# 1:Code First 使用步骤

1:创建类

public class Student  
 {  
 public int StudentID { get; set; }  
 public string LastName { get; set; }  
 public string FirstMidName { get; set; }  
 public DateTime EnrollmentDate { get; set; }  
 public virtual ICollection<Enrollment> Enrollments { get; set; }  
 }

2：创建实体（重点是默认公约）

public class SchoolContext:DbContext  
 {  
 private readonly static string CONNECTION\_STRING = "name=WlfSys\_EFCF\_ConnString";  
  
  
 public DbSet<Student> Students { get; set; }  
 public DbSet<Enrollment> Enrollments { get; set; }  
 public DbSet<Course> Courses { get; set; }  
  
 public SchoolContext()  
 : base(CONNECTION\_STRING)//不写这个 默认的就是SchoolContext  
 {  
  
 }  
  
 protected override void OnModelCreating(DbModelBuilder modelBuilder)  
 {  
 modelBuilder.Conventions.Remove<PluralizingTableNameConvention>();//移除复数表名的契约  
  
 modelBuilder.Conventions.Remove<IncludeMetadataConvention>();//防止黑幕交易 要不然每次都要访问 EdmMetadata这个表  
  
  
  
  
 /\*  
   
 可以删除的公约有：  
Namespace:System.Data.Entity.ModelConfiguration.Conventions.Edm  
• AssociationInverseDiscoveryConvention   
寻找导航上互相引用的类的属性，并将它们配置为逆属性的相同的关系。  
• ComplexTypeDiscoveryConvention   
寻找有没有主键的类型，并将它们配置为复杂类型。  
• DeclaredPropertyOrderingConvention   
确保每个实体的主要关键属性优先于其他属性。  
• ForeignKeyAssociationMultiplicityConvention   
配置是必需的还是可选的关系基于为空性外键属性，如果包含在类定义中。  
• IdKeyDiscoveryConvention   
查找名为 Id 或 <TypeName> Id 的属性，并将他们配置作为主键。  
• NavigationPropertyNameForeignKeyDiscoveryConvention   
使用外键关系，使用 <NavigationProperty> <PrimaryKeyProperty> 模式作为属性的外观。  
• OneToManyCascadeDeleteConvention   
交换机上层叠删除，所需的关系。  
• OneToOneConstraintIntroductionConvention   
将配置为一个： 一个关系的外键的主键。  
• PluralizingEntitySetNameConvention   
配置为多元化的类型名称的实体数据模型中的实体集的名称。  
• PrimaryKeyNameForeignKeyDiscoveryConvention   
使用外键关系，使用 <PrimaryKeyProperty> 模式作为属性的外观。  
• PropertyMaxLengthConvention   
配置所有的字符串和字节 [] 属性，默认情况下具有最大长度。  
• StoreGeneratedIdentityKeyConvention   
配置默认情况下将标识所有整数的主键。  
• TypeNameForeignKeyDiscoveryConvention   
使用外键关系，使用 <PrincipalTypeName> <PrimaryKeyProperty> 模式作为属性的外观。  
  
   
 \*/  
  
 }  
  
 }  
}

3：配置数据库连接字符串，默认的名称和实体名称相同

<connectionStrings>  
 <add name="WlfSys\_EFCF\_ConnString"  
 providerName="System.Data.SqlClient" connectionString="Data Source=.;Initial Catalog=WLFSchool;Integrated Security=True;Pooling=False" />  
 </connectionStrings>

4 数据库初始化

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web;  
using System.Data.Entity;//记得引用命名空间  
using ContosoUniversity.Models;  
  
  
namespace ContosoUniversity.DAL  
{  
 public class SchoolInitializer : DropCreateDatabaseIfModelChanges<SchoolContext>  
 {  
   
  
 protected override void Seed(SchoolContext context)  
 {  
 var students = new List<Student>  
 {  
 new Student { FirstMidName = "Carson", LastName = "Alexander", EnrollmentDate = DateTime.Parse("2005-09-01") },  
 new Student { FirstMidName = "Meredith", LastName = "Alonso", EnrollmentDate = DateTime.Parse("2002-09-01") },  
 new Student { FirstMidName = "Arturo", LastName = "Anand", EnrollmentDate = DateTime.Parse("2003-09-01") },  
 new Student { FirstMidName = "Gytis", LastName = "Barzdukas", EnrollmentDate = DateTime.Parse("2002-09-01") },  
 new Student { FirstMidName = "Yan", LastName = "Li", EnrollmentDate = DateTime.Parse("2002-09-01") },  
 new Student { FirstMidName = "Peggy", LastName = "Justice", EnrollmentDate = DateTime.Parse("2001-09-01") },  
 new Student { FirstMidName = "Laura", LastName = "Norman", EnrollmentDate = DateTime.Parse("2003-09-01") },  
 new Student { FirstMidName = "Nino", LastName = "Olivetto", EnrollmentDate = DateTime.Parse("2005-09-01") }  
 };  
 students.ForEach(s => context.Students.Add(s));  
 context.SaveChanges();  
  
 var courses = new List<Course>  
 {  
 new Course { Title = "Chemistry", Credits = 3, },  
 new Course { Title = "Microeconomics", Credits = 3, },  
 new Course { Title = "Macroeconomics", Credits = 3, },  
 new Course { Title = "Calculus", Credits = 4, },  
 new Course { Title = "Trigonometry", Credits = 4, },  
 new Course { Title = "Composition", Credits = 3, },  
 new Course { Title = "Literature", Credits = 4, }  
 };  
 courses.ForEach(s => context.Courses.Add(s));  
 context.SaveChanges();  
  
 var enrollments = new List<Enrollment>  
 {  
 new Enrollment { StudentID = 1, CourseID = 1, Grade = 1 },  
 new Enrollment { StudentID = 1, CourseID = 2, Grade = 3 },  
 new Enrollment { StudentID = 1, CourseID = 3, Grade = 1 },  
 new Enrollment { StudentID = 2, CourseID = 4, Grade = 2 },  
 new Enrollment { StudentID = 2, CourseID = 5, Grade = 4 },  
 new Enrollment { StudentID = 2, CourseID = 6, Grade = 4 },  
 new Enrollment { StudentID = 3, CourseID = 1 },  
 new Enrollment { StudentID = 4, CourseID = 1, },  
 new Enrollment { StudentID = 4, CourseID = 2, Grade = 4 },  
 new Enrollment { StudentID = 5, CourseID = 3, Grade = 3 },  
 new Enrollment { StudentID = 6, CourseID = 4 },  
 new Enrollment { StudentID = 7, CourseID = 5, Grade = 2 },  
 };  
 enrollments.ForEach(s => context.Enrollments.Add(s));  
 context.SaveChanges();  
  
 }  
 }  
}

这样还没有完 还要再 Global.asax 下的Application\_Start()加上如下代码

Database.SetInitializer<SchoolContext>(new SchoolInitializer());

# 2页面helper的用法示例

@helper Topercent(int input)

{

@\* int SummeryCount = 1;

if (ViewBag.PaperCount == null || ViewBag.PaperCount== 0)

{}

else { SummeryCount = ViewBag.PaperCount; }

@(input\*100/SummeryCount); \*@

@input;

}

3：使用异步Controller

public class AccountingController:AsyncController

{

public void ExportAsync()

{ }

public void ExportCompleted()

{ }

}

# 3：mvc使用分页功能

# 4：mvc上传文件

控制器：

[HttpPost]

public ActionResult Upload\_file()

{

HttpPostedFileBase file = Request.Files["file"];

if (file != null)

{

string filePath = Path.Combine(Server.MapPath("/Content/Upload"), Path.GetFileName(file.FileName));

file.SaveAs(filePath);

}

return View("Index");

}

视图：

@using (Html.BeginForm("Upload\_file", "Upload",IsPost))

{

<input type="file" name="file"/>

<input type="submit" value="Upload\_file" />

}

@\*<form action="/Test/Upload/Upload\_file" method="post" enctype="multipart/form-data">

<input type="file" name="file" /><br />

<input type="submit" name="Submit" id="Submit" value="Upload\_file" />

</form>\*@

# 5:个版本部署ASP.NET MVC 3

IIS6.0

1. 安装Microsoft .net FrameWork 4.0安装包;

2. 安装ASP.NET MVC 3;

3. 设置“Web扩展服务”中的“ASP.NET v4.0.0.30319”为允许，如下图所示：

4. 在IIS中发布网站，创建虚拟目录，ASP.NET版本选择4.0.30196；

5. 添加MVC的解析：

右击IIS中的网站选择“属性”-“主目录”-“配置”-“映射”-“插入”，配置如下：

可执行文件：C:\WINDOWS\Microsoft.NET\Framework\v4.0.30196\aspnet\_isapi.dll，

扩展名：.\*

确认文件是否存在：不选中

点击确定，最后如下图所示

注意事项：

1. NET Framework4下载地址：

http://www.microsoft.com/downloads/zh-cn/details.aspx?FamilyID=0A391ABD-25C1-4FC0-919F-B21F31AB88B7

2. MVC 3 下载地址：

http://www.microsoft.com/downloads/en/details.aspx?FamilyID=d2928bc1-f48c-4e95-a064-2a455a22c8f6

3. 若在安装完毕NET Framework后才安装的IIS，请重新在IIS中注册NET Framework，在运行中输入：

C:\Windows\Microsoft.NET\Framework\v4.0.30319\aspnet\_regiis.exe -i

4. 若出现需要修改"CompilerVersion”错误，请参照下面代码修改或添加网站的Web.config中configuration——system.codedom配置节：

<system.codedom>   
<compilers>   
<compiler language="c#;cs;csharp" extension=".cs" type="Microsoft.CSharp.CSharpCodeProvider,System, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" warningLevel="4">   
<providerOption name="CompilerVersion" value="v4.0"/>   
<providerOption name="WarnAsError" value="false"/>   
</compiler>   
<compiler language="vb;vbs;visualbasic;vbscript" extension=".vb" type="Microsoft.VisualBasic.VBCodeProvider, System, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" warningLevel="4">   
<providerOption name="CompilerVersion" value="v4.0"/>   
<providerOption name="OptionInfer" value="true"/>   
<providerOption name="WarnAsError" value="false"/>   
</compiler>   
</compilers>   
</system.codedom>

# 6:MVC布局页的使用

1视图页面指定布局页面时注意路径

@{

ViewBag.Title = "Index";

Layout = "../Shared/TestViewPage.cshtml";

}

2布局页面指定视图页面的版块

@RenderSection("Scripts",required:true)

<footer>

@RenderSection("Footer")

</footer>

3内容页的使用

@section Scripts{

<script src="../../../../Scripts/jquery-1.5.1-vsdoc.js" type="text/javascript">

</script>

}

4布局页使用默认段落

@if (IsSectionDefined("foot"))

{

@RenderSection("Footer")

}

else { <span> 。。。。。。¦</span>}

@RenderSection("foot ",false)

5视图页中使用

@section Milestone{}

# 7:AJAX ActionLink 的使用

<fieldset>

<legend>AJAX</legend>

<div>

@Ajax.ActionLink("Get News", "GetNews",

new AjaxOptions

{

UpdateTargetId="result",

InsertionMode=InsertionMode.Replace,

HttpMethod="GET",

}

)

</div>

<div id="result"></div>

</fieldset>

控制器：

public ActionResult GetNews()

{

var news = db.News.Include("Category");

return PartialView("\_GetNews", news.ToList());

}

# 8：jquery 本色的ajax使用

$(function () {

$("#FromUpload").submit(function (event) {

event.preventDefault();

alert("submit");

var from = $(this);

$.ajax({

type: "post",

url: from.attr("action"),

data: from.serialize(),

beforeSend: function () {

alert("beforeSend");

},

error: function () {

alert("error");

},

success: function (data) {

alert("success");

var jsonResult = data;

alert(jsonResult.result);

alert(jsonResult.date);

},

complete: function () {

alert("complete");

}

})

});

});