http://blog.csdn.net/yisuowushinian/article/details/17646121

因为以前主要是做WebFrom开发，对MVC开发并没有太深入的了解。自从来到创新工场的新团队后，用的技术都是自己以前没有接触过的，比如：MVC 和EF还有就是WCF，压力一直很大。在很多问题都是不清楚的情况下，问周围的人，别人也只是给自己讲一个大概。而且前两天因为问了一个比较细的问题，还被别人的一句话打击。“我只能告诉你方法，你还指望我手把手的交给你呀，不会你得自己学呀。。。”。没办法只能自己找时间在下面找一些资料学习。

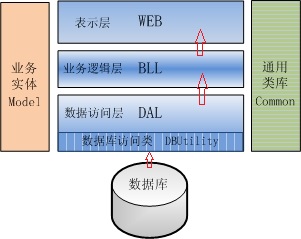
在初步了解MVC后，发现很多人对于MVC和三层架构开发概念上会有很大的混淆，所以把这两天的学习笔记整理一下，分享给自己的同学们。同时也做一个小Demo，让没有接触过MVC开发的同学，能对MVC有一个简单的了解。

**一，MVC和三层架构的区别**

**①什么是三层架构？**

在学校的时候，和同学或者老师一起讨论MVC的时候，别人可能会说，“不就是三层架构嘛！实体层（Model）,用来创建对象的实体；业务逻辑层（BLL），用来处理复杂的数据间的关系或者是业务间的关系；数据库访问层（DAL），用来用来访问数据库的；当然还会有，View(视图层)，用来展示数据；”其实自己虽然知道不是这样，但是自己也仅仅是了解一点点，也解释不清楚，所以也就算了。（具体的三层架构之间的操作的关系可以看我以前的博客[【ASP.NET开发】.NET三层架构简单解析](http://www.cnblogs.com/qq731109249/archive/2013/01/16/2863201.html)）。

它们之间互相操作的关系，基本的示意图如下：



还记的暑假在郑州找实习工作的时候，被别人问过一个n层架构的概念，当时就懵了，尼玛的压根就没听说过啊！不过后来查询资料知道，所谓的n曾架构就是根据系统的需要把业务逻辑层（BLL）或者数据库访问层（DAL）再抽象成几个层次（具体的就是再抽象成类），便于逻辑的处理和代码模块的维护。其基本原理还是基于三层架构。

**②什么是MVC呢？**

MVC其实是软件架构的一种模式，也就是我们经常说的设计模式。其中主要包括三个模块，就是如MVC名字所显示的那样：模型（Model），视图（View），控制器（Controller）；

其中这里的模型（Model）和视图（View ）是完全区别于三层架构中的模型（Model）和视图（View）的。

**1）MVC中的模型（Model）指的是数据模型，**用于封装与应用程序的业务逻辑相关的数据，除此之外还可以封装数据的处理方法（相当于业务逻辑）。这是完全区别于三层架构的模型层（Model）的。

MVC中模型（Model）的特点：

①有对数据直接访问的权利，如：对数据库的访问；

②模型（Model）“不依赖”视图（View）和控制器（Controller），即模型（Model）不关心它会被如何显示或者如何被操作；

③模型（Model）中数据的变化一般会通过一种刷新机制被“公布”；

④为了实现③中的“机制”用于监视此模型的视图必须事先在此模型上注册。从而，视图可以了解在数据模型上发生的改变。

**2）视图（View），**这里的视图基本跟三层中的视图一样，都是为了显示数据，没有程序上的逻辑。为了实现视图上数据的刷新，视图（View）需要访问它监视的模型（Model），所以应该事先在被它监视的数据那里进行注册。

**3）控制器（Controller），**这个概念是在三层中不存在的概念。它主要起到不同层面的组织作用，用于控制应用程序的流程。主要处理事件并作出相应。“事件”主要包括：用户的行为和数据的改变。

以上就是关于三层架构和MVC在概念上的区别。

**二，WebForm网站和MVC网站运行机制的区别**

**①WebForm网站的运行机制**

比如说我们现在要访问一个WebForm站点：[www.google.com.hk/Default.aspx](http://www.google.com.hk/Default.aspx)(仅仅是示例)。我们的浏览器和服务器都是做了哪些动作呢？

**1）首先浏览器会向目的服务器发送请求报文。**

配置过IIS的都知道，网站挂载在服务器上，我们是通过访问虚拟目录的方式访问网站的。这时候目的主机的IIS接收的是访问该虚拟目录下Default.aspx文件的请求；（当然这也是一个非常复杂的过程，包括请求DNS服务器，找到目的主机IP，根据IP地址访问目的主机。复杂的网络过程就不叙述，有兴趣的自己找资料学习）；

**2）服务器端的IIS软件接收到请求后，把请求交给.NET FramWork进行处理；**

**3）.NET FramWork会创建Default\_aspx类的对象，也就是我们所说的页面对象。（**在WebFrom网站创建完，并且编译后Default.aspx会被编译成Default\_aspx类）

到现在的整个过程都还是Http请求，IIS的内部机制会去实现一个IHttphandler的接口，其中该接口实现一个ProcessRequestfang方法

MSDN是这样解释的



该ProcessRequest（）方法会去调用对应页面的Page\_Load（） 方法

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

2 {

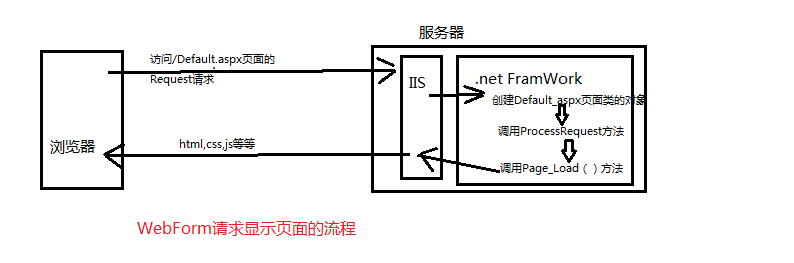
3 //处理的业务逻辑或者是访问数据库的代码

4 //要输出的Html或者其它内容

5 }

**4）返回给浏览器（包括Html，CSS，Js等等）**

流程示意图如下：



**②MVC网站的运行机制**

 还比如说我们现在要访问一个MVC站点：[www.google.com.hk/FirstPage/Default](http://www.google.com.hk/Firstpage/Default)(仅仅是示例)。我们的浏览器和服务器又做了哪些动作呢？

**1)浏览器向服务器发送Request请求报文（FirstPage/Default）**

**2)服务器端的IIS相应Request请求**

**3）.NET FramWork根据路由配置，解析URL，并创建FirstPage类的对象，并调用相应的Default方法**

1 public ActionResult Default()

2 {

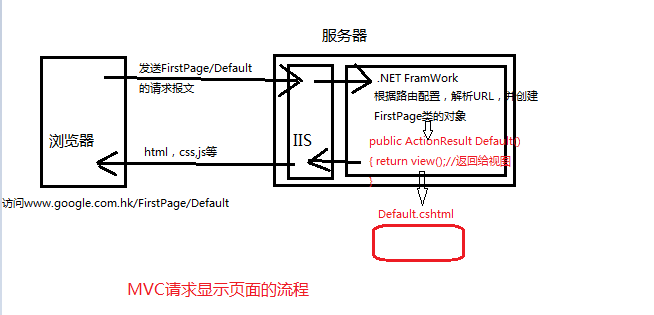
3

4 return View();//返回给视图

5 }

**4)然后会访问视图文件夹下的Default.cshtml，返回给浏览器（其中包括html,css,js等等）**

流程的示意图如下：



这只是一个比较简单的运行过程。其实在这过程中发生了很多事情，比如说：执行Global.asax中的Application\_Start()方法来完成一些初始化的工作等等，会在以后的文章中继续解析。

以上就是WebForm网站和MVC网站运行机制的区别。

**那么到底使用MVC的优点比WebForm到底有哪些优点呢？**

①最重要的就是.NET程序员在开发的时候再也不会使用那些被很多人诟病的微软封装的控件了。

②MVC设计模式降低了模型（Model,业务和数据）和视图的耦合关系。包括我们在开发WebForm网站使用三层架构的思想也是为了降低数据和视图的耦合等；

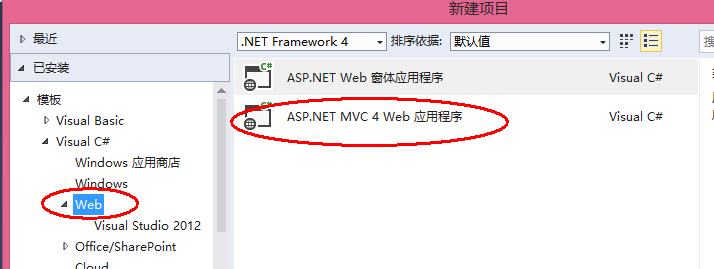
③可以复用视图，也就是说同样的数据可以使用不同的视图以不同的图标展示出来。

 -------------------------------------------------------理论到此结束---------------------------------------------------------------

常用的WebForm开发方式就不举例了，我下边主要演示一个基本的MVC程序的创建和运行过程。

要创建MVC程序VS肯定是必不可少的，我的开发环境是VS2013，大家请酌情考虑自己的VS版本。

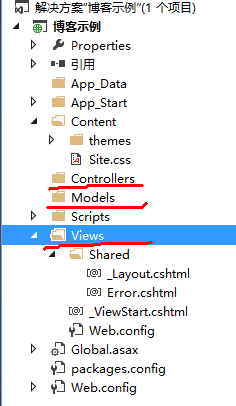
**①首先的打开“文件”→“新建项目”，在左侧的项目栏选择“Visual C#”→“Web”,在右边就可以看到有“ASP.NET Web窗体应用程序”和“ASP.NET MVC 4 Web应用程序”如下图：**



**②因为要创建MVC程序就选择点击第二个选项然后会看到项目模板，默认的是选择“Internet应用程序”，如果选择默认的话，就会默认的创建一个具有基本功能的站点。我们这里选择“基本”，VS只会创建包含基本框架简的模板**



**③创建完以后我们会在解决方案下看到代码的树状结构，其中包含Models,Views和Controllers三个文件夹**



**④我们首先创建实体对象Model，在Models文件夹下创建一个Child类，并声明属性和初始化**

[复制代码](javascript:void(0);)

1 public class Child

2 {

3 //编号

4 private int id;

5

6

7 public int Id

8 {

9 get { return id; }

10 set { id = value; }

11 }

12

13 //姓名

14 private string strName;

15

16 public string StrName

17 {

18 get { return strName; }

19 set { strName = value; }

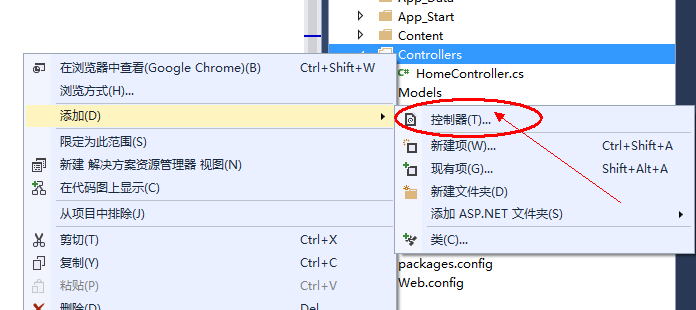
20 }

21

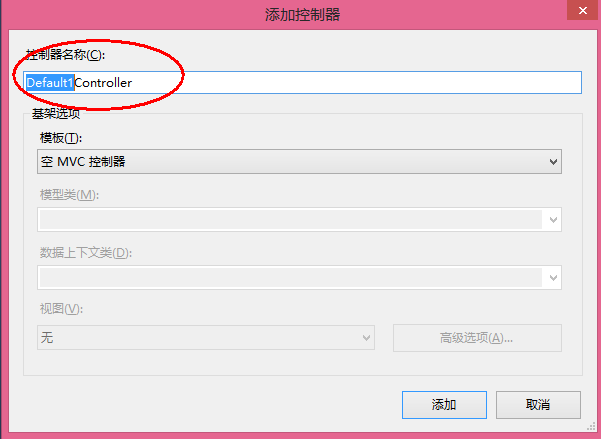
22 }

[复制代码](javascript:void(0);)

**⑤在Controllers文件夹上右键，“添加”，会看到有“控制器”的选项，如下图：**



点击"控制器"后会出现一个“添加控制器”窗体，在控制器名称中”Default1Controller“中”Default1“是默认选中的，我们修改其名称为”HomeController“，



注意：这里的”Default1Controller“中的”\*Controller“是必须保留的，主要是因为MVC框架有一个”约定大于配置“的规则

单击”添加“添加控制器，默认生成的控制器代码，如下：

[复制代码](javascript:void(0);)

1 //默认生成的控制器代码

2 public class HomeController : Controller

3 {

4 //

5 // GET: /Home/

6

7 //包含一个Action方法

8 public ActionResult Index()

9 {

10 //返回给视图

11 return View();

12 }

13

14 }

[复制代码](javascript:void(0);)

**⑥然后我们创建一个集合来初始化在Models中创建的Child类的属性值**

[复制代码](javascript:void(0);)

1 #region 初始化数据集合 +List InitData()

2 /// <summary>

3 /// 初始化数据集合

4 /// </summary>

5 /// <returns></returns>

6 public List<Models.Child> InitData()

7 {

8

9 List<Models.Child> list = new List<Models.Child>()

10 {

11 new Child(){Id=1,StrName="你好啊！"},

12 new Child(){Id=2,StrName="不好啊！"}

13 };

14

15 return list;

16 }

17

18 #endregion

[复制代码](javascript:void(0);)

然后开始编写Action方法，即”ActionResult Index()“方法，

[复制代码](javascript:void(0);)

1 /// <summary>

2 /// Action方法（相当于MVC设计模式的Model）

3 /// </summary>

4 /// <returns></returns>

5

6 public ActionResult Index()

7 {

8 //可以处理当前业务（你比如读取数据库，判断等）

9 StringBuilder strBuilder = new StringBuilder();

10 //创建数据集合，获取数据

11 List<Models.Child> list = InitData();

12 //遍历集合获取生成的Html代码

13 list.ForEach(d =>

14 {

15 strBuilder.AppendLine("<div>" + d.Id.ToString() + "</div>");

16 });

17

18 //使用ViewBag传输数据给同名的Indexcshtml视图

19 //ViewBag是一个dynamic类型的集合，可以动态添加任意类型的任意名称和属性

20 ViewBag.HtmlStrBuilder = strBuilder.ToString();

21 //加载同名视图Index.cshtml

22 return View();

23 }

[复制代码](javascript:void(0);)

在其中我们使用ViewBag把数据传输给同名的Views文件夹下的同名视图。

**⑦创建视图接收数据，上面代码的Index方法上右键，会出现”添加视图“，如下图**



 之后会在Views文件夹下，生成一个Home文件夹，Home文件夹下面会有一个Index.cshtml文件(这个就是视图文件)

 ⑧视图接收数据

[复制代码](javascript:void(0);)

1 <html>

2 <head>

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

4 <title>Index</title>

5 </head>

6 <body>

7 <div>

8 <!---------相当于把Action方法中保存的内容输出--------->

9 @Html.Raw( @ViewBag.HtmlStrBuilder)

10 </div>

11 </body>

12 </html>

[复制代码](javascript:void(0);)

到这里基本上就完成了一个MVC程序的创建，我们运行程序，结果如下：

  
注意：因为MVC的运行机制跟WebForm不一样，所以浏览的方式也不一样。我们可以直接在Views文件夹上右键，选择在浏览器中查看就可以预览到结果。

-----------------------------------------分割线-------------------------------------------

到这里，关于ASP.NET WebForm和Mvc开发的区别和基本MVC程序的创建过程已经完成。

下次我会做的是使用EF（Entity FramWork），来操作数据库（抛开WebForm中的ADO.NET），实现一个MVC小网站程序的创建。请大家继续关注！