[**解析ASP.NET Mvc开发之查询数据实例**](http://blog.csdn.net/yisuowushinian/article/details/17743407)

http://blog.csdn.net/yisuowushinian/article/details/17743407

2014-01-02 01:27 27075人阅读 [评论](http://blog.csdn.net/yisuowushinian/article/details/17743407#comments)(5) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/yisuowushinian/article/details/17743407#report)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

ASP.NET（14） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

版权声明：本文为博主原创文章，未经博主允许不得转载。

**﻿﻿目录：**

**1）**[**从明源动力到创新工场这一路走来**](http://blog.csdn.net/yisuowushinian/article/details/17649631)

**2）**[**解析ASP.NET WebForm和Mvc开发的区别**](http://blog.csdn.net/yisuowushinian/article/details/17646121)

---------------------------------------------------------------------------------------------------------------------------------------------------------

对于.NET平台上开发WebForm项目，程序员操作数据的方法主要是通过使用ADO[**.NET**](http://lib.csdn.net/base/dotnet)。而我们MVC操作**[数据库](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "_blank)**呢？ 与ADO[**.Net**](http://lib.csdn.net/base/dotnet)相比又有怎样的优势呢？

**一，大家都在谈的EF到底是什么？**

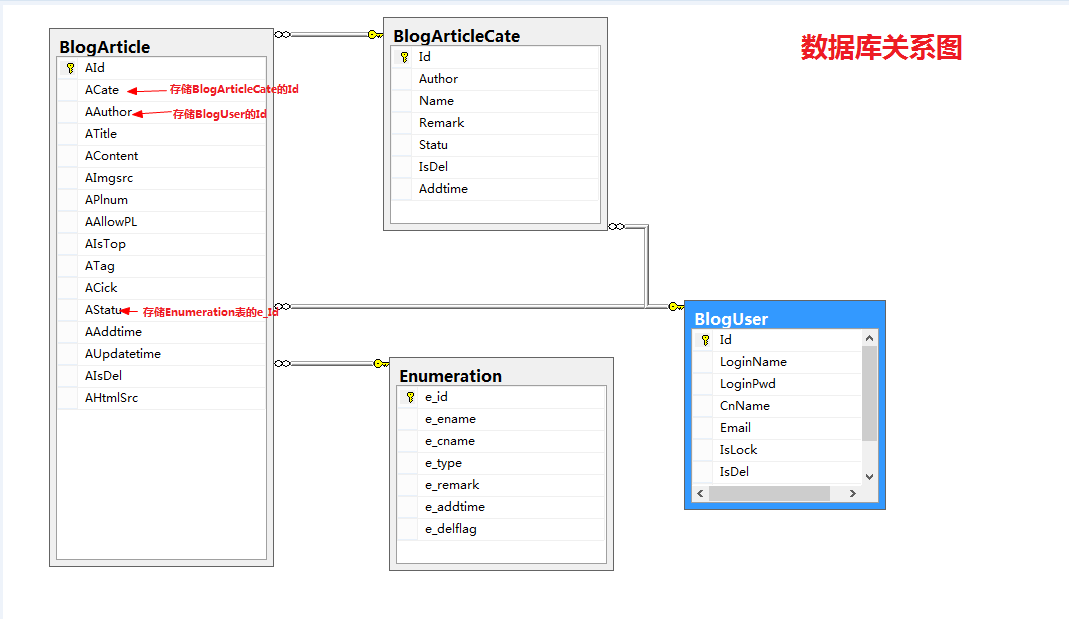
EF ，全称Entity FramWork。就是微软以ADO.NET为基础发展的所谓ORM（对象关系映射框架，或者说是数据持久化框架）。

简单的来说就是根据实体对象操作数据表中数据的一种面向对象的操作框架，具体的底层也是调用ADO.NET。具体概念可以参考维基百科（[点击我转到](http://zh.wikipedia.org/wiki/Entity_Framework" \t "_blank)）。

下面我们就来演示怎么使用EF来操作数据库：

[点击这里](http://pan.baidu.com/s/1jG4vd34)可以下载我创建数据库的脚本文件，当然你也可以是使用自己的数据库。

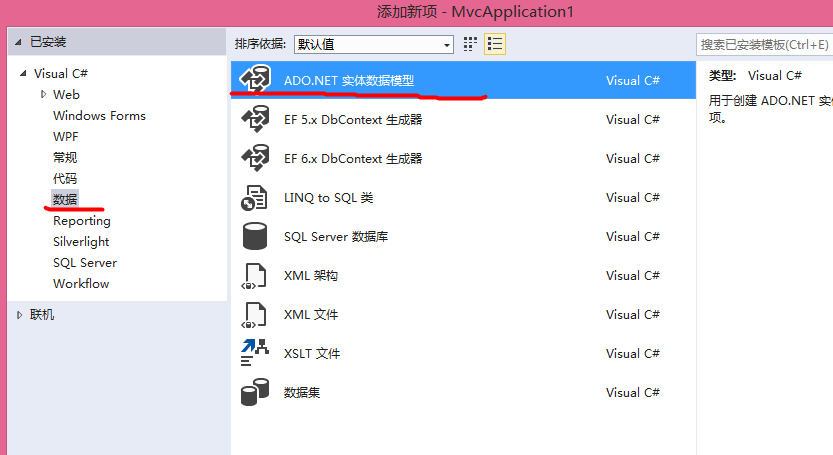
在数据库关系图中，表之间的关系是这样的：



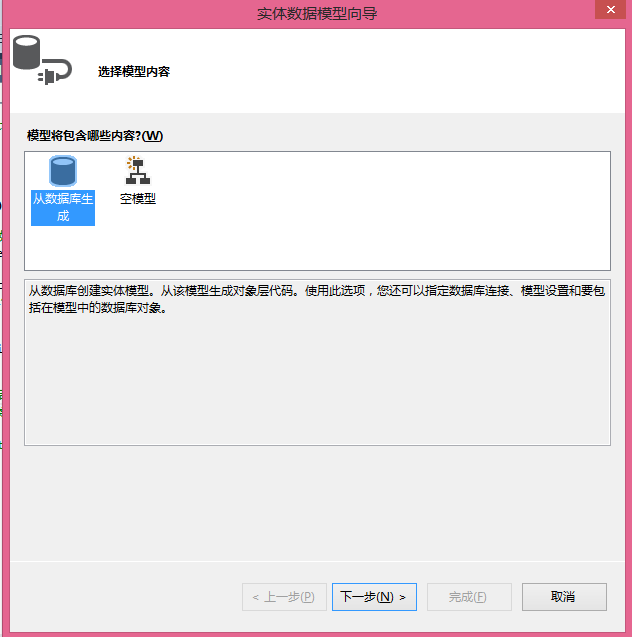
**1）怎么创建实体对象模型文件**

**①创建基于ADO.NET的实体数据模型**

我这里在我的Models文件夹（可以在任意地方哦）上“右键”→“新建项”，然后点击进入，在左侧的模板中，选择"数据"，在右侧就会看到“ADO.NET实体数据模型”的选项，如下图：



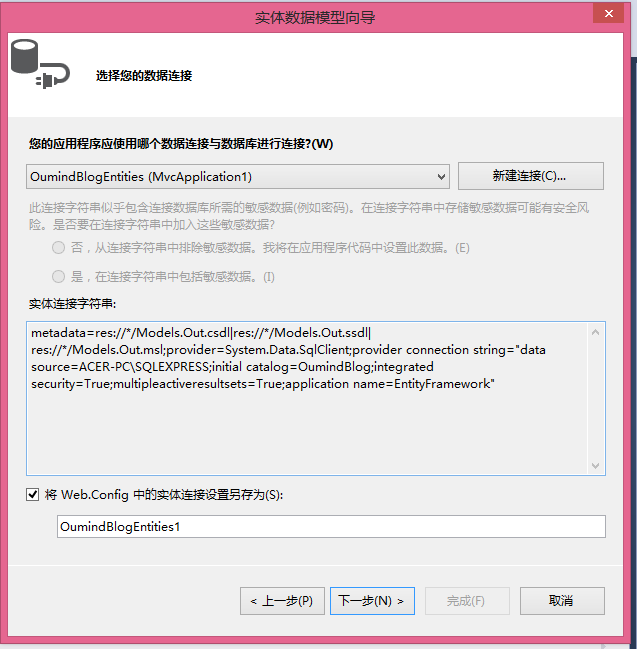
**②点击创建，进入”实体数据模型向导“，这里我们可以选择，要怎样创建实体数据模型**



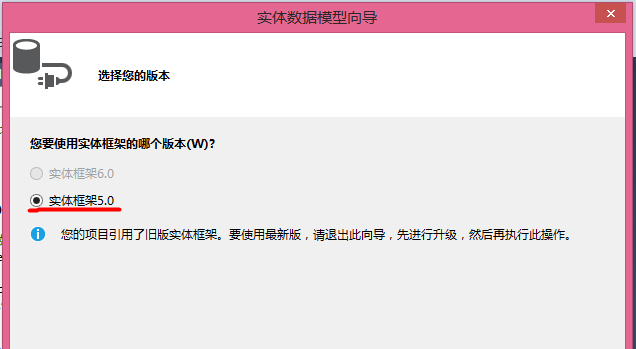
注意：这里你可以分别选择一种模型，在下边的框中看到这两种模型的区别，这里不多做解释；

因为我们是从已经有的数据库创建实体模型，所以选择第一种”从数据库生成“

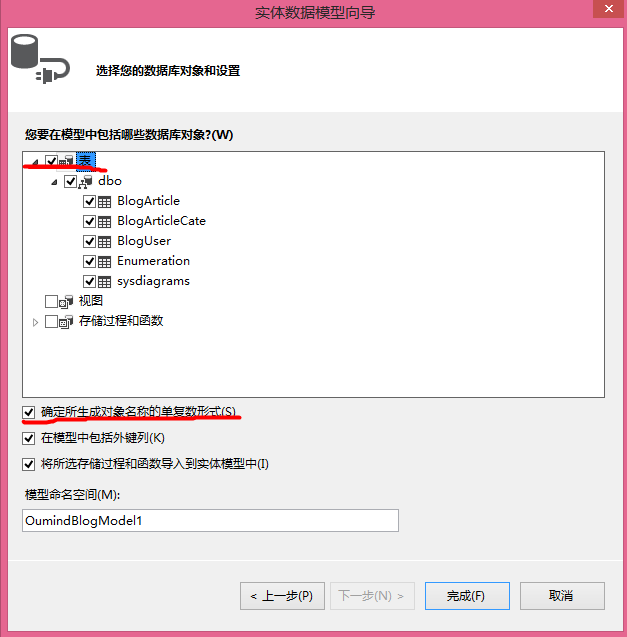
**③点击下一步，进入连接数据库的向导，通过新建连接，可以连接到自己的SQL Server服务器，然后选择我们所要连接的数据库，同时会生成实体连接字符串，如下图：**



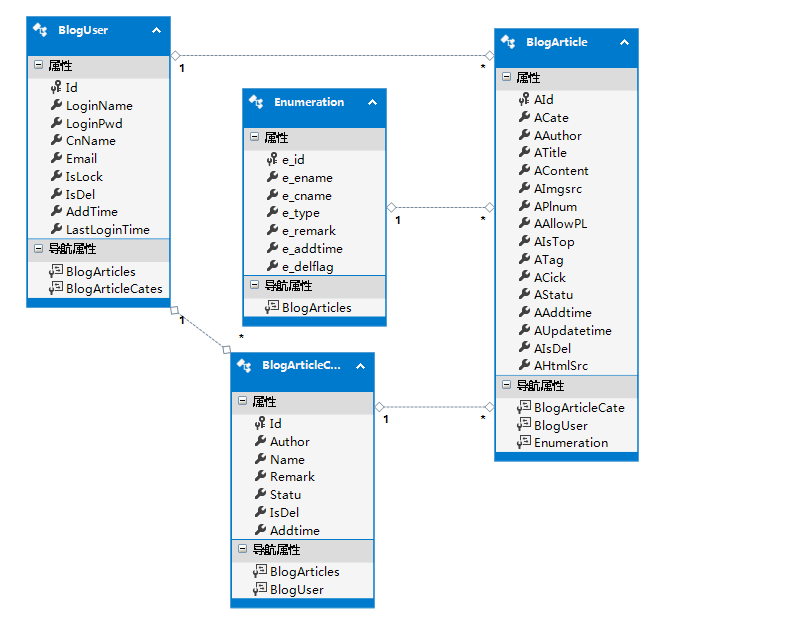
**④点击下一步，选择”实体框架版本“**



**⑤下一步，选择需要在模型中包含的数据库对象，选择”确定所生成对象的单复数形式“，其它保持默认，点击完成。如下图：**



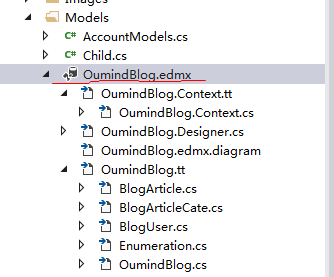
这时会在我们的Models文件夹下边生成一个edmx文件，同时**EF框架的设计器帮我们生成了根据数据表关系生成的实体类的关系图**，如下图：



注意：这里是不是发现这个表跟我们在数据库关系图中看到的图有点相似啊？是的，很相似，但是所表达的意义是不一样的。数据库视图所展示的是数据表之间的关系，而这里展示的EF帮我们根据数据表关系生成的实体类的关系；

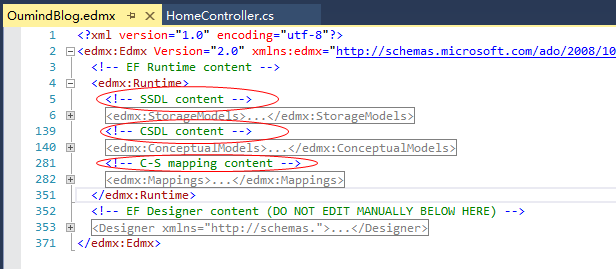
**2)生成的edmx文件到底是什么？**

edmx文件的代码树如下：



**①怎么会是XML文件？**

我们通过”右键“→”打开方式“→在”XML编辑器“中打开，可以发现这个edmx文件是一个标准的XML文件，其中主要有三大块，**描述的就是我们的实体对象和数据库映射的关系**。如下图：



**②.tt文件下怎么生成了.cs文件？**

**数据库上下文类。**

　　在代码树中我们很容易的发现，在.tt文件下边生成了.cs类文件。”OumindBlog.Context.cs“文件的代码和作用如下：

**[csharp]** [view plain](http://blog.csdn.net/yisuowushinian/article/details/17743407) [copy](http://blog.csdn.net/yisuowushinian/article/details/17743407)

[在CODE上查看代码片](https://code.csdn.net/snippets/138186)

1. //继承自DbContext类，
2. //数据上下文类，用于操作数据库。负责维护实体状态，以及根据实体对象包装类的状态属性，生成不同的SQL语句执行
3. **public** partial **class** OumindBlogEntities : DbContext
4. {
5. **public** OumindBlogEntities()
6. : **base**("name=OumindBlogEntities")
7. {
8. }
10. **protected** **override** **void** OnModelCreating(DbModelBuilder modelBuilder)
11. {
12. **throw** **new** UnintentionalCodeFirstException();
13. }
15. **public** DbSet<BlogArticle> BlogArticles { **get**; **set**; }
16. **public** DbSet<BlogArticleCate> BlogArticleCates { **get**; **set**; }
17. **public** DbSet<BlogUser> BlogUsers { **get**; **set**; }
18. **public** DbSet<Enumeration> Enumerations { **get**; **set**; }
19. }

**实体类。**

　　而另一个.tt文件生成的类是做什么用呢？我们打开一个看代码：

**[csharp]** [view plain](http://blog.csdn.net/yisuowushinian/article/details/17743407) [copy](http://blog.csdn.net/yisuowushinian/article/details/17743407)

[在CODE上查看代码片](https://code.csdn.net/snippets/138186)

1. //EF按照实体模型生成的实体类
2. **public** partial **class** BlogArticleCate
3. {
4. **public** BlogArticleCate()
5. {
6. **this**.BlogArticles = **new** HashSet<BlogArticle>();
7. }
9. **public** **int** Id { **get**; **set**; }
10. **public** **int** Author { **get**; **set**; }
11. **public** **string** Name { **get**; **set**; }
12. **public** **string** Remark { **get**; **set**; }
13. **public** **int** Statu { **get**; **set**; }
14. **public** **bool** IsDel { **get**; **set**; }
15. **public** System.DateTime Addtime { **get**; **set**; }
17. //生成的外键属性
18. **public** **virtual** ICollection<BlogArticle> BlogArticles { **get**; **set**; }
19. **public** **virtual** BlogUser BlogUser { **get**; **set**; }
20. }

**很容易的发现这些字段和我们数据库中的字段是对应的，这就是EF按照实体模型生成的实体类，而且在还成了外键属性。**

 ---------------------------------------------------预备工作到此结束--------------------------------------------------

二，使用EF操作数据库

在上面”OumindBlog.Context.cs“中生成继承DbContext的OumindBlogEntities 的类，来维护实体状态，并且操作数据库，所以我们首先要创建该类的对象，并且操作数据的代码如下：

**[csharp]** [view plain](http://blog.csdn.net/yisuowushinian/article/details/17743407) [copy](http://blog.csdn.net/yisuowushinian/article/details/17743407)

[在CODE上查看代码片](https://code.csdn.net/snippets/138186)

1. //创建数据库上下文类的对象
2. OumindBlogEntities db = **new** OumindBlogEntities();
4. #region 查询文章列表+ActionResult Article()
5. /// <summary>
6. /// 查询文章列表
7. /// </summary>
8. /// <returns></returns>
9. **public** ActionResult Article()
10. {
11. //通过db对象获取文章列表
12. db.BlogArticles.Where(p => p.AIsDel == **false**);//使用Lamabda表达式来获取为被删除的文章
14. //使用Lamabda表达式来获取数据
15. //返回一个List<T>的对象来存储文章列表
16. List < Models.BlogArticle > list= db.BlogArticles.Where(p => p.AIsDel == **false**).ToList();

19. //也可以使用Linq来获取数据
20. List<Models.BlogArticle> list1 = (from p **in** db.BlogArticles where p.AIsDel == **false** select p).ToList();
22. //使用ViewData来传递list对象
23. ViewData["DataList"] = list;
25. **return** View();
26. }
27. #endregion

接着我们为Article创建视图，并且接收数据，  
因为我们需要使用BlogArticle对象来显示数据，所以应该首先导入命名空间

<!------------首先导入命名空间------------->>

@using MvcApplication1.Models;

 然后显示数据的代码是：

**[html]** [view plain](http://blog.csdn.net/yisuowushinian/article/details/17743407) [copy](http://blog.csdn.net/yisuowushinian/article/details/17743407)

[在CODE上查看代码片](https://code.csdn.net/snippets/138186)

1. <!------------获取数据并显示html------------->**>**
2. **<div>**
3. **<table** id="tbList"**>**
4. **<tr>**
5. **<th>**id**</th>**
6. **<th>**标题**</th>**
7. **<th>**分类**</th>**
8. **<th>**状态**</th>**
9. **<th>**时间**</th>**
10. **<th>**操作**</th>**
11. **</tr>**
12. <!--遍历 Action方法 设置给 ViewData 的集合数据，生成HTML代码-->
13. @foreach (BlogArticle a in ViewData["DataList"] as List**<BlogArticle>**)
14. {
15. **<tr>**
16. **<td>**@a.AId**</td>**
17. **<td>**@a.ATitle**</td>**
18. **<td>**@a.BlogArticleCate.Name**</td>**
19. **<td>**@a.Enumeration.e\_cname**</td>**
20. **<td>**@a.AAddtime**</td>**
22. **</tr>**
23. }
24. **</table>**
25. **</div>**

运行的结果如下：



**三，总结**

**1）EF框架根据我们的数据库中数据模型生成实体数据模型；**

**2）实体数据模型为一个edmx文件，且该文件为一个标准的XML文件，主要描述实体对象和数据库映射关系；**

**3）.tt文件为我们生成了数据库上下文类（用于操作数据库）和实体类（表示实体对象和外键属性关系）；**

**4）实体对象可以使用Lamabda表达式或者Linq来查询需要的数据，并且使用一个List对象来存储数据；**

**5）易于理解代码，在实际的操作中不需要像ADO.net一样创造大量的数据库访问层；**