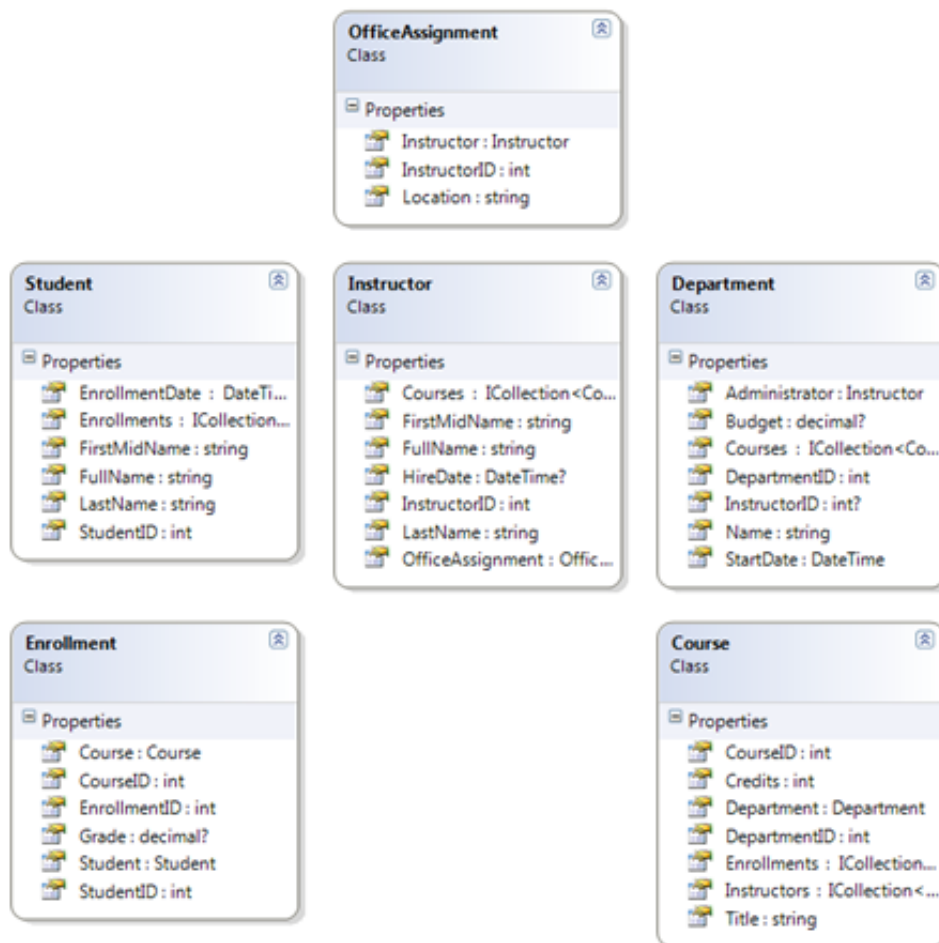


Создание сложной модели данных для приложения ASP.NET MVC

В предыдущих уроках вы научились работать с простой моделью данных, состоящей из трёх сущностей. В этом уроке вы добавите несколько сущностей и связей между ними и научитесь работать с аннотациями для управления классами моделей.

Результат будет выглядеть так:



Использование атрибутов для управления форматированием, валидацией и маппингом базы данных

В этом уроке вы увидите примеры атрибутов, которые вы можете добавить в классы моделей для управления форматированием, валидацией и маппингом базы данных. Затем вы создадите полноценную модель данных School путём добавления атрибутов в уже созданные классы и созданием новых классов для оставшихся в модели типов сущностей.

Атрибут DisplayFormat

Для дат записи студентов, все страницы сейчас отображают время вместе с датой, тогда как нам нужна только дата. С помощью аннотаций можно указать в одном месте настройку, которая будет распространяться на всё. Для этого добавим атрибут к свойству EnrollmentDate в классе

Student.

В *Models\Student.cs* добавьте `using` для `System.ComponentModel.DataAnnotations` и атрибут `DisplayFormat` для свойства `EnrollmentDate`:

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

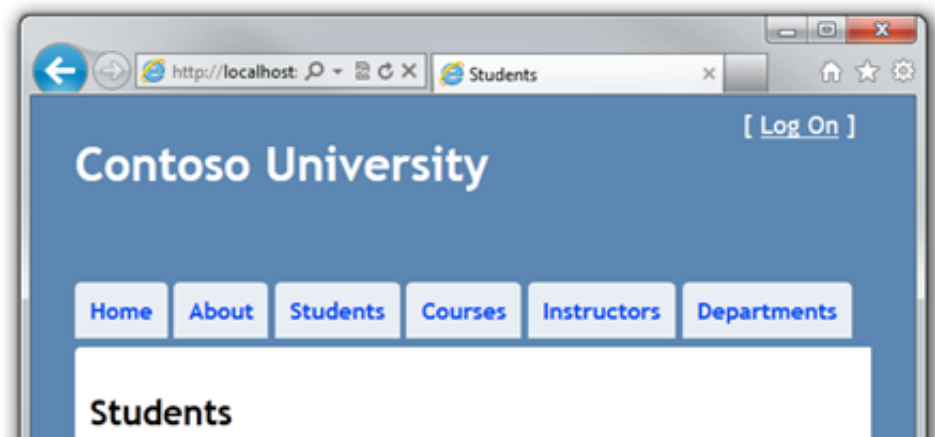
namespace ContosoUniversity.Models
{
    public class Student
    {
        public int StudentID { get; set; }
        public string LastName { get; set; }
        public string FirstMidName { get; set; }

        [DisplayFormat(DataFormatString = "{0:d}", ApplyFormatInEditMode = true)]
        public DateTime EnrollmentDate { get; set; }

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

Строка форматирования указывает, что необходимо отображать только короткую форму даты этого свойства. Настройка `ApplyFormatInEditMode` указывает, что данное форматирование должно применяться также для значений, отображающихся в текстовых строках, предназначенных для редактирования. (допустим, нам не нужно отображать в некоторых полях, например, в которых содержатся значения, связанные с валютой, знак валюты)

Убедитесь, что на страницах *About* и *Student* время в датах больше не показывается.



[Create New](#)

Find by name:

	Last Name	First Name	Enrollment Date
Edit Details Delete	Alexander	Carson	9/1/2011
Edit Details Delete	Alonso	Meredith	9/1/2002
Edit Details Delete	Anand	Arturo	9/1/2003

Page 1 of 3 << < Prev [Next](#) > >>

Атрибут MaxLength

Атрибутами можно определить правила валидации, допустим, в том случае, если необходимо ограничить введённое пользователем значение 50-ю символами. Чтобы воспользоваться этой возможностью, добавьте атрибут Range в свойствах LastName и FirstMidName:

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace ContosoUniversity.Models
{
    public class Student
    {
        public int StudentID { get; set; }

        [MaxLength(50)]
        public string LastName { get; set; }

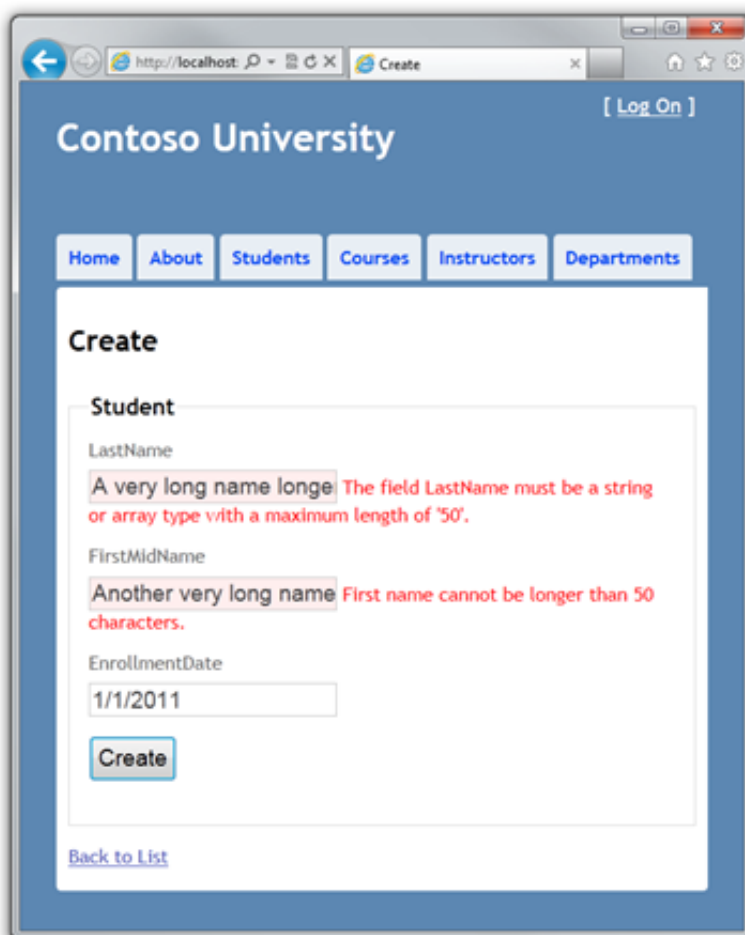
        [MaxLength(50, ErrorMessage = "First name cannot be longer than 50 characters.")]
        public string FirstMidName { get; set; }

        [DisplayFormat(DataFormatString = "{0:d}", ApplyFormatInEditMode = true)]
        public DateTime EnrollmentDate { get; set; }

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

При попытке ввести фамилию из более чем 50-ти символов будет выведено стандартное сообщение об ошибке, в подобном случае же для имени будет выведено наше собственное сообщение об ошибке.

Откройте страницу Create и попробуйте ввести два имени из более чем 50-ти символов, затем нажмите Create чтобы убедиться что валидация работает.



The screenshot shows a web browser window with the URL `http://localhost:...` and the page title "Create". The page is for "Contoso University" and has a navigation bar with links: Home, About, Students, Courses, Instructors, and Departments. The main content area is titled "Create" and contains a form for creating a new student. The form has three fields: "LastName", "FirstMidName", and "EnrollmentDate". The "LastName" field contains the text "A very long name longer" and has a red error message: "The field LastName must be a string or array type with a maximum length of '50'". The "FirstMidName" field contains the text "Another very long name" and has a red error message: "First name cannot be longer than 50 characters". The "EnrollmentDate" field is a date picker set to "1/1/2011". There is a "Create" button and a "Back to List" link at the bottom.

Указывать максимальное значение для вводимых элементов – одна из хороших практик. В ином случае, если используется подход Code First, при создании базы данных имена соответствующих столбцов будут иметь максимально допустимую длину.

Атрибут Column

Атрибутами можно контролировать маппинг классов на базу данных. Допустим, вы использовали имя `FirstMidName` для поля «Имя» потому, что это поле могло также содержать отчество. Но вы хотите, чтобы в базе данных соответствующий столбец назывался `FirstName`, потому что пользователи пользуются в своих запросах именно этим наименованием. Для этого можно воспользоваться атрибутом `Column`.

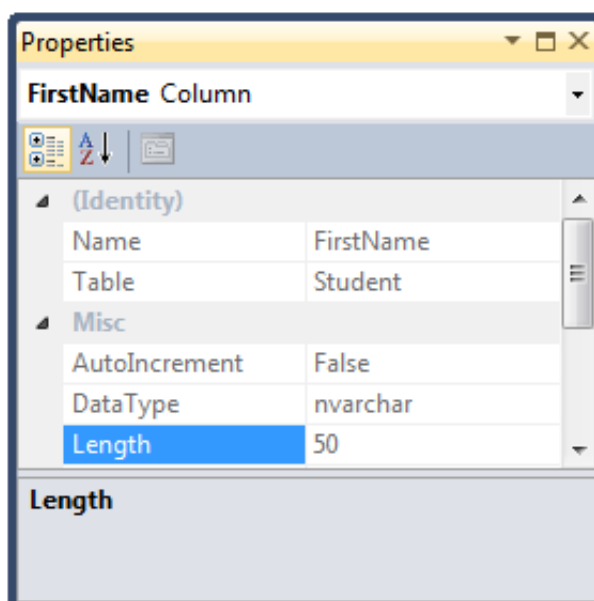
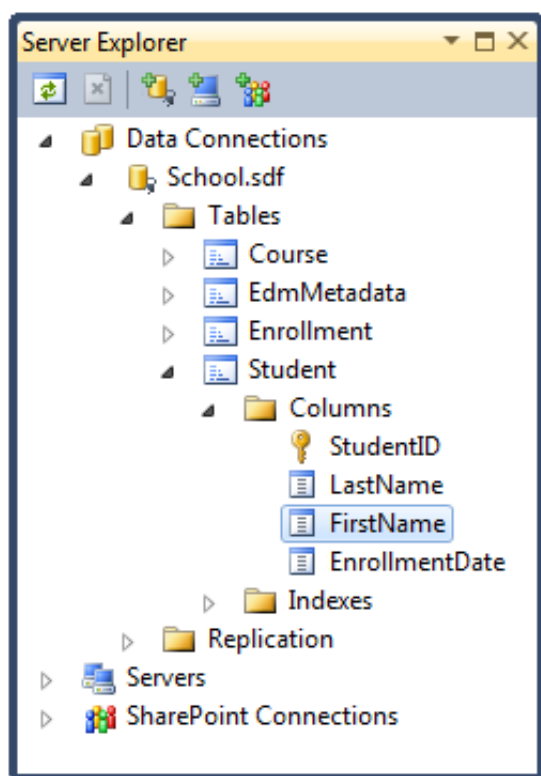
Атрибут `Column` указывает, что при создании базы данных, столбец таблицы `Student`, который маппится на свойство `FirstMidName`, будет называться `FirstName`.

Добавьте этот атрибут к свойству FirstMidName:

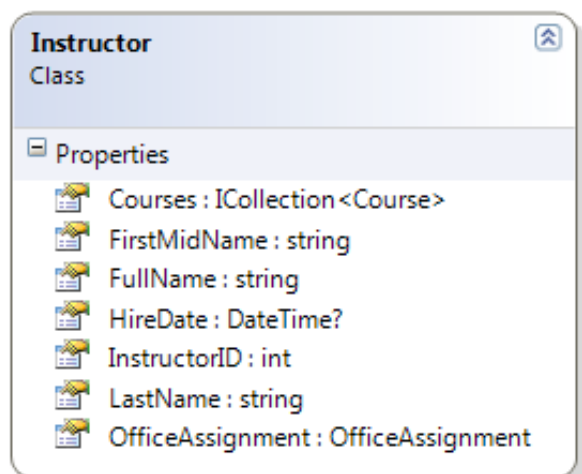
```
[Column("FirstName")]  
public string FirstMidName { get; set; }
```

Откройте страницу Student Index, на которой ничего не изменилось (необходимо не просто запустить сайт и посмотреть на домашнюю страницу, необходимо перейти на страницу Student Index чтобы инициировать обращение к базе данных, что иницирует также ее удаление и пересоздание). Однако, если вы откроете базу данных **Server Explorer** вы увидите, что в таблице Student столбец называется FirstName.

В окне **Properties** можно заметить, что длина ограничена 50-тью символами благодаря использованию атрибута MaxLength.



Создание сущности Instructor



Создайте *Models\Instructor.cs* со следующим кодом:

```
using System;  
using System.Collections.Generic;  
using System.ComponentModel.DataAnnotations;  
  
namespace ContosoUniversity.Models  
{  
    public class Instructor  
    {
```

```

        public Int32 InstructorID {
get; set; }

        [Required(ErrorMessage = "Last name is required.")]
        [Display(Name="Last Name")]
        [MaxLength(50)]
        public string LastName { get; set; }

        [Required(ErrorMessage = "First name is required.")]
        [Column("FirstName")]
        [Display(Name = "First Name")]
        [MaxLength(50)]
        public string FirstMidName { get; set; }

        [DisplayFormat(DataFormatString = "{0:d}", ApplyFormatInEditMode = true)]
        [Required(ErrorMessage = "Hire date is required.")]
        [Display(Name = "Hire Date")]
        public DateTime? HireDate { get; set; }

        public string FullName
        {
            get
            {
                return LastName + ", " + FirstMidName;
            }
        }

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

```

Обратите внимание, что некоторые из свойств аналогичны свойствам сущности Student. В **Implementing Inheritance** мы используем наследование для решения проблемы избыточности.

Атрибуты Required и Display

Атрибуты свойства LastName указывают на то, что это необходимое для заполнения значение, заголовок для текстового поля должен быть равен “Last Name” (вместо названия свойства “LastName”) и длина введённого значения должна быть не более 50-ти символов..

```
[Required(ErrorMessage = "Last name is required.")]
[Display(Name="Last Name")]
[MaxLength(50)]
public string LastName { get; set; }
```

Вычисляемое свойство FullName

FullName является вычисляемым свойством, которое содержит значение, вычисляемое путем соединения двух других свойств. Поэтому оно имеет только аксессор, в базе данных же не создаётся соответствующего столбца.

```
public string FullName
{
    get
    {
        return LastName + ", " + FirstMidName;
    }
}
```

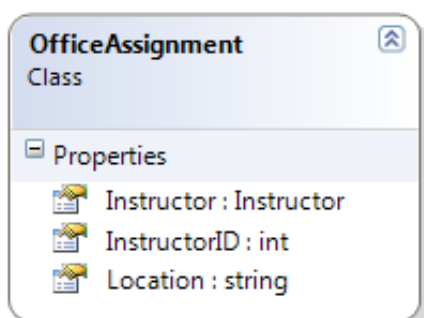
Navigation Properties Courses и OfficeAssignment

Свойства Courses и OfficeAssignment - navigation properties. Как было упомянуто ранее, они обычно помечаются модификатором virtual чтобы использовать lazy loading. Кроме этого, если данное свойство может содержать несколько сущностей, его тип должен быть ICollection.

Поскольку преподаватель может вести несколько курсов, переменная Courses определена как коллекция сущностей Course. С другой стороны, преподаватель может иметь только один офис, поэтому OfficeAssignment определен как единственная сущность типа OfficeAssignment (которая может быть равна null, если у преподавателя нет офиса).

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

Создание сущности OfficeAssignment



Создайте `Models\OfficeAssignment.cs` со следующим содержанием:

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
;
```

```

        namespace ContosoUniversity.Models
    {
        public class OfficeAssignment
        {
            [Key]
            public int InstructorID { get; set; }

            [MaxLength(50)]
            [Display(Name = "Office Location")]
            public string Location { get; set; }

            public virtual Instructor Instructor { get; set; }
        }
    }

```

Атрибут Key

Между сущностями `Instructor` и `OfficeAssignment` существует связь один-к-нулю-или-единице. Назначенный офис существует только в сочетании с назначенным преподавателем, поэтому его первичный ключ также является внешним ключом на сущность `Instructor`. Entity Framework, однако, не может автоматически распознать `InstructorID` как первичный ключ поскольку имя данного свойства не соответствует конвенциям именования первичных ключей, не равно `ID` и `classNameID`. Поэтому атрибутом `Key` указывается, что данный атрибут является первичным ключом.

```

[Key]
public int InstructorID { get; set; }

```

Атрибут `Key` можно использовать также для той ситуации, когда у вас уже есть первичный ключ, но вы хотите назвать свойство иначе чем `classNameID` или `ID`.

Instructor Navigation Property

Сущность `Instructor` nullable `OfficeAssignment` navigation property (преподаватель может не иметь офиса), а сущность `OfficeAssignment` имеет не-nullable `Instructor` navigation property (офис не может существовать без преподавателя). Когда `Instructor` имеет ассоциированную с ним сущность `OfficeAssignment`, оба имеют друг на друга ссылки в своих navigation property.

Продолжение читайте в следующей части статьи...

--

Это перевод оригинальной статьи [Creating a More Complex Data Model for an ASP.NET MVC](#)

Аррисаион. Благодарим за помощь в переводе Александра Белоцерковского.



blogs.msdn.com



<http://blogs.msdn.com/b/vyunev/archive/2011/10/07/asp-net-mvc-p6.aspx>



<http://goo.gl/uvEu1>

