**[学习ASP.NET MVC(九)——“Code First Migrations ”工具使用示例](http://www.cnblogs.com/chillsrc/p/4131023.html)**

在上一篇文章中，我们学习了如何使用实体框架的“Code First Migrations ”工具，使用其中的“迁移”功能对模型类进行一些修改，同时同步更新对应数据库的表结构。

在本文章中，我们将使用“Code First Migrations ”工具。使用其中的“Send”更新方法，添加Rating字段到每个Book对象与数据库表中。

**一、在Book模型中添加“Rating”属性**

首先，在Visual Studio打开Models\ Book.cs文件，在Book类中添加一个新的Rating属性。代码如下：

public string Rating { get; set; }

完整的Book类。代码如下。

[复制代码](javascript:void(0);)

public class Book

{

public int BookID { get; set; }

public string Category { get; set; }

public string Name { get; set; }

public int Numberofcopies { get; set; }

public int AuthorID { get; set; }

public decimal Price { get; set; }

public DateTime PublishDate { get; set; }

public string Rating { get; set; }

}

[复制代码](javascript:void(0);)

第二、使用菜单“生成-->生成解决方案”来生成应用程序。或按下CTRL-SHIFT-B。   
第三、现在你已经更新了模型类，你还需要更新在\ Views\Book\ Index.cshtml和\Views\Book\ Create.cshtml视图模板，以便在浏览器视图中显示新的“Rating”属性。   
在解决方案管理器中打开，Views\Book\Index.cshtml文件，并在PublishDate列标题后面添加一个<TH>Rating</TH>列。然后在PublishDate列后面添加一个<TD>列，用来呈现Rating值。下面是更新后的Index.cshtml视图模板的代码：

[复制代码](javascript:void(0);)

@model IEnumerable<MvcApplication1.Models.Book>

@{

ViewBag.Title = "Index";

}

<h2>Index</h2>

<p>

@Html.ActionLink("Create New", "Create")

</p>

<table>

<tr>

<th>

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

</th>

<th>

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

</th>

<th>

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

</th>

<th>

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

</th>

<th>

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

</th>

<th>

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

</th>

<th>

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

</th>

<th></th>

</tr>

@foreach (var item in Model) {

<tr>

<td>

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

</td>

<td>

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

</td>

<td>

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

</td>

<td>

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

</td>

<td>

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

</td>

<td>

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

</td>

<td>

@Html.DisplayFor(model => item.Rating)

</td>

<td>

@Html.ActionLink("Edit", "Edit", new { id=item.BookID }) |

@Html.ActionLink("Details", "Details", new { id=item.BookID }) |

@Html.ActionLink("Delete", "Delete", new { id=item.BookID })

</td>

</tr>

}

</table>

[复制代码](javascript:void(0);)

第四、在“解决方案资源管理器”中打开“Views\Book\Create.cshtml文件，并在文件的末尾附近添加下面的标记。这里使用一个文本框，方便您可以创建一个新的书籍时指定一个“Rating”。代码如下。

[复制代码](javascript:void(0);)

<div class="editor-label">

@Html.LabelFor(model => model.Rating)

</div>

<div class="editor-field">

@Html.EditorFor(model => model.Rating)

@Html.ValidationMessageFor(model => model.Rating)

</div>

[复制代码](javascript:void(0);)

第五、现在已经更新了应用程序的代码，应用程序已经可以支持新的“Rateing”属性。   
按F5，运行应用程序，在浏览器中浏览/Book/Index网址。此时你会看到下面的这个错误。如下图1、图2。

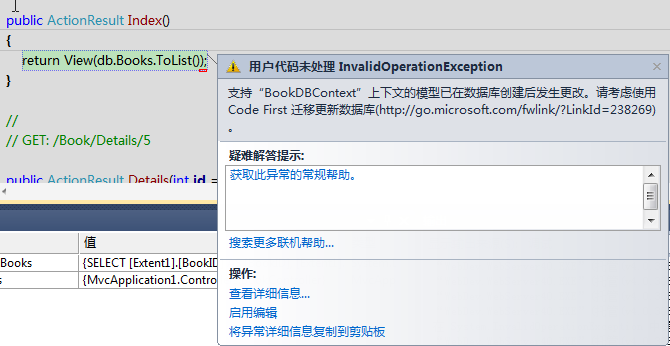


图1

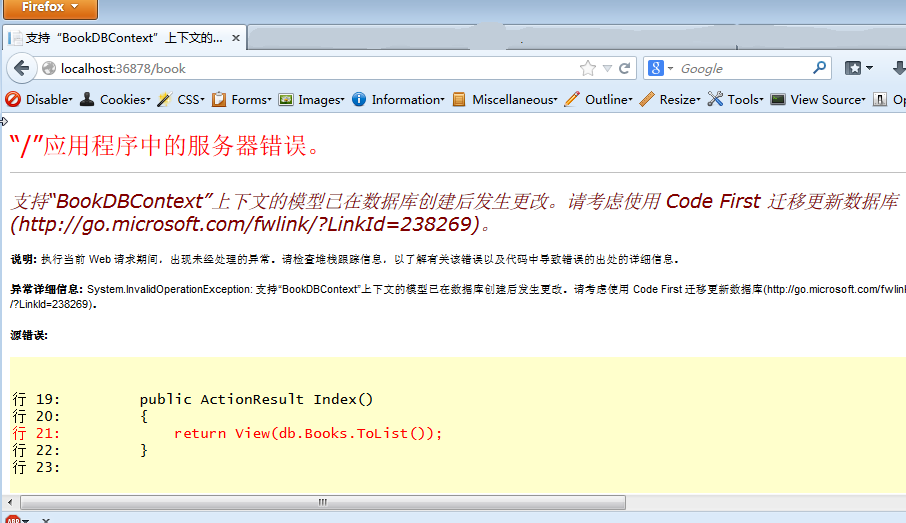


图2

之所以出现这个错误信息，是因为在应用程序更新的Book模型类与现有数据库的Book表的架构不同。 （即Book模型类中有一个新的属性Rating，而数据库中的Book表中却没有这个字段。 ）  
这个错误可以通过以下几个方法来解决这个错误：

1 ) 由实体构架使用新实体的新模型类架构自动删除并重新创建数据库。这种做法能非常方便的创建一个测试数据库；它可以让你快速进行开发，模型和数据库架构是在一起的。不过这种方法的缺点是，你将失去数据库中已经存在的数据 ——所以不建议在生产环境使用这种方法！使用这种方式自动生成一个数据库，并在数据库中插入初始化的测试数据是非常方便的方式，并且可以快速开发应用程序。

2 ) 手动修改现有数据库的架构，让它匹配新的模型类。这种方法的好处是，你可以保有你已经存在的测试数据。您可以手动或通过创建一个数据库更改脚本进行此更改。  
3，使用“Code First Migrations ”工具更新数据库架构。  
在本文章中，我们将使用“Code First Migrations ”工具。  
更新“Send”方法，以便它提供了新的列的值。打开Migrations\Configuration.cs文件，添加Rating字段到每个Book对象中。代码如下。

[复制代码](javascript:void(0);)

new Book

{

Name = "DB 2",

PublishDate = DateTime.Parse("1986-2-23"),

Category = "IBM",

AuthorID=1,

Numberofcopies=22,

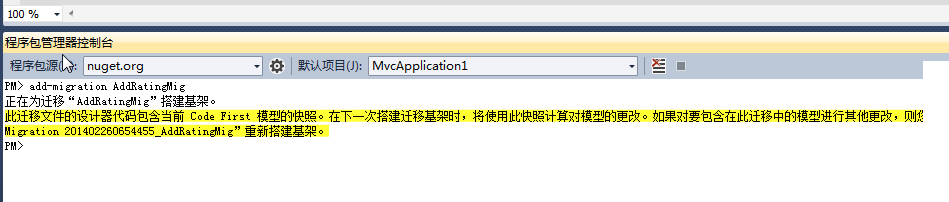
Price = 9.99M,

Rating = "H"

}

[复制代码](javascript:void(0);)

第六、执行菜单--》工具--》生成--》生成解决方案，然后在程序包管理器控制台窗口中输入“add-migration AddRatingMig”命令。 如下图。

  
下图显示了在程序包管理器控制台窗口中的输出（时间戳前面加上AddRatingMig会有所不同。）  
这个“add-migration ”命令告诉迁移框架，研究当前Book的模型与当前Book数据库模式，并建立必要的代码到数据库迁移到新的模型。该AddRatingMig是任意的，是用来命名迁移文件。它有助于使用一个有意义的名称迁移步骤。   
当这个命令执行完成后，Visual Studio将打开一个新定义的DbMIgration派生类的类文件，你可以参阅新创建的代码，代码如下。

[复制代码](javascript:void(0);)

public partial class AddRatingMig : DbMigration

{

public override void Up()

{

AddColumn("dbo.Books", "Rating", c => c.String());

}

public override void Down()

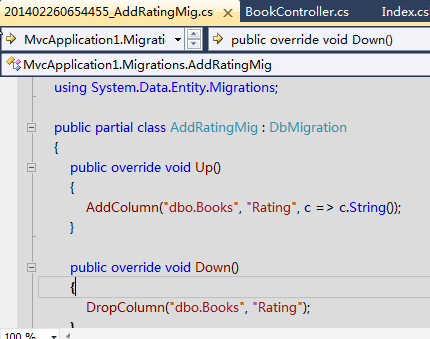
{

DropColumn("dbo.Books", "Rating");

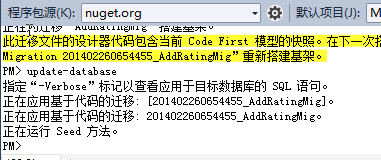
}

}

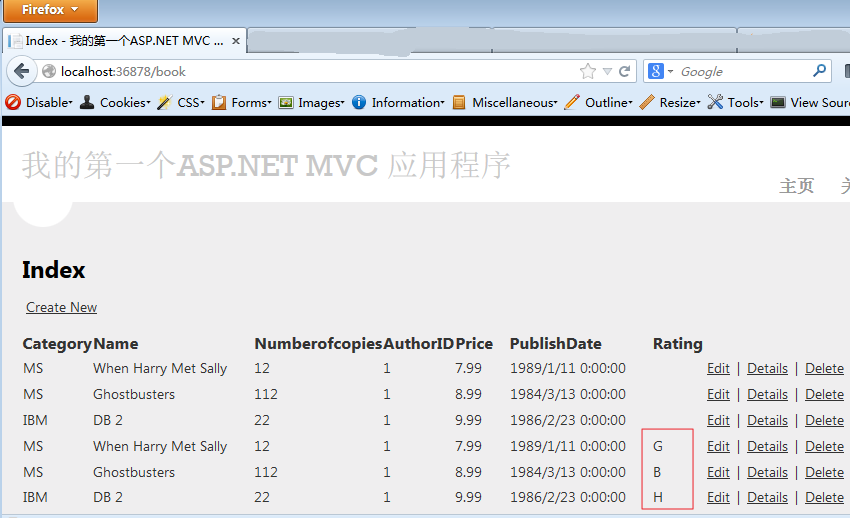
[复制代码](javascript:void(0);)



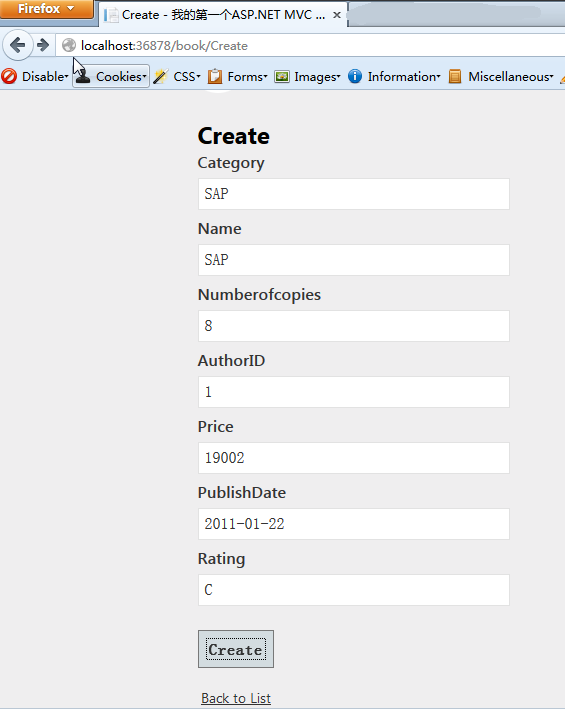
第七、执行菜单--》工具--》生成--》生成解决方案，然后在程序包管理器控制台窗口中输入“update-database”命令。 如下图。  
下图显示了在程序包管理器控制台窗口中的输出（时间戳前面加上AddRatingMig会有所不同。）



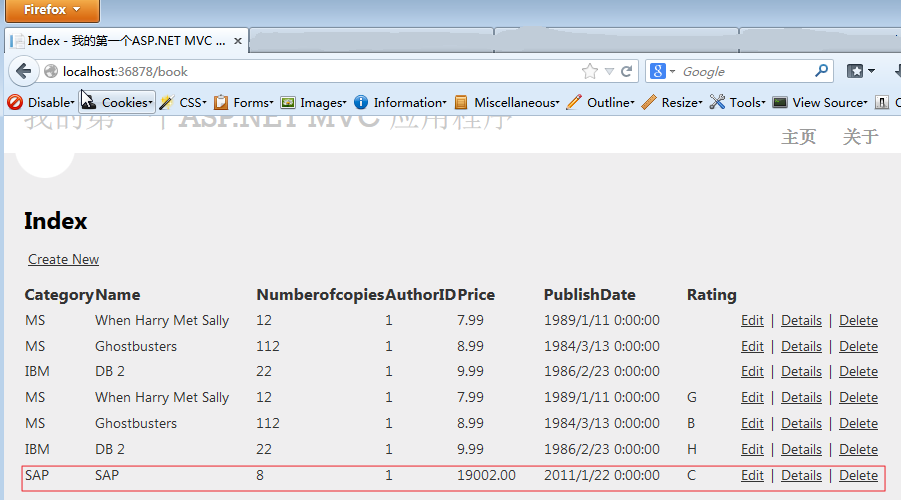
按F5，重新运行这个应用程序，使用浏览器浏览/Book/Index网址。你可以看到新的列表页面，会增加一个Rating字段进行显示。如下图。



第八、单击“Create New”链接，将导航到添加一个新的书籍信息的页面。请注意，在这个新的页面中，您可以添加一个等级字段信息。



单击“Create”按钮。一本新的书籍数据，包括评级字段，现在都显示在列表中了：如下图。



你也应该将Rating字段添加到Edit，Details及SearchIndex三个视图模板。   
你可以再次进入程序包管理器控制台窗口中执行“update-database”命令，窗口中输入内容并没有变化，因为架构模型相匹配。

在这第八、第九两篇文章中，您学会了如何修改模型对象；学会如何在修改模型对象之后更新数据库的结构；学会如何使用样本数据来填充新创建的数据库。