新建一个ContosoUniversity控制台程序

## 1.**设置网站样式**

打开views/shared\\_Layout.cshtml，

1. HTM辅助器

Html.ActionLink（超链接文本，行为，控制器，路由值，HTML属性值）

|  |
| --- |
| @Html.ActionLink("柳州职业技术学院", "Index", "Home", new { area = "" }, new { @class = "navbar-brand" }) |

|  |
| --- |
| <a class="navbar-brand" href="/">应用程序名称</a> |

.HTM辅助器Html.ActionLink（超链接文本，行为，控制器）

|  |
| --- |
| @Html.ActionLink("学生", "Index", "Student") |

显示实体属性名

@Html.DisplayNameFor(model => model.Name)

显示实体值

@Html.DisplayFor(modelItem => item.Name)

等价于

@item.Name

Html.BeginForm()

<form action="当前页面地址" method="post">

</form>

校验

@Html.ValidationSummary()

@Html.ValidationMessageFor()

@Html.EditorFor()

## 2.**安装 Entity Framework 6**

## **3.创建数据模型**

1. 接下来你将创建 Contoso 大学应用程序的实体类。 首先将以下三个实体：Student.cs、 Enrollment.cs 、Course.cs，学生类、注册类、课程类。
2. 2.模型N-M关系
3. 转换为两个1-N关系
4. 1-N关系 学生和注册记录之间是1：N关系。1个学生可以有多个注册记录
5. 1-N关系 课程和注册记录之间是1：N关系。1门课程可以有多个注册记录
6. Model中使用导航属性来表现1-N关系

|  |
| --- |
| 1. public class Student 2. { 3. public int ID { get; set; } 4. public string Name { get; set; } 5. public DateTime EnrollmentDate { get; set; } 6. public virtual ICollection<Enrollment> Enrollments { get; set; } 7. } |

|  |
| --- |
| 1. public class Enrollment 2. { 3. public int EnrollmentID { get; set; } 4. public int CourseID { get; set; } 5. public int StudentID { get; set; } 6. public Grade? Grade { get; set; } 7. public virtual Course Course { get; set; } 8. public virtual Student Student { get; set; } 9. } |
| public class Course   1. { 2. [DatabaseGenerated(DatabaseGeneratedOption.None)] 3. public int CourseID { get; set; } 4. public string Title { get; set; } 5. public int Credits { get; set; } 6. public virtual ICollection<Enrollment> Enrollments { get; set; } 7. } |
|  |

Enrollments 属性是导航属性。 一个 Course 体可以与任意数量的 Enrollment 实体相

## **4.创建数据库上下文**

## SchoolContext

### **指定实体集**

此代码将创建[DbSet](https://msdn.microsoft.com/library/system.data.entity.dbset(v=vs.113).aspx)每个实体集的属性。 在实体框架术语中，实体集通常对应于数据库表和一个实体对应于表中的行。

4.1建一个种子类

SchoolInitializer

4.2seed数据库初始化

（1）重写seed方法

|  |
| --- |
| public class SchoolInitializer : System.Data.Entity.DropCreateDatabaseIfModelChanges<SchoolContext>  {  protected override void Seed(SchoolContext context)  {  //构建学生数据  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();  //将学生数据加入实体集，保存实体集状态    //构建课程数据  //将课程数据加入实体集，保存实体集状态  //构建注册数据  //将注册数据加入实体集，保存实体集状态  }  ｝ |

（2）设置WEB.CONFIG

|  |
| --- |
| <entityFramework>  <contexts>  <context type="ContosoUniversity.DAL.SchoolContext, ContosoUniversity">  <databaseInitializer type="ContosoUniversity.DAL.SchoolInitializer, ContosoUniversity" />  </context>  </contexts>  </entityFramework> |

4.3.对象初始化

|  |
| --- |
| new 类型名｛属性1=值1，属性2=值2…｝ |

4.4集合初始化

|  |
| --- |
| new List<类型>{  对象1，  对象2，  …  } |

初始化在Web.config

<contexts> <context type="ContosoUniversity.DAL.SchoolContext, ContosoUniversity"><databaseInitializertype="ContosoUniversity.DAL.SchoolInitializer, ContosoUniversity" /> </context>

</contexts>

## **5.设置为使用 SQL Server Express LocalDB 数据库**

请确保更新Web.config根项目文件夹中的文件。 此外，还有Web.config中的文件视图无需更新的子文件夹。

<connectionStrings>

<add name="SchoolContext" connectionString="Data Source=(LocalDb)\MSSQLLocalDB;InitialCatalog=ContosoUniversity1;IntegratedSecurity=SSPI;" providerName="System.Data.SqlClient"/>

</connectionStrings>

## **6.创建一个学生控制器和视图**

 StudentController

## **6.1创建详细信息页**

密钥的值传递给方法作为id参数和来自将数据路由中详细信息索引页上的超链接。

路由数据是在路由表中指定的 URL 段中找到的模型联编程序的数据

url: "{controller}/{action}/{id}",

添加突出显示的代码以显示注册列表

此代码循环通过 Enrollments 导航属性中的实体。 每个Enrollment实体在属性中，它将显示课程标题和成绩。

## **6.2更新“创建”页**

安全警告-ValidateAntiForgeryToken特性帮助抵御[跨站点请求伪造](https://docs.microsoft.com/zh-cn/aspnet/mvc/overview/security/xsrfcsrf-prevention-in-aspnet-mvc-and-web-pages)攻击。 它需要相应Html.AntiForgeryToken()语句在视图中

Bind属性是一种方法来防止过度发布创建方案中

Create.chstml还包括@Html.AntiForgeryToken()，适用于ValidateAntiForgeryToken属性中的控制器，以帮助防止[跨站点请求伪造](https://docs.microsoft.com/zh-cn/aspnet/mvc/overview/security/xsrfcsrf-prevention-in-aspnet-mvc-and-web-pages)攻击。

## **6.3更新编辑 HttpPost 方法**

## **6.4更新“删除”页**

6.5向学生索引页添加列排序链接

要添加到学生索引页进行排序，将更改Index方法Student控制器并将代码添加到Student索引视图

### **6.5.1向 Index 方法添加排序功能**

### **6.6.2将列标题超链接添加到学生索引视图**

ViewBag属性具有相应查询的超链接设置字符串值。

## **6.6.3向学生索引页添加搜索框**

## **6.6.4向学生索引页添加分页**

加分页需要安装 PagedList.MVC NuGet 包

NuGet PagedList.Mvc会自动安装包PagedList包作为依赖项

命令是：Install-Package PagedList.Mvc

### 6.6.5再向**向 Index 方法添加分页功能**

1. 此将添加page参数、 当前的排序顺序参数和当前的筛选器参数向方法签名

### 6.6.6再**向学生索引视图添加分页链接**

1. 页面顶部的 @model 语句指定视图现在获取的是 PagedList 对象，而不是 List 对象。

## **7.创建显示学生统计信息的“关于”页**

### **7.1创建视图模型**

EnrollmentDateGroup.cs

### **7.2修改主控制器**

1. HomeController.cs，

### 7.3再**修改“关于”视图**

# 8.迁移、种子数据与模型改变

## 8.1迁移

在设计迁移之前，若数据库已经存在，可在连接字符串中变更一次数据库名称

## 8.2启动迁移

|  |
| --- |
| enable-migrations |
| enable-migrations命令在项目文件夹中创建迁移文件夹，在该文件夹中放入Configuration.cs可编辑以配置 Migrations 的文件。 |

## 8.3新增变迁

Add-Migration 将基于自上次迁移创建以来对模型所做的更改来构建下一次迁移

如果启动迁移前数据库已经存在，启动迁移后系统将自动查找现有数据库并自动自行一次Add-Migration命令。

|  |
| --- |
| add-migration InitialCreate |
| add-migration 迁移名称（概述迁移原由等） |

## 8.4更新数据库

Update-Database 将对数据库应用任意挂起的迁移

## 8.5种子文件

可以将种子函数的内容移动到迁移配置类文件Configuration.cs中

多方实体实行新增获更新

联系实体依靠查询两个多方获得ID，使用更新

|  |
| --- |
|  |

### 8.6设置完**执行第一次迁移**

add-migration

再更新数据库

update-database命令将运行Up方法来创建数据库

## **8.7使用特性自定义数据模型**

### **DataType 特性**

## [DataType 枚举](https://msdn.microsoft.com/library/system.componentmodel.dataannotations.datatype.aspx)提供了多种数据类型，如日期、 时间、 电话号码、 货币、 电子邮件地址和的详细信息。

## Display 特性指定文本框的标题应是“名”、“姓”、“全名”和“注册日期”，而不是每个实例中的属性名称（其中没有分隔单词的空格）。

### **FullName 计算属性**

FullName 是计算属性，可返回通过串联两个其他属性创建的值。 因此只有get访问器，但没有FullName将在数据库中生成列。

再进行数据迁移

add-migration MaxLengthOnNames

和更新数据库

update-database

### 加**列属性**

添加了[列属性](https://msdn.microsoft.com/library/system.componentmodel.dataannotations.schema.columnattribute.aspx)更改备份 SchoolContext，因此它不会与数据库匹配的模型

因此又做一次迁移

add-migration ColumnFirstName

再更新数据库

update-database

## **9.创建讲师实体Instructor**

### **课程和 OfficeAssignment 导航属性**

## Courses 和 OfficeAssignment 是导航属性

## 表示一名讲师可以教授任意数量的课程

## **10.创建课程实体OfficeAssignment**

### **增加键属性**

## Key属性如果实体具有其自己的主键

### 增加**ForeignKey 属性**

## 当两个实体之间的一对一关系或一对零或一一种关系 (如OfficeAssignment和Instructor)，EF 不能计算出的关系的哪一端是主体，依赖哪一端。

### **Instructor 导航属性**

### Instructor实体具有一个可以为 nullOfficeAssignment导航属性 （因为一名讲师可能没有办公室分配），并OfficeAssignment实体具有不可为 nullInstructor导航属性 （因为办公室分配不能没有讲师-InstructorID不可为 null)。

## **11.修改 Course 实体**

Course 实体具有外键属性DepartmentID它指向相关Department实体，同时它具有Department导航属性。

### **DatabaseGenerated 特性**

[DatabaseGenerated 特性](https://msdn.microsoft.com/library/system.componentmodel.dataannotations.schema.databasegeneratedattribute.aspx)与[None](https://msdn.microsoft.com/library/system.componentmodel.dataannotations.schema.databasegeneratedoption(v=vs.110).aspx)参数CourseID属性指定主键值是由用户提供而不是由数据库生成。

### **11.1外键和导航属性**

外键属性和导航属性中的Course实体可反映以下关系：向一个系分配课程后，出于上述原因，会出现 DepartmentID 外键和 Department 导航属性。

参与一门课程的学生数量不定，因此 Enrollments 导航属性是一个集合：

public virtual ICollection<Enrollment> Enrollments { get; set; }

一门课程可能由多位讲师讲授，因此 Instructors 导航属性是一个集合：

public virtual ICollection<Instructor> Instructors { get; set; }

## **12.创建部门实体Department**

### **12.1增加列属性**

[Column(TypeName="money")]

public decimal Budget { get; set; }

### **外键和导航属性**

外键和导航属性可反映以下关系：一个系可能有也可能没有管理员，而管理员始终是讲师。

一个系可以有多门课程，因此没有Courses导航属性：

public virtual ICollection<Course> Courses { get; set; }

## **13.修改注册实体Enrollment**

### **外键和导航属性**

外键属性和导航属性可反映以下关系：注册记录面向一门课程，因此存在 CourseID 外键属性和 Course 导航属性：public int CourseID { get; set; } public virtual Course Course { get; set; }

注册记录面向一名学生，因此存在 StudentID 外键属性和 Student 导航属性：

public int StudentID { get; set; } public virtual Student Student { get; set; }

### **多对多关系**

## **14.修改测试数据设定数据库种子**

## **15.添加迁移并更新数据库**

## **16.创建该显示院系名称的课程页**

1. 更改将标题从****索引****到****课程****。

## **17.创建显示课程和注册的讲师页**

### **17.1修改讲师索引视图**

## **18.自定义课程的创建和编辑页面**

# 在中CourseController.cs，删除四个Create和Edit方法

## **19.添加讲师的编辑页面**

## **20.添加课程分配给讲师编辑页**

## **更新 DeleteConfirmed 方法**

## 在中InstructorController.cs，删除DeleteConfirmed方法并插入以下代码在其原位置。

## **21.向创建页添加办公室位置和课程**

## **22.创建部门控制器**

## **22.1将存储的过程用于插入、 更新和删除**