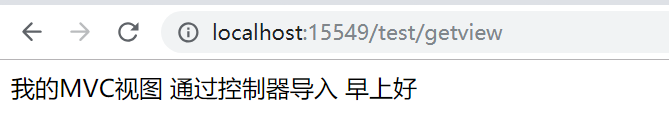
资源管理器重的Views/Test文件夹中会添加一个新的View文件。

[](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：

* Step 4. 运行 按F5键运行应用



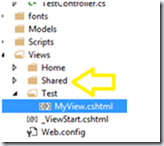
### 实验2Q&A

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。

## 将数据从控制器传递到视图

View从Controller中获取数据实例，该类数据在Model中定义数据类型。

**Model**

**asp.net mvc中的Model负责业务数据定义**

### 实验3 使用ViewData

ViewData相当于数据字典，包含Controller和View之间传递的所有数据。Controller会在该字典中添加新数据项，View从字典中读取数据。

**1. 创建Model 类**

在Model文件夹下新建Employee类，如下。

|  |
| --- |
|  |

**2. 在Controller 中获取Model**

**3. 创建ViewData 并返回View**

在GetView 方法中创建Employee 对象：

|  |
| --- |
|  |

**4. 在View中显示Employee 数据**

打开MyView.cshtml。引用数据

|  |
| --- |
|  |

**5. 测试输出**

按F5运行

|  |
| --- |
|  |

关于实验 3的问与答

1. Razor语法（在cshtml文件中加入C#代码，使用@）

2.写Razor代码带花括号和没有花括号有什么区别？

在实验3中@empName能够使用以下脚本来代替

@{

Response.Write(emp.FirstName);

}

@符号后没有花括号只是简单的显示变量或表达式的值

3. 为什么需要强制转换类型

ViewData每次添加新值，会装箱成Object类型，因此每次都需要拆箱来提取值。

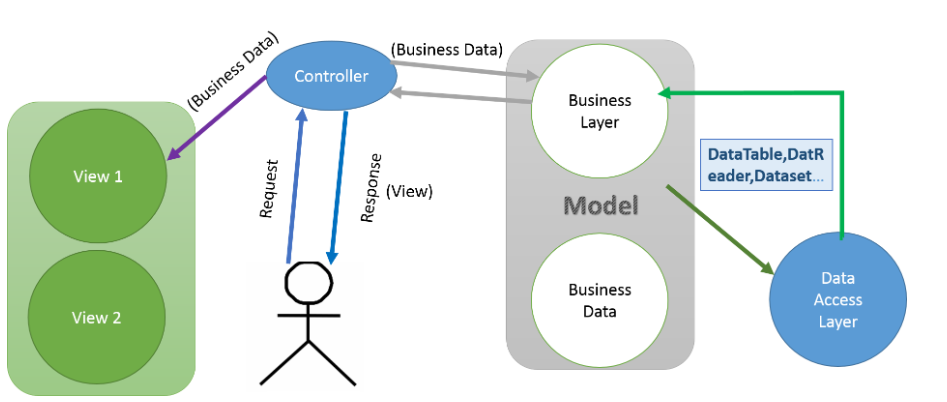
4. 为什么 Employee中的对象实例在Controller创建 ？

在本例只是为了实现实验目的，实际使用中，从数据库或Web服务中获取的。

5. 数据库逻辑，数据访问层，业务层分别指的是什么？

ASP.NET MVC中不包含数据访问层的定义。

业务层是Model的一部分。



### 实验4 使用viewBag

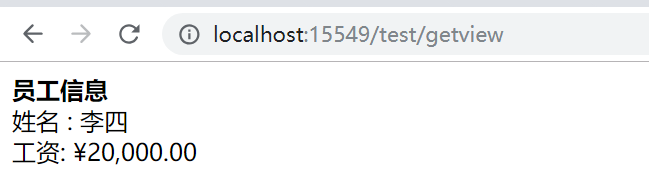
1.在控制器中使用以下语句ViewBag.Employee = emp;替代实验3的ViewData

|  |
| --- |
|  |

2.在view中修改代码以实现数据提取

|  |
| --- |
|  |

3.测试输出结果，与实验3相同



4.ViewData与ViewBag对比：

|  |  |
| --- | --- |
| **ViewData** | **ViewBag** |
| 它是Key/Value字典集合 | 它是[dynamic](http://msdn.microsoft.com/en-us/library/dd264741.aspx)类型对像 |
| 从Asp.net MVC 1 就有了 | ASP.NET MVC3 才有 |
| 基于Asp.net 3.5 framework | 基于Asp.net 4.0与.net framework |
| 快 | 慢 |
| 在ViewPage中查询数据时需要转换合适的类型 | 在ViewPage中查询数据时不需要类型转换 |
| 有一些类型转换代码 | 可读性更好 |

关于实验4

### 实验5 理解强类型视图

1.在控制器中修改返回代码

|  |
| --- |
| return View("MyView",emp); |

2.创建View的强类型

在View的顶部添加以下代码：@model Employee

3. 显示数据

在View内部输入@Model.就会查看到Model类的属性

|  |
| --- |
|  |

关于实验5

1.是否必须设置强类型视图或不使用ViewData和ViewBag？

设置强类型视图是最佳解决方案。

2.是否能将View设置为多个Model使用的强类型？

不可以，实际项目中在一个View中想要显示多个Model时，可以考虑设置viewModel

### 实验6 实现ViewModel

实验5中已经违反了MVC的基本准则。根据MVC，V是View纯UI，不包含任何逻辑层。而我们在实验5中以下两点违反了MVC的体系架构规则。

(1)使用货币显示工资——逻辑层

(2)使用简单的逻辑改变了HTML元素的内容（土豪/屌丝）。——逻辑层

ViewModel 解决方法

ViewModel是ASP.NET MVC应用中隐式声明的层。它是用于维护Model与View之间数据传递的，是View的数据容器。

Model 和 ViewModel 的区别

Model是业务相关数据，是根据业务和数据结构创建的。ViewModel是视图相关的数据。是根据View创建的。

具体的工作原理

Controller 处理用户交互逻辑或简单的判断。处理用户需求

Controller 获取一个或多个Model数据

Controller 决策哪个View最符合用户的请求

Controller 将根据Model数据和View需求创建并且初始化ViewModel对象。

Controller 将ViewModel数据以ViewData或ViewBag或强类型View等对象传递到View中。

Controller 返回View。

View 与 ViewModel 之间是如何关联的？

View将变成ViewModel的强类型View。

Model和 ViewModel 是如何关联的？

Model和ViewModel 是互相独立的，Controller将根据Model对象创建并初始化ViewModel对象。

**实验6—— 实现ViewModel**

**步骤1. 新建文件夹**

在项目中创建新文件夹并命名为ViewModels。

**步骤2. 新建EmployeeViewModel类**

为了达到实验目的，首先列出我们的实验需求：

(1). 使用货币显示数量

(2). 根据薪资显示不同的级别

(3). 当前登录用户也需要在View中显示。

类代码如下

|  |
| --- |
|  |

步骤4.在控制器中创建viewModel对象

|  |
| --- |
|  |

步骤5 在view中引用强视图类型

|  |
| --- |
|  |

步骤6 测试输出

|  |
| --- |
|  |

**关于实验6**

（1）是否意味着，每个Model都有一个ViewModel？

每个View有其对应的ViewModel。

（2）Model与ViewModel之间存在关联是否是好的实现方法？

最好的是Model与ViewModel之间相互独立。

（3）需要每次都创建ViewModel吗？假如View不包含任何呈现逻辑只显示Model数据的情况下还需要创建ViewModel吗？

建议是每次都创建ViewModel，每个View都应该有对应的ViewModel，尽管ViewModel包含与Model中相同的属性。

（4）假定一个View不包含任何呈现逻辑，只显示Model数据，我们不创建ViewModel会发生什么？

无法满足未来的需求，如果未来需要添加新数据，我们需要从头开始创建全新的UI，所以如果我们保持规定，从开始创建ViewModel，就不会发生这种情况。在本实例中，初始阶段的ViewModel将与Model几乎完全相同。

### 实验7 视图和集合

在本实验中，在View中显示Employee列表。

步骤1：创建业务类—以多个构建数据

在model中创建业类EmployeeBusinessLayer类

|  |
| --- |
|  |

**步骤2. 修改EmployeeViewModel 类**

删除UserName属性

|  |
| --- |
|  |

步骤2 创建泛型集合viewmodel

在ViewModels 文件夹下，创建新类并命名为EmployeeListViewModel

|  |
| --- |
|  |

**步骤4. 控制器—构建集合**

|  |
| --- |
|  |

**步骤5修改强类型View的类型**

|  |
| --- |
|  |

**步骤6 显示View中所有的Employee**



测试结果

|  |
| --- |
|  |

关于实验7

（1）是否可以制定强类型View列表?

是的 为什么要新建EmployeeListViewModel单独的类而不直接使用强类型View的列表？

1. 策划未来会出现的呈现逻辑

2. UserName属性。UserName是与employees无关的属性，与完整View相关的属性。

（2）为什么删除EmployeeViewModel 的UserName属性，而不是将它作为EmployeeListViewModel的一部分?

UserName 是相同的，不需要EmployeeViewModel中添加UserName。