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()

具体参考—MVC高级编程第4章HTML辅助器

2.模型N-M关系

转换为两个1-N关系

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

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

Model中使用导航属性来表现1-N关系

|  |
| --- |
| public class Student  {  public int ID { get; set; }  public string Name { get; set; }  public DateTime EnrollmentDate { get; set; }  public virtual ICollection<Enrollment> Enrollments { get; set; }  } |

|  |
| --- |
| public class Enrollment  {  public int EnrollmentID { get; set; }  public int CourseID { get; set; }  public int StudentID { get; set; }  public Grade? Grade { get; set; }  public virtual Course Course { get; set; }  public virtual Student Student { get; set; }  } |

|  |
| --- |
| public class Course  {  [DatabaseGenerated(DatabaseGeneratedOption.None)]  public int CourseID { get; set; }  public string Title { get; set; }  public int Credits { get; set; }  public virtual ICollection<Enrollment> Enrollments { get; set; }  } |

3.数据特性—取消数据库主键自增长特性，改为用户自己为记录赋予ID

|  |
| --- |
| using System.Collections.Generic;  using System.ComponentModel.DataAnnotations.Schema;  namespace ContosoUniversity.Models  {  public class Course  {  [DatabaseGenerated(DatabaseGeneratedOption.None)]  public int CourseID { get; set; }  public string Title { get; set; }  public int Credits { get; set; }  public virtual ICollection<Enrollment> Enrollments { get; set; }  }  } |

数据特性具体参考—MVC高级编程第5章数据注解和验证

4.seed数据库初始化

（1）重写seed方法

|  |
| --- |
| publicclass SchoolInitializer : System.Data.Entity.DropCreateDatabaseIfModelChanges<SchoolContext>  {  protected override void Seed(SchoolContext context)  {  //构建学生数据  //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> |

5.对象初始化(设置以使用测试数据初始化数据库=>DAL文件夹)

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

6.集合初始化

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

7. 创建一个学生控制器和视图

|  |
| --- |
| (1) 创建**控制器:**  **1》使用MVC5的带有视图用Entity Framework然后选择添加:**  **模型类：学生 (Student)。 （如果看不到下拉列表中的此选项，生成项目，然后重试。）**  **数据上下文类： SchoolContext (ContosoUniversity.DAL)。**  **生成视图，使用同名的视图。**  **控制器名称： StudentController (不是 StudentsController)。** |

**(2)** 控制器的行为了解（行为==C#的方法）

|  |  |
| --- | --- |
| **1》**您看到类变量已创建了实例化数据库上下文对象：  **private SchoolContext db = new SchoolContext();**  2》//安全策略A ：[ValidateAntiForgeryToken]防止你输入的账号和密码不经过加密后，通过get请求，暴露在前台。  安全警告-ValidateAntiForgeryToken特性帮助抵御跨站点请求伪造攻击。 它需要相应Html.AntiForgeryToken()语句在视图中，您将会看到更高版本。  //安全策略B：   |  | | --- | | C:\Users\钟强\AppData\Roaming\Tencent\Users\1196341787\QQ\WinTemp\RichOle\(IW(3W`I_B2S1_[2K_]]G09.png |   3》[HttpPost] 这是用来定义在提交或确定请求HTTP下的方式为Post  4》变量后加？号 表示为空  C:\Users\钟强\AppData\Roaming\Tencent\Users\1196341787\QQ\WinTemp\RichOle\_4N6POHB32EK]E$$4_MC~J9.png  5》关闭数据库上下文  protected override void Dispose(bool disposing)  {  if (disposing)  {  db.Dispose();  }  base.Dispose(disposing);  } |

|  |
| --- |
| **8. 创建详细信息页**  1》调用控制器的Details行为此行为传一个id? 可以为空  方法分析：  public ActionResult Details(int? id)  {  if (id == null) //如果id为空  { //返回 HTTP提示服务器未能识别请求  链接：https://www.imooc.com/article/28813  return new HttpStatusCodeResult(HttpStatusCode.BadRequest);  }  Student student = db.Students.Find(id);//如果有通过数据集用id找一列存到  if (student == null)//判断数据是否为空  {  return HttpNotFound();返回一空404  }  return View(student); 传数据给视图并返回视图  }  2》查看路由数据URL是否正确 控制器=>行为=>id  3》视图的修改：  1.通过创建 Url ActionLink Razor 视图中的语句。 在下面的代码中，id参数匹配的默认路由，因此id添加到路由数据。  @Html.ActionLink("Select", "Index", new { id = item.PersonID })  2.在下面的代码，courseID不匹配默认路由中的参数，因此，它将添加为查询字符串。  @Html.ActionLink("Select", "Index", new { courseID = item.CourseID })   1. 创建的详细信息页 2. 添加HTML和C#代码到视图=>将看到所选学生的课程和年级列表： |

**9. 向学生索引页添加列排序链接**

### 10. 向学生索引视图添加搜索框

|  |
| --- |
| 1.在控制器Index行为上加一个searchString 参数-----用来从视图的文本框中接收要添加到索引的搜索字符串值。  2.使用if语句：  if (!String.IsNullOrEmpty(searchString))//判断传过来的searchString是非空还是非null  {  //把值用where 表达式找到是否包含此值  students = students.Where(s => s.LastName.Contains(searchString)  }   1. 视图显示：   向学生索引视图添加搜索框----创建标题栏、 文本框中，标记和一个搜索按钮。打开之前立即突出显示的代码到table  @using (Html.BeginForm())  {  <p>  Find by name: @Html.TextBox("SearchString")  <input type="submit" value="Search" /></p>  } |