

<b>Введение .....</b>	<b>3</b>
Обзор разрабатываемого приложения .....	3
<b>Lab 1. Разработка базовой основы Web приложения .....</b>	<b>4</b>
Упражнение 1. Создание Web приложения .....	4
Упражнение 2. Создание базы данных .....	7
Упражнение 3. Работа с моделью данных.....	11
<b>Lab 2. Применение EntityDataSource.....</b>	<b>18</b>
Упражнение 1. Добавление и настройка элемента EntityDataSource .....	18
Упражнение 2. Настройка правила: разрешить удаление.....	21
Упражнение 3. Применение GridView для чтения и обновления данных .....	24
Упражнение 4. Настройка EntityDataSource для улучшения производительности.....	27
Упражнение 5. Отображение данных с помощью свойства навигации (Navigation Property) .....	28
Упражнение 6. Применение DetailsView для вставки данных .....	29
Упражнение 7. Отображение данных в списке Drop-Down .....	30
<b>Lab 3. Фильтрация, упорядочивание и группирование данных .....</b>	<b>32</b>
Упражнение 1. Настройка свойств элемента EntityDataSource для более наглядного отображения данных .....	34
Упражнение 2. Реализация поиска данных .....	39
<b>Lab 4. Работа со связанными данными .....</b>	<b>43</b>
Упражнение 1. Отображение и обновление связанных данных в элементе GridView .....	43
Упражнение 2. Отображение связанных данных в отдельном элементе управления.....	47
Упражнение 3. Применение события "Selected" EntityDataSource для отображения связанных данных .....	50
<b>Lab 5. Работа со связанными данными (продолжение).....</b>	<b>53</b>
Упражнение 1. Добавление сущности с отношением к другой сущности.....	54
Упражнение 2. Применение отношения многое ко многим.....	56
<b>Lab 6. Реализация наследования Table-per-Hierarchy .....</b>	<b>60</b>
Упражнение 1. Добавление производных сущностей Instructor и Student.....	61
Упражнение 2. Применение сущностей Instructor и Student .....	66
<b>Lab 7. Использование хранимых процедур.....</b>	<b>71</b>

Упражнение 1. Создание хранимых процедур в базе данных.....	71
Упражнение 2. Размещение хранимых процедур в модели данных.....	74
Упражнение 3. Применение хранимых процедур .....	78
<b>Lab 8. Применение функциональности Dynamic Data для форматирования и валидации данных .....</b>	<b>80</b>
Упражнение 1. Применение элементов DynamicField и DynamicControl .....	80
Упражнение 2. Добавление правил валидации и форматирования .....	83

## Введение

Руководство составлено по материалам [Tom Dykstra](#).

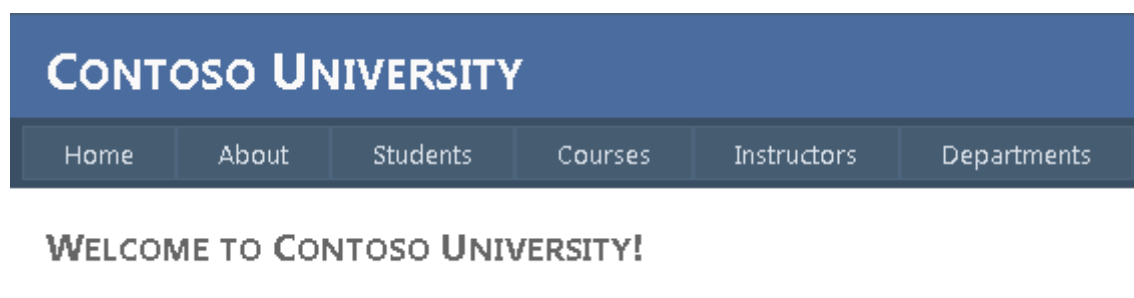
В данном руководстве рассматривается пример веб приложения Contoso University, в котором демонстрируется создание Web-приложения на основе ASP.NET Web Forms, с использованием Entity Framework 4.0 и Visual Studio 2010.

Итоговый пример можно посмотреть в папке *ASP.NET Web Forms Application Using Entity Framework 4.0 Database First*.

Существуют три способа, которые можно использовать для работы с данными в Entity Framework: *Database First*, *Model First*, и *Code First*. В этом руководстве применяется *Database First*: на основе готовой базы Entity Framework создает в Visual Studio модель данных, включающей в себя классы и свойства, которые соответствуют объектам базы данных – таблицам и столбцам. Модель базы данных хранится в XML файле с расширением .edmx. Дизайнер Entity Framework предоставляет графический интерфейс для отображения и редактирования модели.

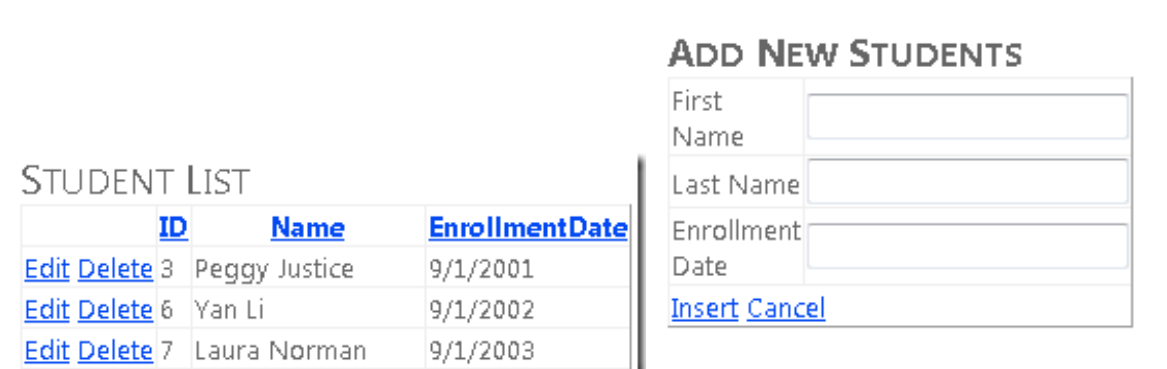
### Обзор разрабатываемого приложения

В результате упражнений этого руководства вы построите вебсайт вымышленного университета Contoso:



Пользователи сайта смогут просмотреть список студентов, обновить данные о студентах, курсах и ввести другую служебную информацию.

Представленные далее рисунки показывают, как это можно сделать.



## COURSES BY DEPARTMENT

Select a Department

CourseID	Title	Credits
1050	Chemistry	4
1061	Physics	4

## COURSES BY NAME

Enter a course name

Department	CourseID	Title	Credits
Economics	4041	Macroeconomics	3
Economics	4022	Microeconomics	3

## INSTRUCTORS

	ID	Name	Hire Date	Office Assignment
<a href="#">Edit</a> <a href="#">Select</a>	1	Abercrombie, Kim	3/11/1995	17 Smith
<a href="#">Edit</a> <a href="#">Select</a>	4	Fakhouri, Fadi	8/6/2002	29 Adams
<a href="#">Edit</a> <a href="#">Select</a>	5	Harui, Roger	7/1/1998	37 Williams
<a href="#">Edit</a> <a href="#">Select</a>	18	Zheng, Roger	2/12/2004	143 Smith
<a href="#">Edit</a> <a href="#">Select</a>	25	Kapoor, Candace	1/15/2001	57 Adams
<a href="#">Edit</a> <a href="#">Select</a>	27	Serrano, Stacy	6/1/1999	271 Williams
<a href="#">Edit</a> <a href="#">Select</a>	31	Stewart, Jasmine	10/12/1997	131 Smith
<a href="#">Edit</a> <a href="#">Select</a>	32	Xu, Kristen	7/23/2001	203 Williams
<a href="#">Edit</a> <a href="#">Select</a>	34	Van Houten, Roger	12/7/2000	213 Smith

## COURSE DETAILS

ID	2030
Title	Poetry
Credits	2
Department	English
Location	
URL	http://www.finearts

## COURSES TAUGHT

	ID	Title	Department
<a href="#">Select</a>	2030	Poetry	English

## STUDENT GRADES

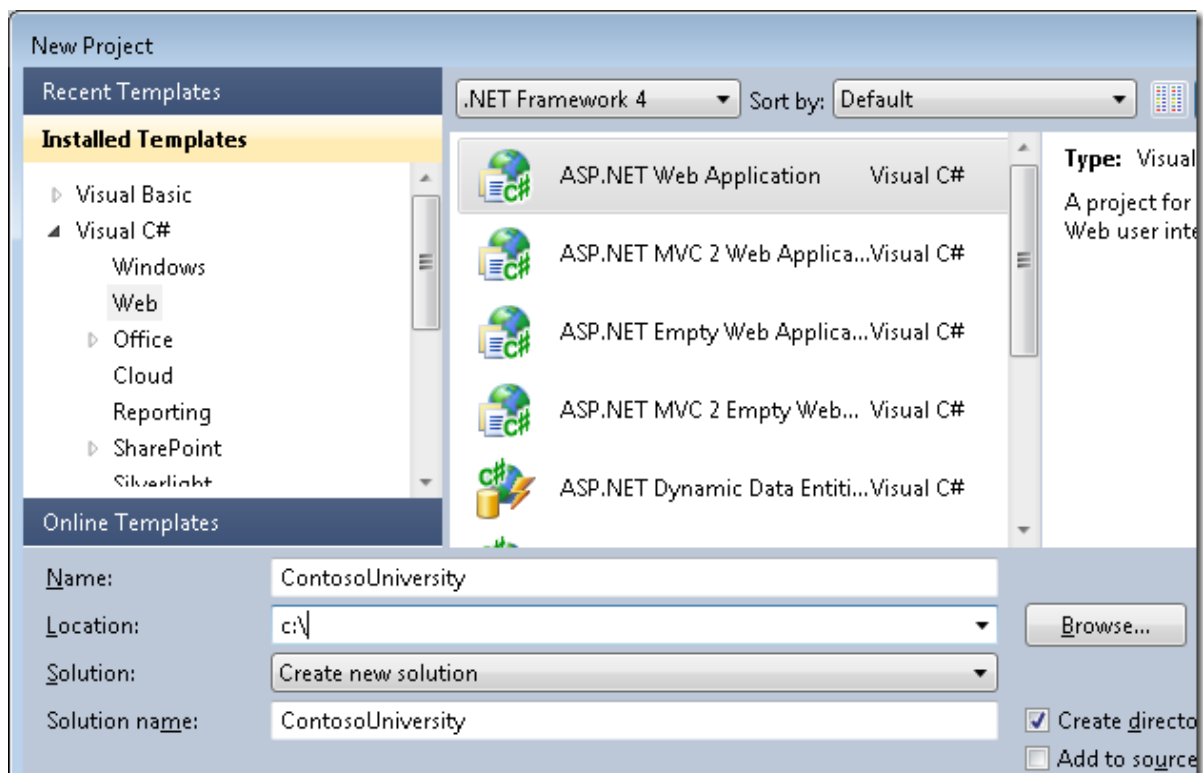
ID	Name	Grade
2	Barzdukas, Gytis	3.50
3	Justice, Peggy	4.00

## Lab 1. Разработка базовой основы Web приложения

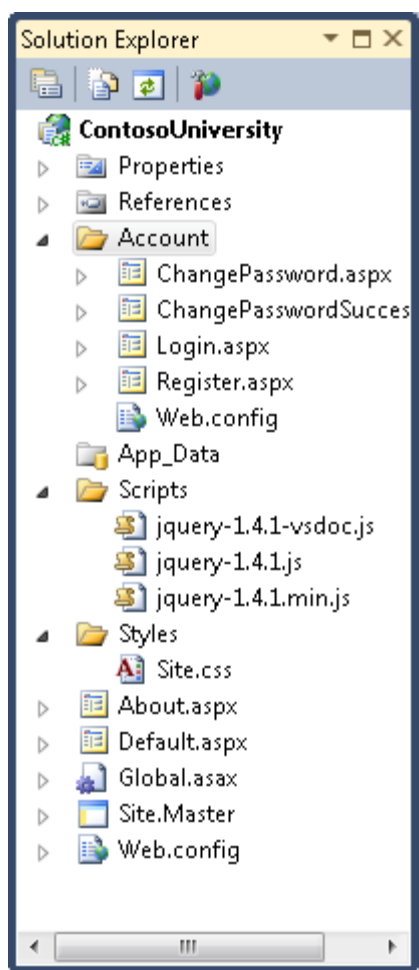
В этой лабораторной работе вы создадите основу web приложения, добавите базу данных и создадите модель данных.

### Упражнение 1. Создание Web приложения

1. Откройте Visual Studio и создайте новый проект приложения ASP.NET Web на основе шаблона **ASP.NET Web Application** (расположение выберите на свое усмотрение):



На основе этого шаблона будет создан проект web-приложения, который включает основные страницы, стили CSS и главную (master) страницу:



2. Откройте файл *Site.Master* и измените "My ASP.NET Application" на "Contoso University".

```
<h1>
    Contoso University
</h1>
```

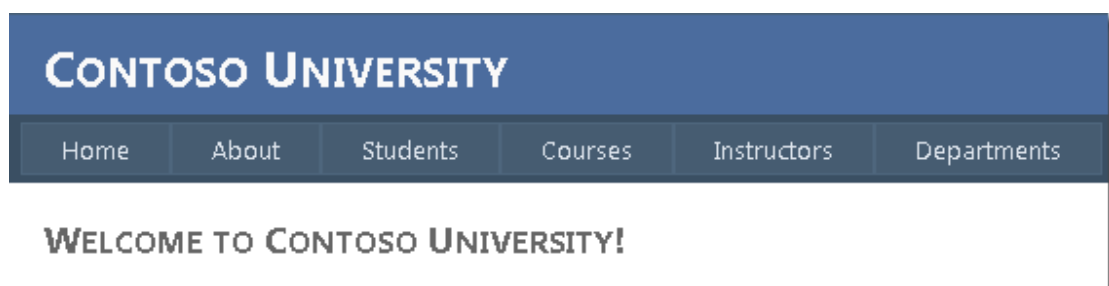
3. Найдите элемент *Menu* с именем *NavigationMenu* и добавьте элементы меню, для открытия страниц, которые будут добавлены в проект далее:

```
<asp:Menu ID="NavigationMenu" runat="server" CssClass="menu"
    EnableViewState="false" IncludeStyleBlock="false" Orientation="Horizontal">
    <asp:Menu ID="NavigationMenu" runat="server" CssClass="menu"
    EnableViewState="false"
        IncludeStyleBlock="false" Orientation="Horizontal">
        <Items>
            <asp:MenuItem NavigateUrl="~/Default.aspx" Text="Home" />
            <asp:MenuItem NavigateUrl="~/About.aspx" Text="About" />
            <asp:MenuItem NavigateUrl="~/Students.aspx" Text="Students">
                <asp:MenuItem NavigateUrl="~/StudentsAdd.aspx" Text="Add Students" />
            </asp:MenuItem>
            <asp:MenuItem NavigateUrl="~/Courses.aspx" Text="Courses">
                <asp:MenuItem NavigateUrl="~/CoursesAdd.aspx" Text="Add Courses" />
            </asp:MenuItem>
            <asp:MenuItem NavigateUrl="~/Instructors.aspx" Text="Instructors">
                <asp:MenuItem NavigateUrl="~/InstructorsCourses.aspx" Text="Course
Assignments" />
            <asp:MenuItem NavigateUrl="~/OfficeAssignments.aspx" Text="Office
Assignments" />
            </asp:MenuItem>
            <asp:MenuItem NavigateUrl="~/Departments.aspx" Text="Departments">
                <asp:MenuItem NavigateUrl="~/DepartmentsAdd.aspx" Text="Add
Departments" />
            </asp:MenuItem>
        </Items>
    </asp:Menu>
```

4. Откройте страницу *Default.aspx* и измените содержимое заголовка элемента *Content* с именем *BodyContent* а остальную разметку удалите:

```
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
    <h2>
        Welcome to Contoso University!
    </h2>
</asp:Content>
```

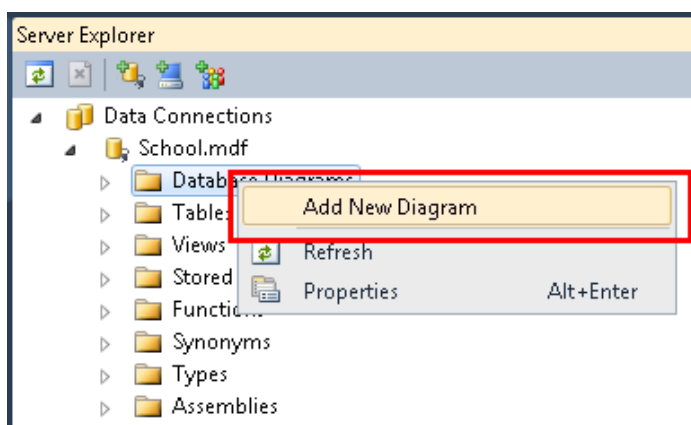
5. Постойте и запустите приложение. Загрузится сайт и отобразится стартовая страница (см. рис). Изучите созданное меню.



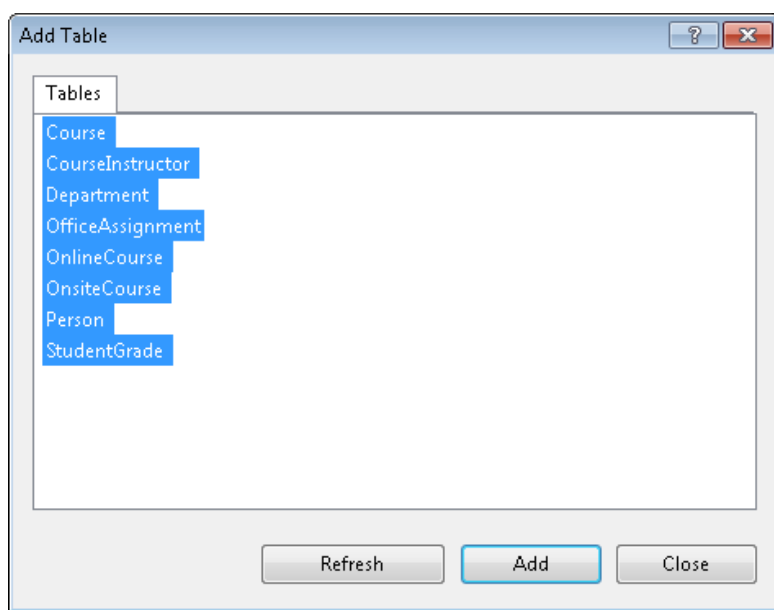
## Упражнение 2. Создание базы данных

В этом упражнении Вы используете дизайнер Entity Framework для автоматического создания модели данных на основе уже существующей база данных *School*. Эта база данных и файл сценария ее создания (если возникнет необходимость в этом) находится в папке *ASP.NET Web Forms Application Using Entity Framework 4.0 Database First*.

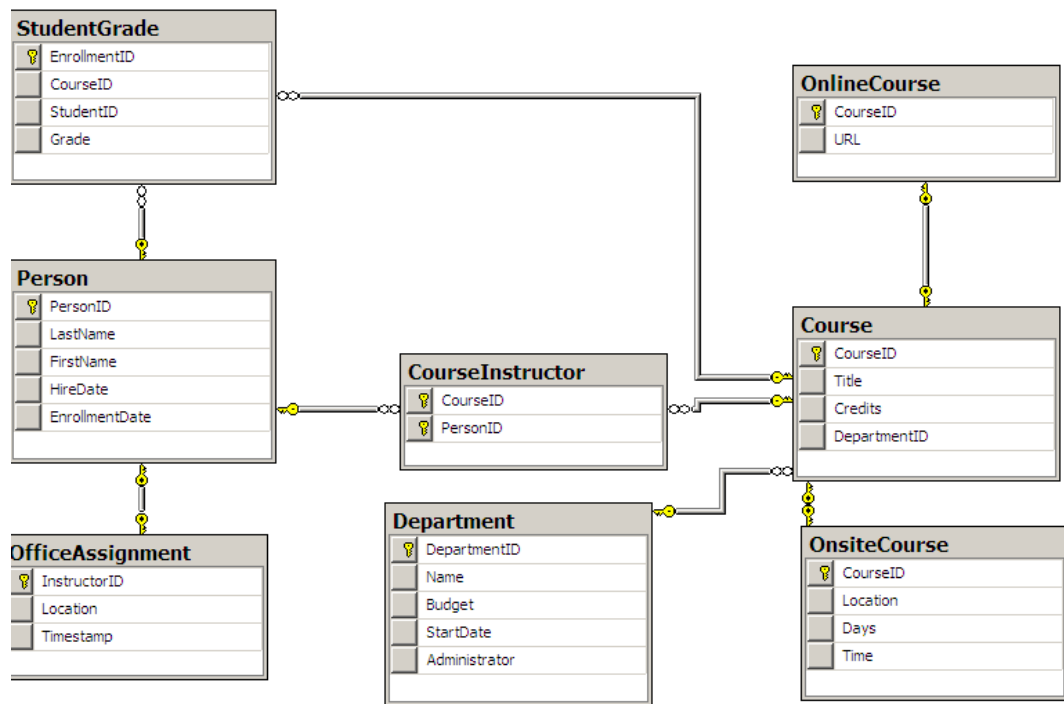
1. Для добавления в проект базы данных выберите в контекстном меню папки *App\_Data* команду **Add → Existing Item**, затем найдите файл базы данных *School.mdf* и добавьте его в папку проекта.
2. Постройте приложение.
3. Создайте схему базы данных, для этого в обозревателе сервера (**Server Explorer**) раскройте последовательно папки **Подключение данных (Data Connections)** и *School.mdf* затем в контекстном меню папки **Схема базы данных (Database Diagrams)** выберите команду **Добавить новую диаграмму (Add New Diagram)**.



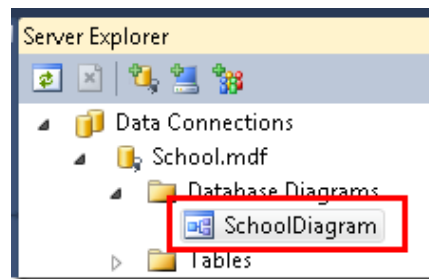
4. В окне **Добавить таблицу (Add Table)** выберите все таблицы и нажмите **Добавить (Add)**.



SQL Server создаст диаграмму, которая показывает таблицы, столбцы и отношения между таблицами (см. рис).



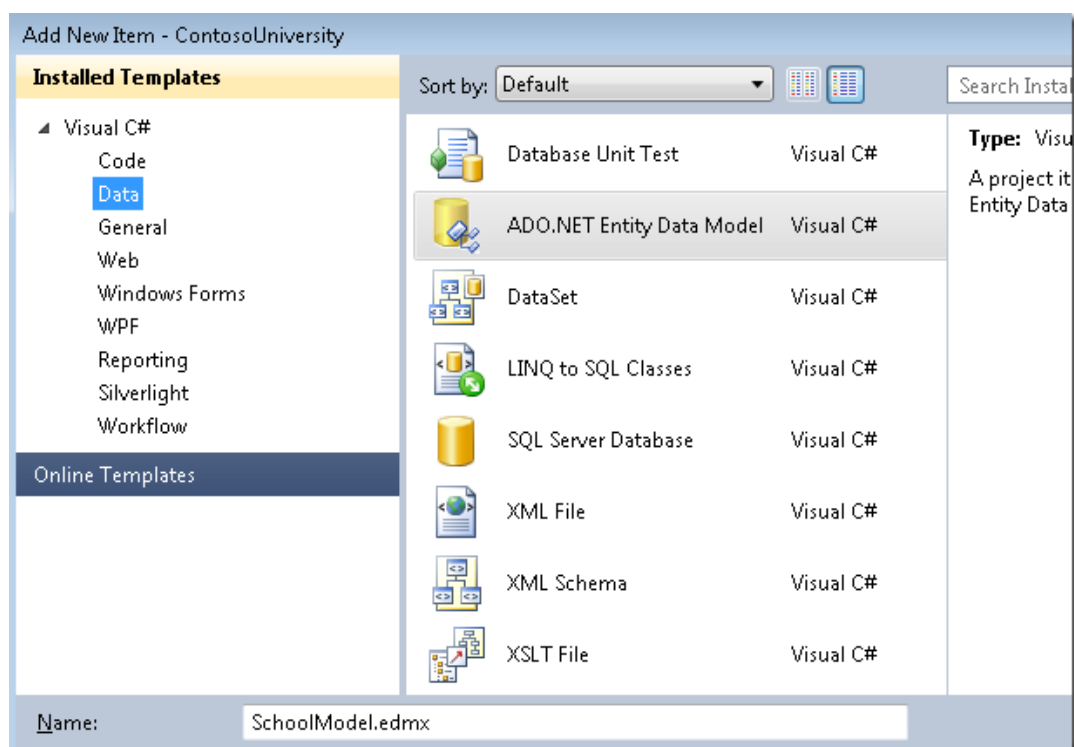
5. Сохраните диаграмму как "SchoolDiagram" и закройте ее. В папке диаграмм сохранится созданная диаграмма.



## Создание модели данных Entity Framework

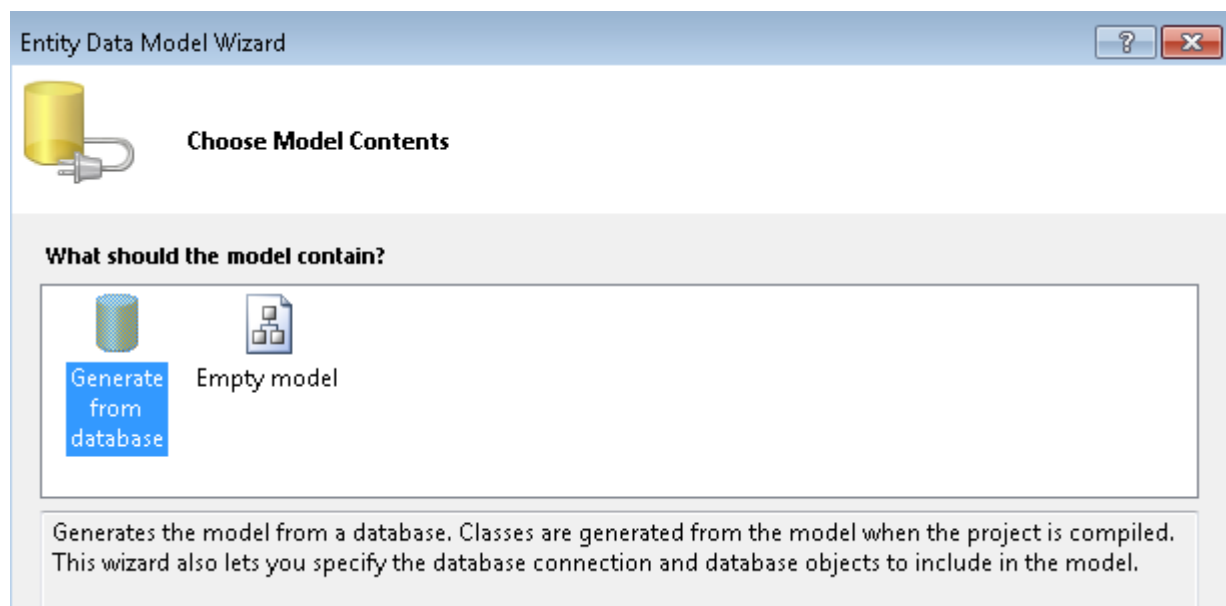
1. Создайте в корне проекта папку для хранения модели, назовите ее *DAL* (будем понимать, как *Data Access Layer*).
2. Добавьте в папку *DAL* требуемую модель, для этого
  - a. в контекстном меню папки выберите **Добавить (Add)**, далее **Создать элемент (New Item)**,
  - b. в окне добавления нового элемента (см. рис) в категории шаблонов выберите **Данные (Data)**, затем **модель ADO.NET EDM (ADO.NET Entity Data Model)**, имя укажите *SchoolModel.edmx* и нажмите **Добавить (Add)**.



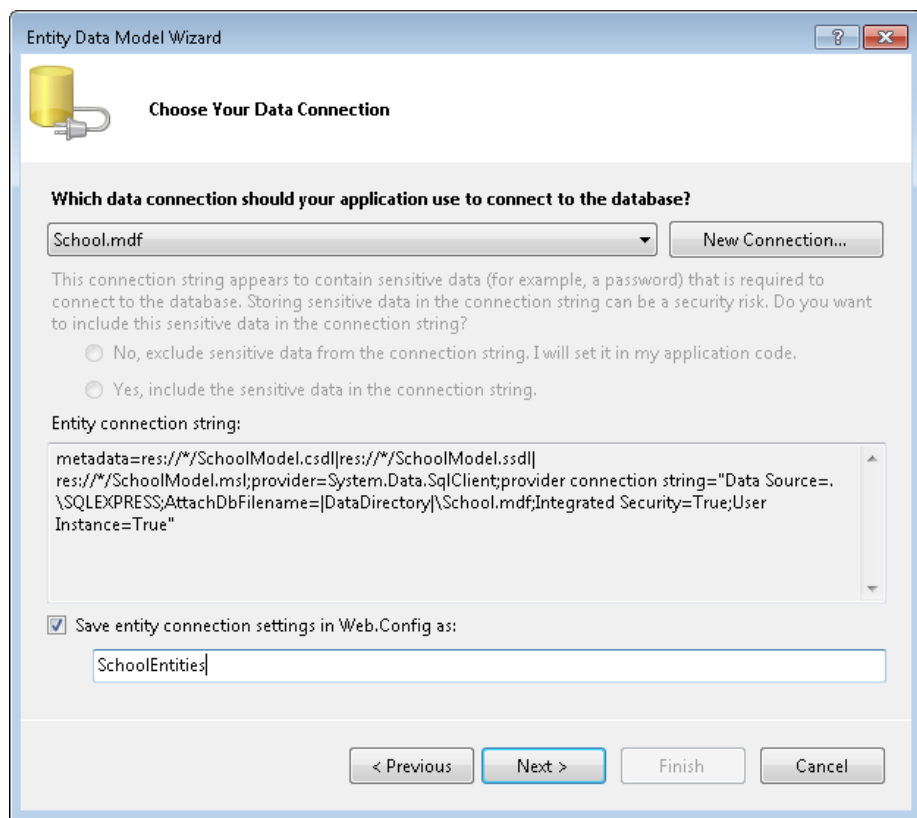


Запустится мастер построения моделей **Entity Data Model Wizard**.

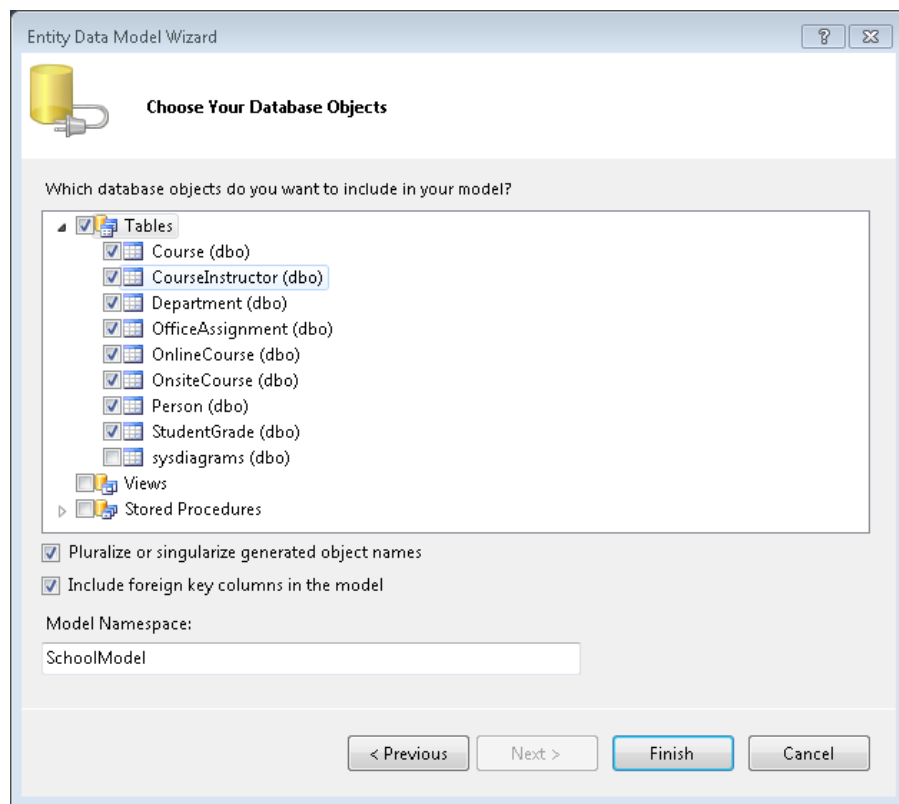
3. На первом шаге мастера оставьте опцию **Создать из базы данных (Generate from database)** отмеченную по умолчанию. Нажмите **Далее (Next)**.



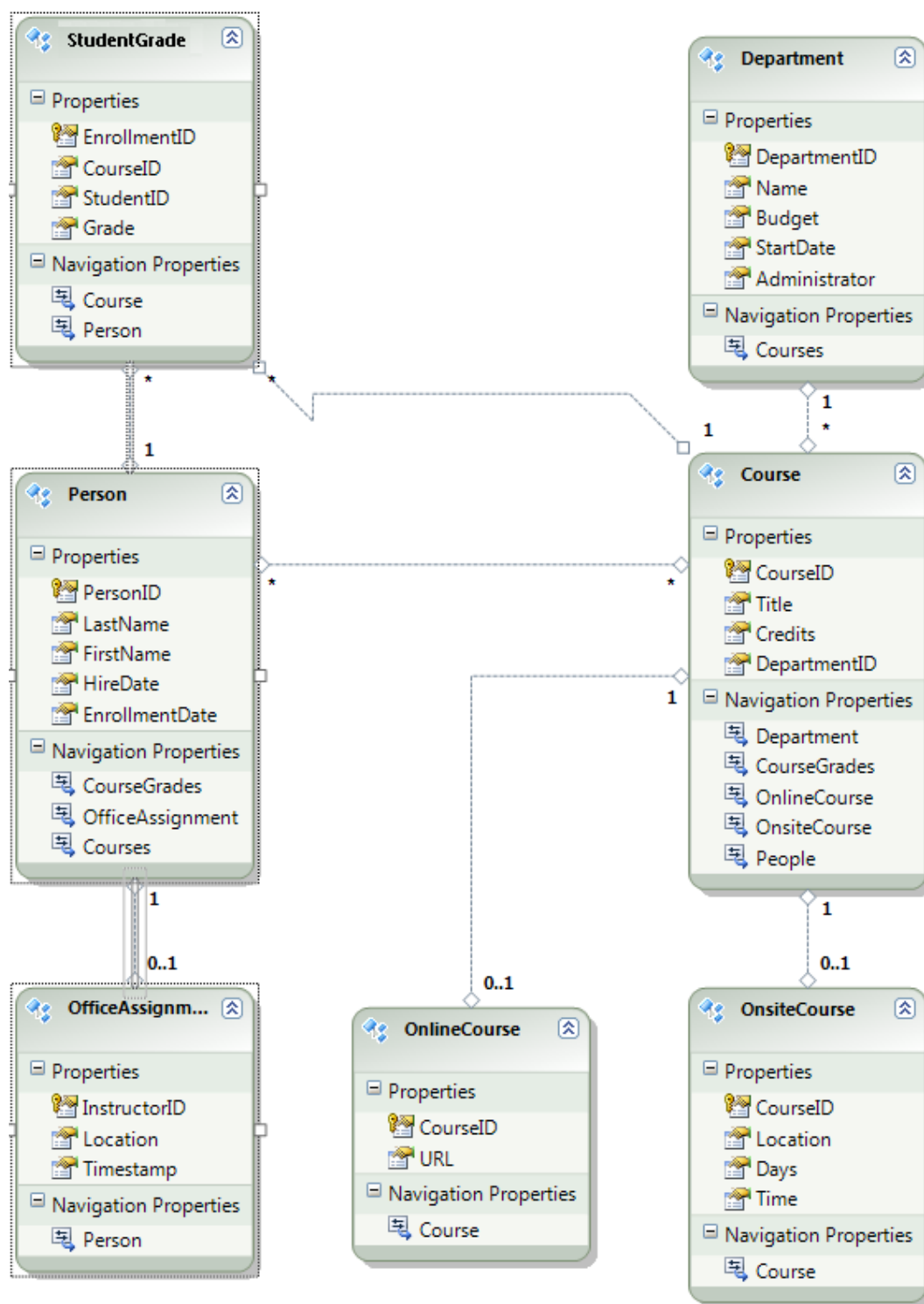
4. На шаге **Choose Your Data Connection** в списке соединения укажите базу данных School (см. рис) и сохраните подключение в файле *Web.config* как **SchoolEntities**, нажмите **Далее (Next)**.



5. На шаге **Choose Your Database Objects** выберите все таблицы кроме sysdiagrams (она была созданная ранее), установите флажки формирования имен и включения столбцов внешних ключей, нажмите **Готово (Finish)**.



После построения модели Visual Studio отобразит графическое представление объектов (сущностей) Entity Framework objects (entities), которые соответствуют таблицам базы данных (см. рис).



### Упражнение 3. Работа с моделью данных

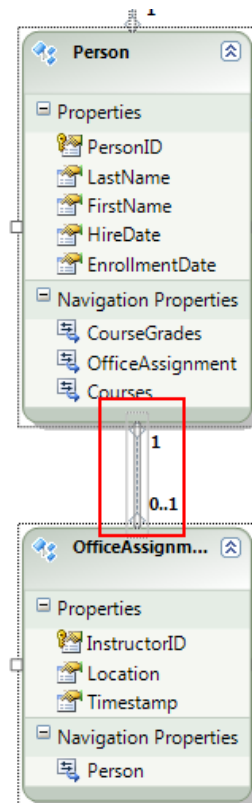
В этом упражнении Вы подробно изучите построенную модель, внесете в нее изменения и познакомитесь с различными способами просмотра модели.

#### Изучение модели данных Entity Framework

Обратите внимание, что диаграмма сущностей очень похожа на схему базы данных, первое отличие состоит в добавлении символов в конце каждой ассоциации, которые указывают тип ассоциации.

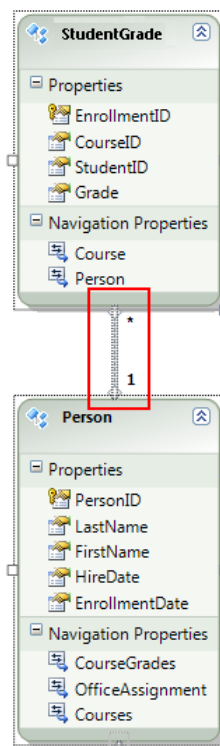
1. Изучите типы ассоциации:

- *one-to-zero-or-one* ассоциация представляется как "1" и "0..1".



В этом случае сущность `Person` может ассоциироваться с сущностью `OfficeAssignment`, а может и не ассоциироваться. А вот сущность `OfficeAssignment` должна иметь отношение с сущностью `Person`. Другими словами, инструктор может быть назначен на должность, но это необязательно, а вот в офис может быть назначен только один инструктор.

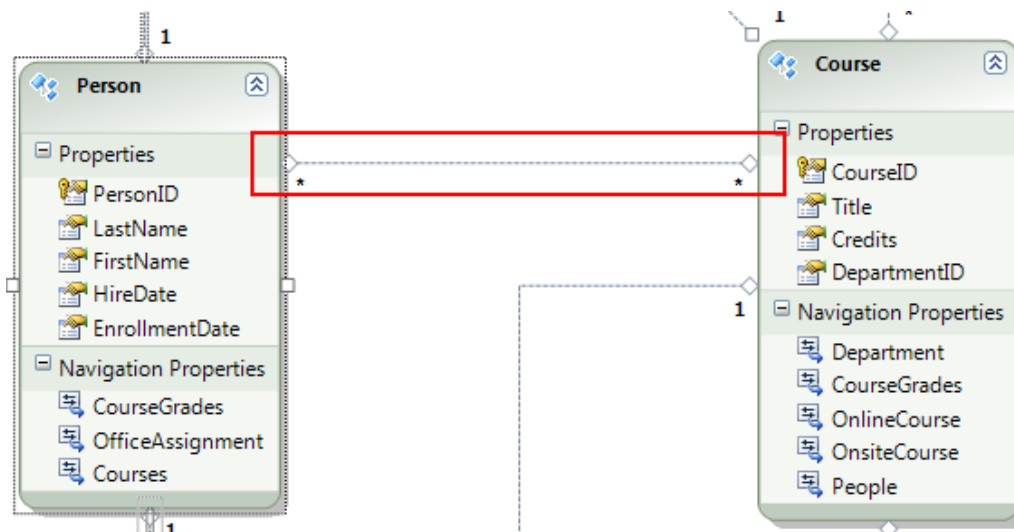
- *one-to-many* ассоциация представлена как "1" и "\*".



В этом случае сущность `Person` может иметь ассоциативную связь с сущностью `StudentGrade`, а в свою очередь, сущность `StudentGrade` должна быть связана только с одним экземпляром сущности `Person`. Сущность `StudentGrade` представляет обучающие курсы, и если студент поступил на обучение, но нет для него обучающего курса, то свойство `Grade` примет значение `null`. Каждый конкретный обучающий курс относится только к одному студенту.

Другими словами, студент может не быть зарегистрирован вообще ни в каком курсе, может быть обучающимся по одному курсу или может быть обучаться по нескольким курсам.

- many-to-many ассоциация представляется как "\*" и "\*".

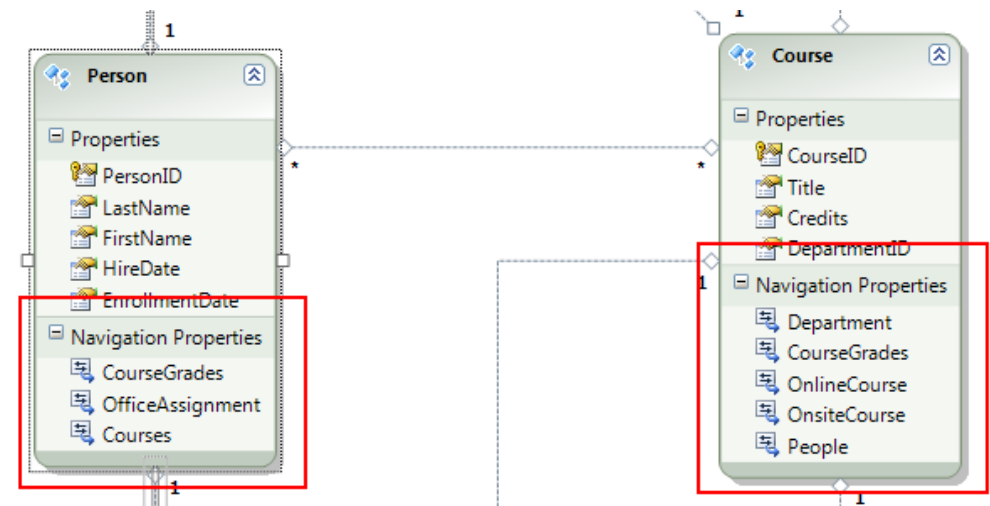


В этом случае `Person` может быть связан с сущностью `Course`, и обратное также верно: сущность `Course` может быть связана с сущностью `Person`.

Другими словами, преподаватель может вести несколько курсов, и курс может преподаваться несколькими преподавателями. В этой базе данных, эти отношения распространяется только на преподавателей, он не связывает студентов с курсами. Студенты связаны с курсами таблицей `StudentGrades`.

2. Обратите внимание на второе отличие между схемой базы данных и моделью данных: дополнительный раздел *Свойства навигации* (**Navigation Properties**) для каждого объекта.

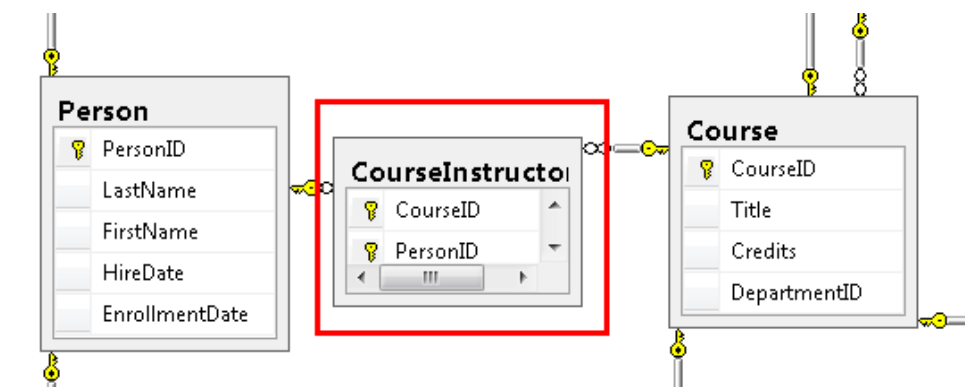
Свойство навигации содержит ссылки на связанные сущности. Например, свойство `Courses` в сущности `Person` содержит коллекцию всех курсов, которые связаны с этим человеком (см. рис).



Yet another difference between the database and data model is the absence of the `CourseInstructor` association table that's used in the database to link the `Person` and `Course` tables in a many-to-many relationship. The navigation properties enable you to get related `Course` entities from the `Person` entity and related `Person` entities from the `Course` entity, so there's no need to represent the association table in the data model.

3. Еще одно отличие между базой данных и моделью данных является отсутствие таблицы ассоциации `CourseInstructor`, которая используется в базе данных, чтобы связать таблицы `Person` и `Course` между которыми отношение многие-ко-многим.

Навигационные свойства позволяют получить связанные объекты `Course` и `Person`, так что нет никакой необходимости представлять таблицу ассоциации в модели данных (см. рис).

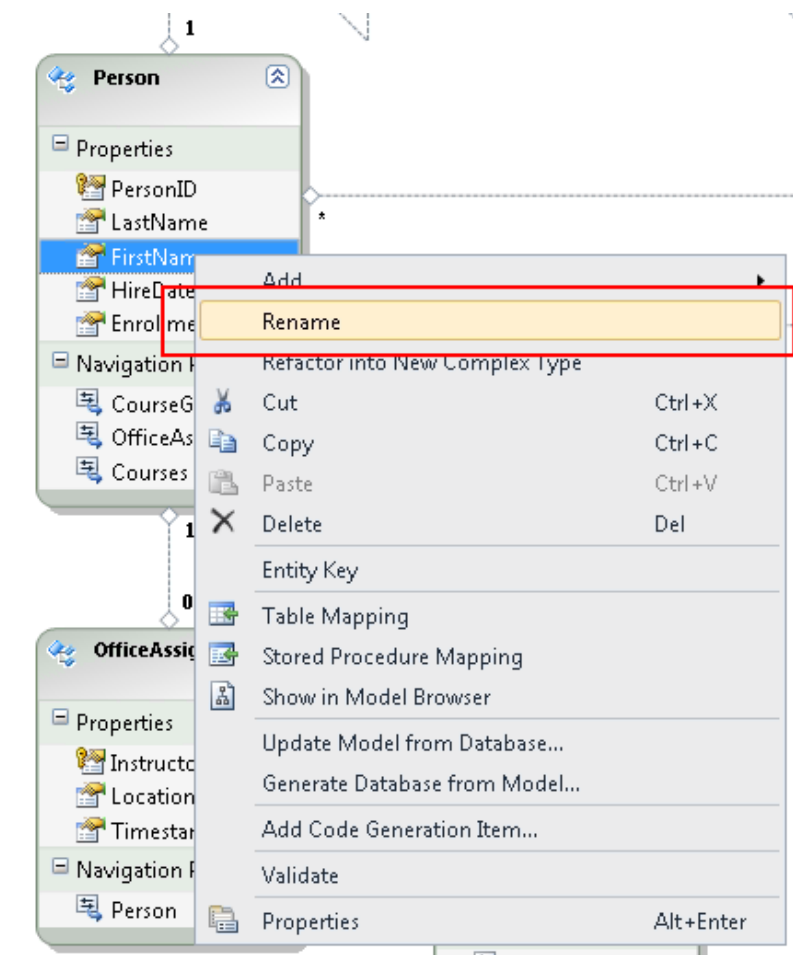


### Изменение модели данных

Созданная модель данных может быть изменена. Изменения эти, как правило, не носят существенный характер и служат для уточнения модели.

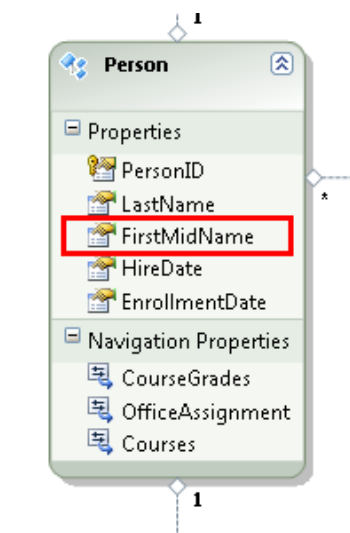
Например, предположим, что столбец `FirstName` таблицы `Person` на самом деле содержит и имя человека, и его отчество. В силу различных причин изменить саму базу данных нет возможности. Вы можете изменить имя свойства `FirstName` в модели данных, оставляя его эквивалент без изменений в базе данных.

1. В дизайнере модели откройте контекстное меню свойства **FirstName** в сущности `Person`, и выберите **Переименовать (Rename)**.



2. Введите новое имя "FirstMidName".

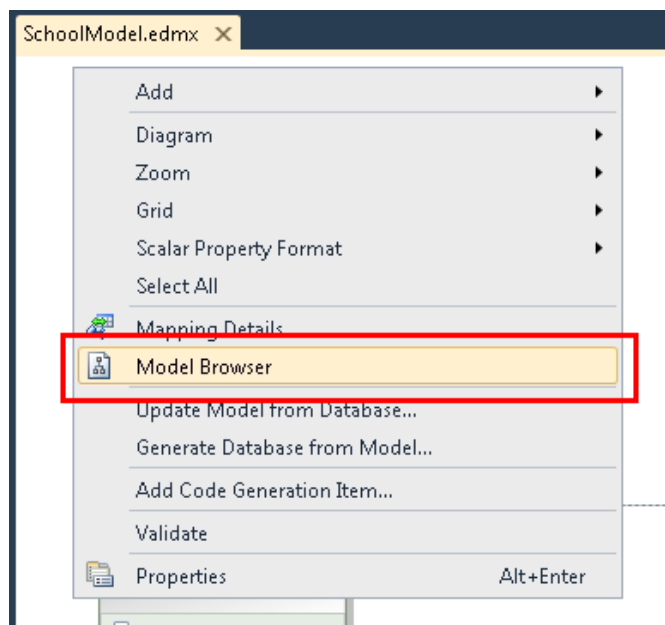
Теперь в коде можно будет обращаться по этому имени.



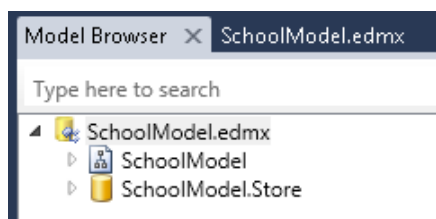
### Просмотр моделей

Среда разработки предоставляет удобный способ для просмотра структуры базы данных, структуры модели данных и отображение между ними.

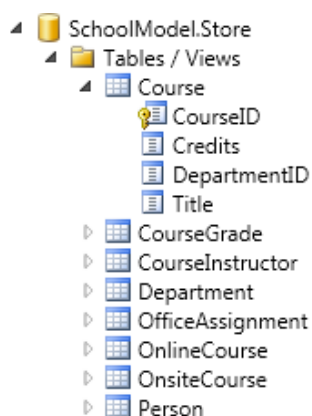
1. Щелкните правой кнопкой мыши пустую область дизайнера, а затем нажмите **Обозреватель моделей (Model Browser)**.



На панели обозревателя моделей (**Model Browser**) отобразится дерево моделей. Узел **SchoolModel** содержит структуру модели данных, а узел **SchoolModel.Store** – структуру базы данных.

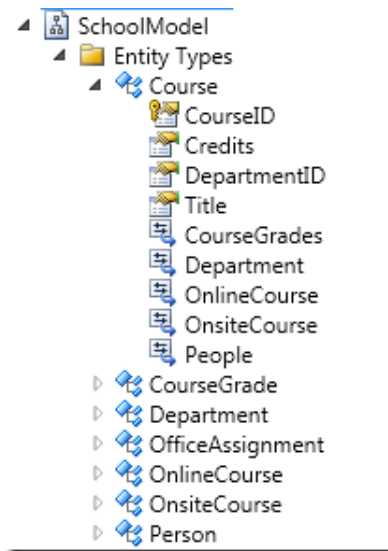


2. Раскройте узел **SchoolModel.Store**, далее раскройте **Таблицы/Представления (Tables / Views)** и просмотрите список таблиц, раскройте таблицу **Course** и изучите перечень столбцов.

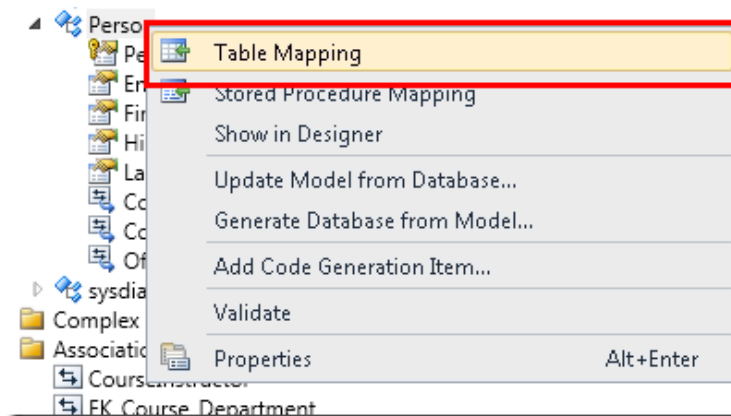


3. Раскройте узел **SchoolModel**, далее раскройте **Типы Сущностей (Entity Types)**, затем раскройте узел **Course** и просмотрите его содержимое.

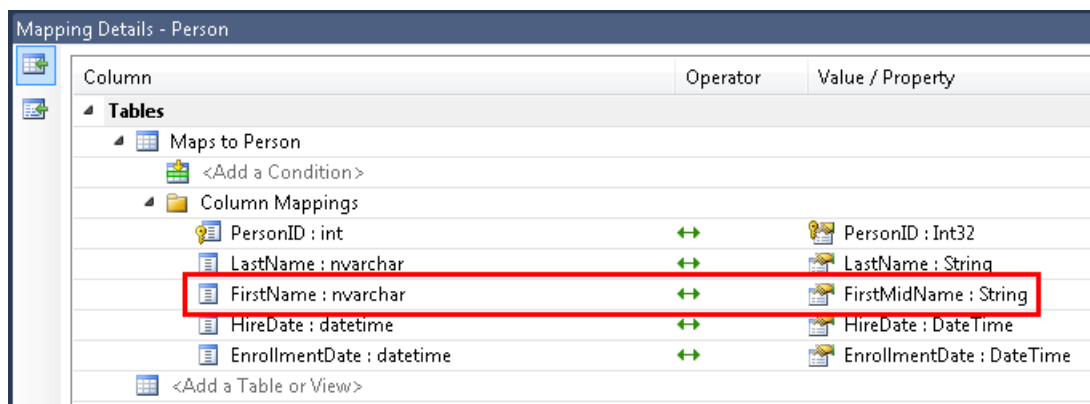




4. Проверьте внесенные Вами изменения в модели данных (свойство **FirstMidName**). Это можно сделать, просто просмотрев соответствующие узлы обозревателя, но удобнее для этой цели использовать таблицу сопоставления.
5. В обозревателе моделей в контекстном меню сущности `Person` выберите команду **Таблицы сопоставления (Table Mapping)**.



6. В окне панели **Сведения о сопоставлении (Mapping Details)** найдите в базе данных столбец `FirstName` и проверьте, что он сопоставляется со свойством `FirstMidName`, которое было переименовано в модели данных.



Платформа Entity Framework использует XML для хранения информации о базе данных, модели данных, и отображений между ними. Файл `SchoolModel.edmx` представляет собой XML-файл, который содержит эту информацию.

Дизайнер представляет модель данных в графическом формате, но вы можете также просмотреть файл как XML.

7. В контекстном меню файла EDMX (в обозревателе решений) нажмите **Открыть с помощью (Open With)** и выберите **редактор (текстовый) XML (XML (Text) Editor)**.

Следует учитывать при просмотре модели, что дизайнер и редактор XML это только два различных способа открытия модели и нельзя работать с одним и тем же файлом в двух режимах открытия.

8. Постройте проект. Возможные изменения в модели данных не доступны для дизайнера, пока проект не будет построен.

В результате выполненных упражнений Вы создали веб-сайт, базу данных, и модель данных. В следующей работе вы начнете оперировать с данными, используя модель данных и элемент управления ASP.NET `EntityDataSource`.

## Lab 2. Применение EntityDataSource

В этой лабораторной работе Вы познакомитесь с элементом управления `EntityDataSource`, который обеспечивает удобную работу с моделью данных Entity Framework. Вы создадите элементы `GridView` для отображения и редактирования данных, `DetailsView` для добавления новых студентов и `DropDownList` для выбора кафедр, который вы будете использовать в дальнейшем для отображения связанных с ними обучающих курсов.

### STUDENT LIST

	Name	EnrollmentDate	Number of Courses
<a href="#">Edit</a> <a href="#">Delete</a>	Abercrombie, Kim		0
<a href="#">Edit</a> <a href="#">Delete</a>	Barzdukas, Gytis	9/1/2005	2
<a href="#">Edit</a> <a href="#">Delete</a>	Justice, Peggy	9/1/2001	2

### ADD NEW STUDENTS

FirstMidName	John
LastName	Smith
EnrollmentDate	1/1/2011
<a href="#">Insert</a> <a href="#">Cancel</a>	

### COURSES BY DEPARTMENT

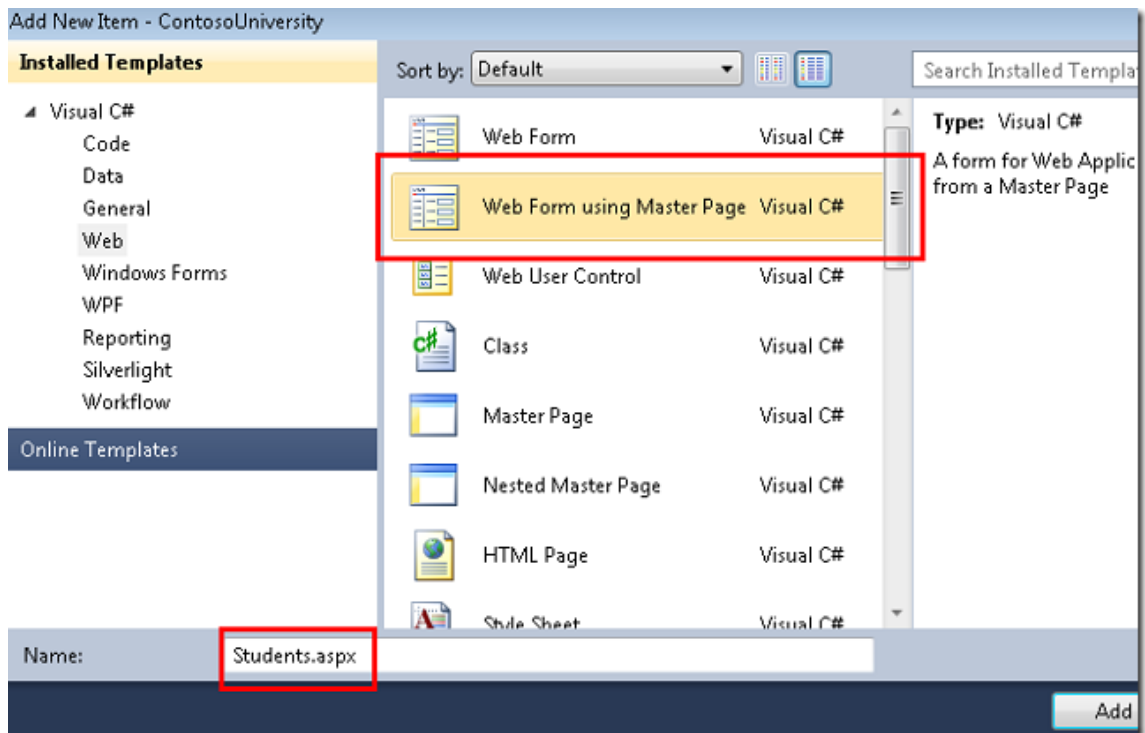
Select a department:

Engineering	▼
Engineering	
English	
Economics	
Mathematics	

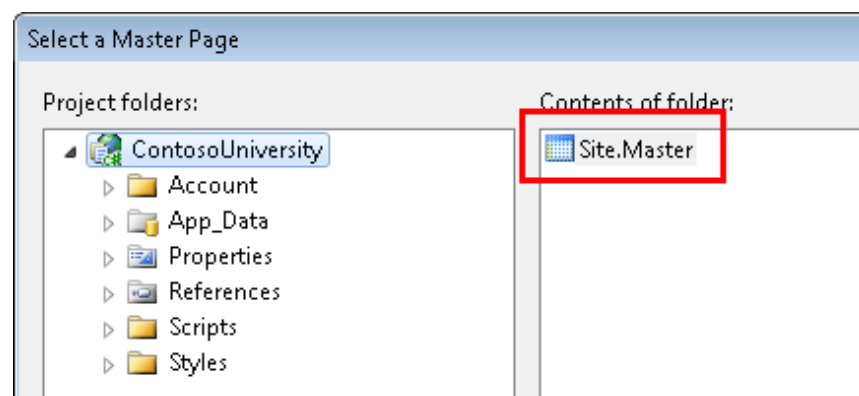
### Упражнение 1. Добавление и настройка элемента EntityDataSource

В этом упражнении вы настроите элемент `EntityDataSource` для чтения объектов `Person` из множества `People`.

1. Добавьте в проект новую web страницу с помощью шаблона **Веб-форма, использующая главную страницу (Web Form using Master Page)**, и присвойте ей имя *Students.aspx*.



2. Укажите *Site.Master* как мастер страницы. All of the pages you create for these tutorials will use this master page.



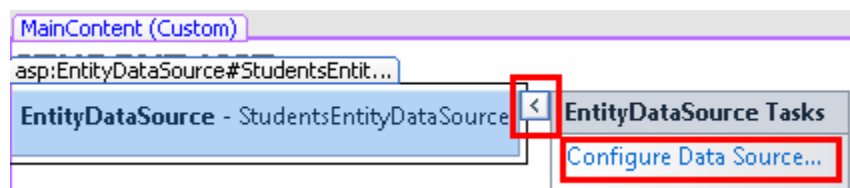
3. В коде разметки новой страницы добавьте текст Student List стиля заголовка h2 в элемент содержимого Content с именем Content2:

```
<asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">
  <h2>
    Student List</h2>
</asp:Content>
```

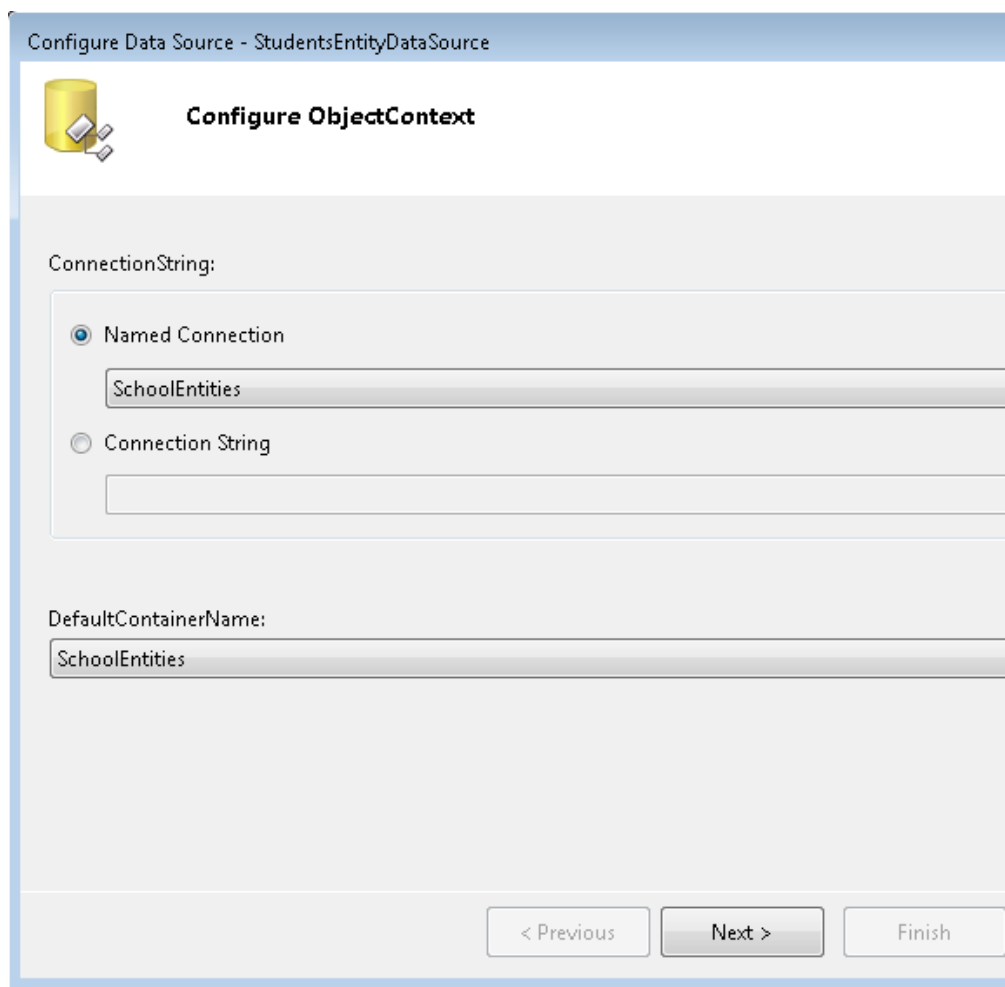
4. Откройте страницу *Students.aspx* в обозревателе и убедитесь, что добавленная строка отображается как надо.
5. Откройте панель инструментов (**Toolbox**) и из вкладки **Данные (Data)** перенесите элемент *EntityDataSource* на страницу после тега *</h2>* и измените значение свойства *ID* на *StudentsEntityDataSource*:

```
<asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">
  <h2>
    Student List</h2>
  <asp:EntityDataSource ID="StudentsEntityDataSource" runat="server">
  </asp:EntityDataSource>
</asp:Content>
```

6. Откройте страницу *Students.aspx* в режиме дизайнера (**Design**), кликните смарт тег элемента источника данных и выберите **Настроить источник данных (Configure Data Source)** для запуска мастера **Настройка источника данных (Configure Data Source)**.



7. На первом шаге мастера **Настройка ObjectContext (ConfigureObjectContext)** выберите для переключателя **Именованное соединение (Named Connection)** значение **SchoolEntities** и для **DefaultContainerName** тоже должно быть значение **SchoolEntities**. Нажмите **Далее (Next)**.



*Замечание:* Если вы получили сообщение об ошибке загрузки данных для строки соединения, то возможно забыли построить проект перед настройкой элемента источника данных.

8. На шаге **Настройка выбора данных (Configure Data Selection)** выберите для набора **EntitySetName** значение **People**. В поле **Select** поставьте флажок **Select All**. Установите также флажки автоматического обновления и удаления (enable update and delete). Нажмите **Готово (Finish)**.
9. Сохраните изменения.

Configure Data Source - StudentsEntityDataSource

**Configure Data Selection**

EntitySetName:  
People

EntityTypeFilter:  
(None)

Select:

- ☒ Select All (Entity Value)
- ☐ PersonID
- ☐ LastName
- ☐ FirstMidName
- ☐ HireDate
- ☐ EnrollmentDate

☐ Enable automatic inserts

☒ Enable automatic updates

☒ Enable automatic deletes

< Previous    Next >    Finish

10. Откройте страницу *Students.aspx* в режиме разметки и изучите код, добавленный матером.

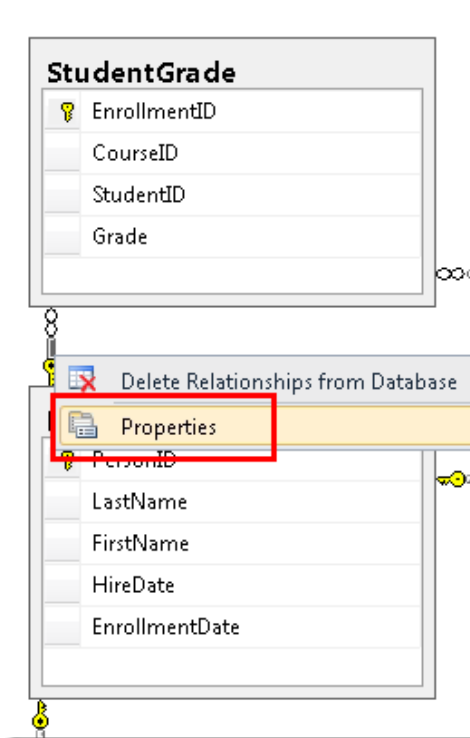
### **Упражнение 2. Настройка правила: разрешить удаление**

В этом упражнении Вы создадите страницу, которая позволит пользователям удалять студентов из таблицы `Person`, которая имеет связи с другими таблицами (`Course`, `StudentGrade` и `OfficeAssignment`).

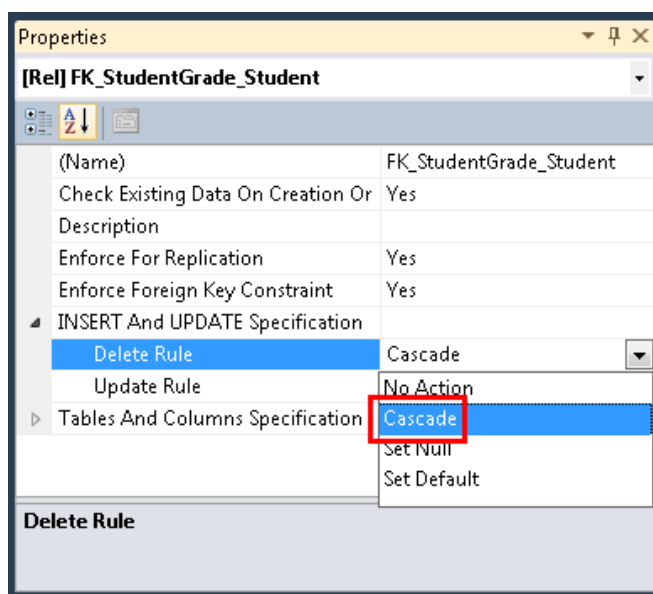
По умолчанию, база данных ограничивает удаление строки в `Person`, если есть связанные строки в одной из других таблиц. Вы можете вручную удалить связанные строки, или лучше настроить базу данных так, чтобы удалять их автоматически, когда вы удалите строку в `Person`.

Для записей о студентах в этом упражнении Вы настроите базу данных для автоматического удаления связанных данных. Так как студенты могут иметь связанные строки только в таблице `StudentGrade`, необходимо настроить только одно из трех отношений.

1. Откройте диаграмму базы данных.
2. В контекстном меню отношения между таблицами `Person` и `StudentGrade` выберите **Свойства (Properties)**.



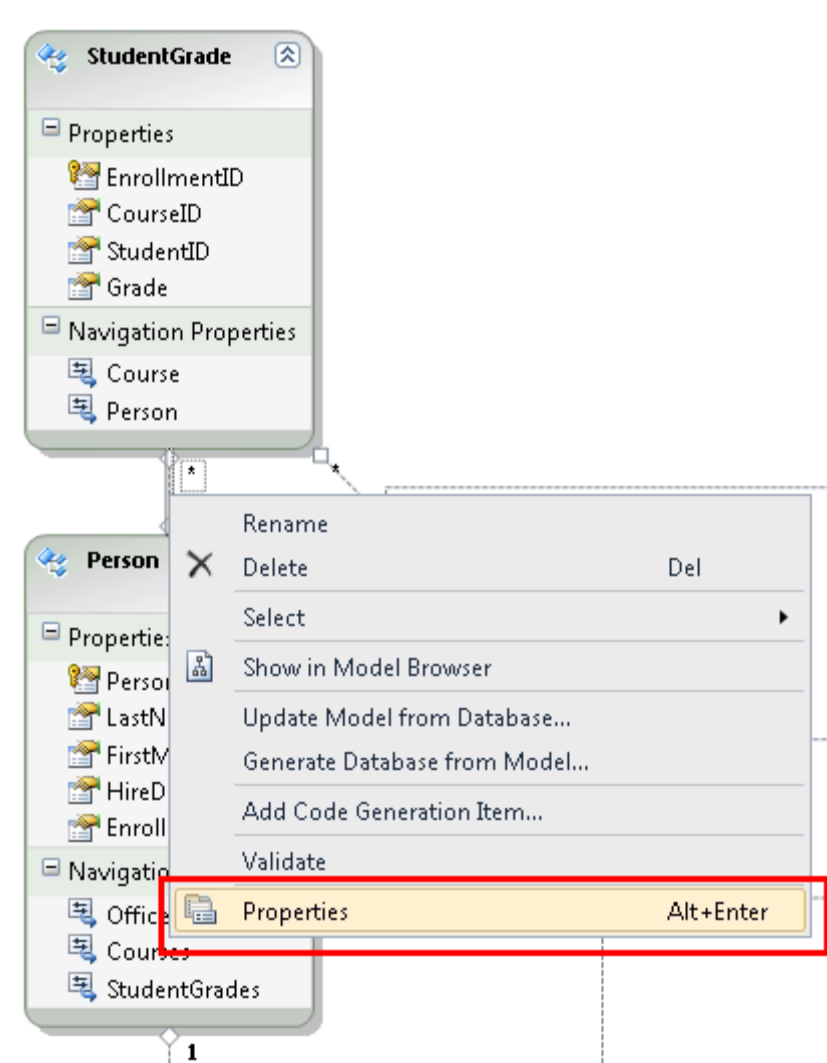
3. В окне свойств раскройте **Спецификация INSERT и UPDATE (INSERT and UPDATE Specification)** и установите для свойства правила удаления (**DeleteRule**) значение **Каскадом (Cascade)**.



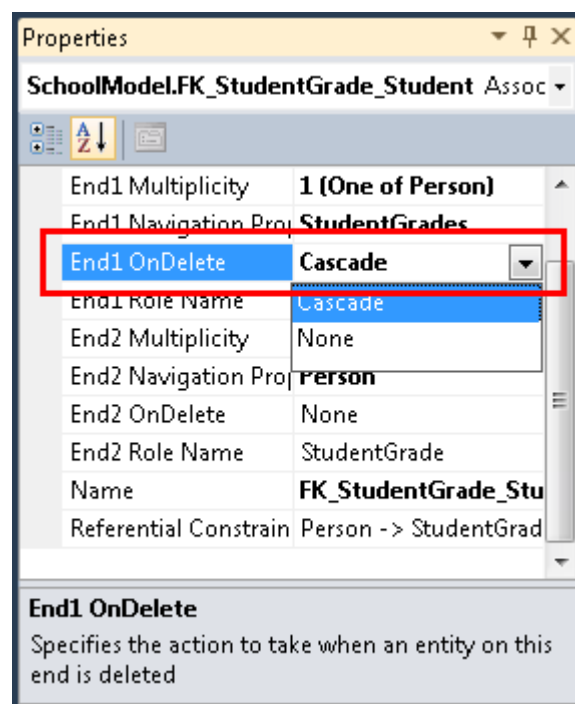
4. Сохраните изменения и закройте диаграмму. Если перед сохранением появится окно с вопросом, хотите ли вы, чтобы обновить базу данных, то согласитесь.

Чтобы убедиться, что модель поддерживает объекты, которые находятся в памяти, синхронизированы с тем, что содержит база данных, вы должны установить соответствующие правила в модели данных.

5. Откройте модель *SchoolModel.edmx* в режиме дизайнера и в контекстном меню связи между Person и StudentGrade, выберите **Свойства (Properties)**.



6. На панели свойств для свойства Событие **OnDelete** элемента **End1 (End1 OnDelete)** установите значение **Cascade**.

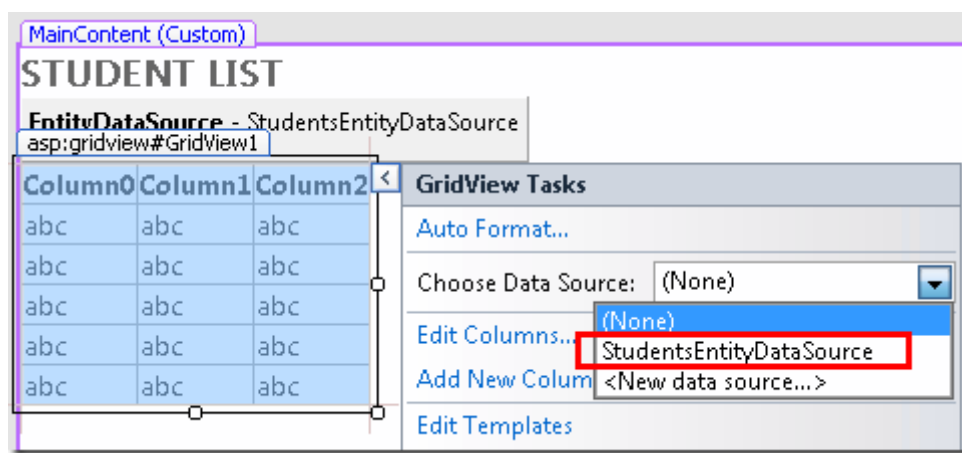


7. Сохраните и закройте файл *SchoolModel.edmx*.
8. Перестройте проект.

### Упражнение 3. Применение GridView для чтения и обновления данных

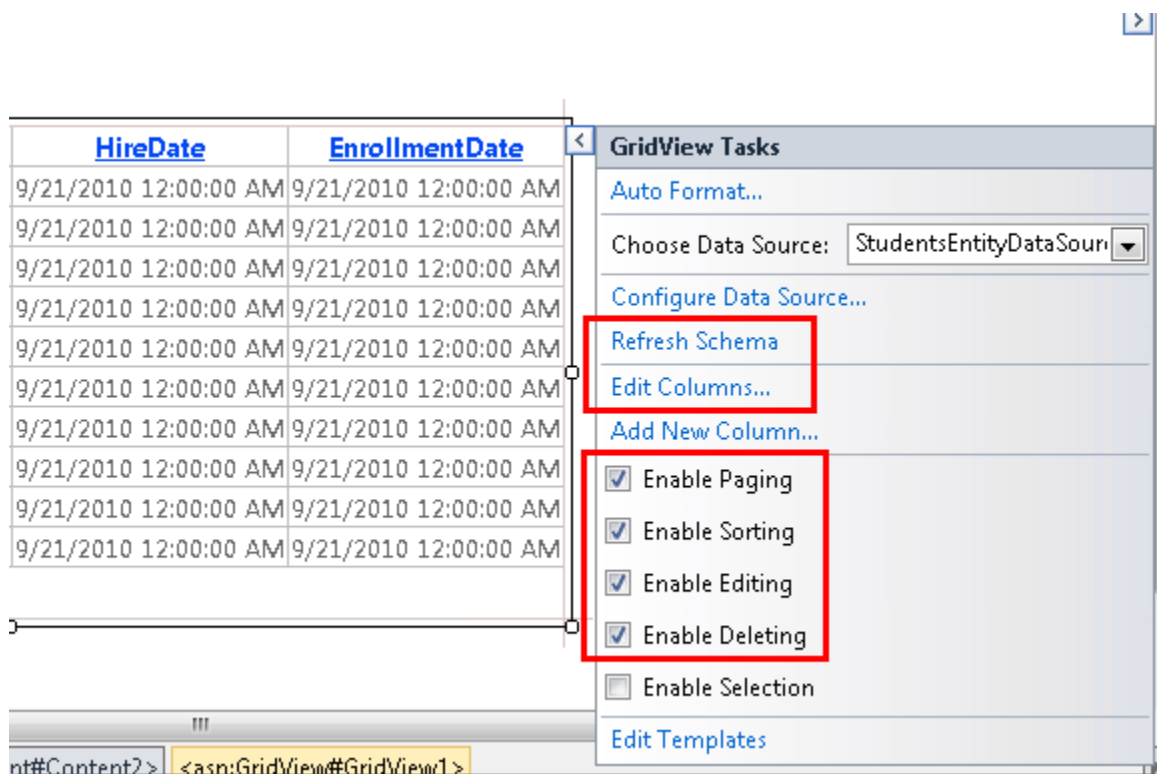
В этом упражнении Вы используете элемент GridView для отображения, обновления или удаления студентов.

1. Откройте файл *Students.aspx* в режиме конструктора.
2. Откройте панель инструментов (**Toolbox**) и из вкладки **Данные (Data)** перенесите элемент GridView справа от элемента EntityDataSource.
3. С помощью окна свойств укажите свойству (ID) значение StudentsGridView.
4. Кликните смарт тег элемента GridView и выберите **StudentsEntityDataSource** в качестве источника данных.

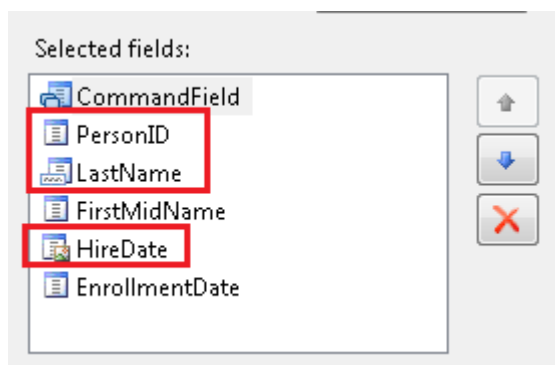


5. Кликните **Обновить схему (Refresh Schema)** (согласитесь если появится вопрос-предупреждение).
6. Установите флажки **Включить постраничный просмотр (Enable Paging)**, **Включить сортировку (Enable Sorting)**, **Включить правку (Enable Editing)** и **Включить удаление (Enable Deleting)**.
7. Кликните **Правка столбцов (Edit Columns)**.

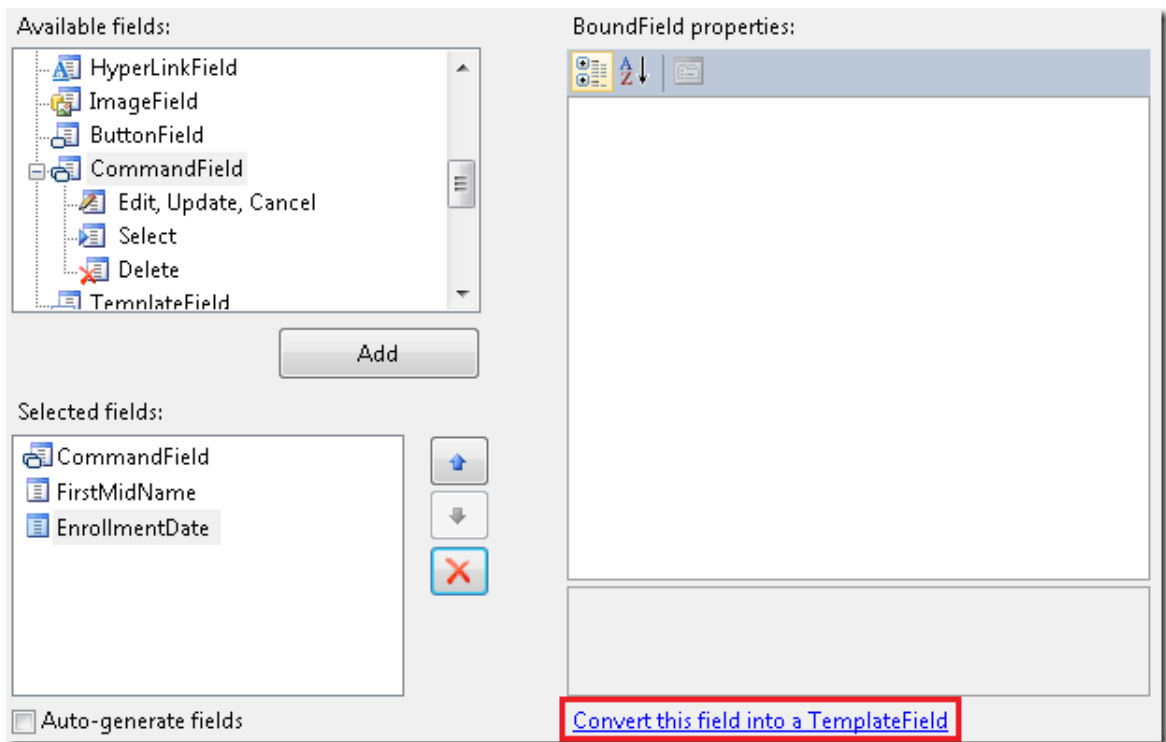




8. В окне **Выбранные поля (Selected fields)** удалите **PersonID**, **LastName** и **HireDate**. Как правило на практике эти столбцы не нужны.



9. Выберите поле **FirstMidName** и кликните **Преобразовать это поле в TemplateField (Convert this field into a TemplateField)**.
10. Сделайте тоже и с полем **EnrollmentDate**.



11. Нажмите **ОК** и переключитесь в режим разметки.

12. Изучите изменения, которые внесены в разметку. Элемент GridView должен представляться следующим образом:

```
<asp:GridView ID="StudentsGridView" runat="server" AllowPaging="True"
    AllowSorting="True" AutoGenerateColumns="False" DataKeyNames="PersonID"
    DataSourceID="StudentsEntityDataSource">
    <Columns>
        <asp:CommandField ShowDeleteButton="True" ShowEditButton="True" />
        <asp:TemplateField HeaderText="FirstMidName"
SortExpression="FirstMidName">
            <EditItemTemplate>
                <asp:TextBox ID="TextBox1" runat="server" Text='<%#
Bind("FirstMidName") %>'></asp:TextBox>
            </EditItemTemplate>
            <ItemTemplate>
                <asp:Label ID="Label1" runat="server" Text='<%#
Bind("FirstMidName") %>'></asp:Label>
            </ItemTemplate>
        </asp:TemplateField>
        <asp:TemplateField HeaderText="EnrollmentDate"
SortExpression="EnrollmentDate">
            <EditItemTemplate>
                <asp:TextBox ID="TextBox2" runat="server" Text='<%#
Bind("EnrollmentDate") %>'></asp:TextBox>
            </EditItemTemplate>
            <ItemTemplate>
                <asp:Label ID="Label2" runat="server" Text='<%#
Bind("EnrollmentDate") %>'></asp:Label>
            </ItemTemplate>
        </asp:TemplateField>
    </Columns>
</asp:GridView>
```

13. Первый столбец после командного поля является шаблонным полем и отображает имя студента. Измените код разметки для более наглядного отображения:

```

        <asp:TemplateField HeaderText="Name" SortExpression="LastName">
            <EditItemTemplate>
                <asp:TextBox ID="LastNameTextBox" runat="server" Text='<%#
Bind("LastName") %>'></asp:TextBox>
                <asp:TextBox ID="FirstNameTextBox" runat="server" Text='<%#
Bind("FirstMidName") %>'></asp:TextBox>
            </EditItemTemplate>
            <ItemTemplate>
                <asp:Label ID="LastNameLabel" runat="server" Text='<%#
Eval("LastName") %>'></asp:Label>,
                <asp:Label ID="FirstNameLabel" runat="server" Text='<%#
Eval("FirstMidName") %>'></asp:Label>
            </ItemTemplate>
        </asp:TemplateField>

```

В режиме отображения два элемента Label отображают имя и фамилию. В режиме редактирования отображаются два текстовых поля для реализации возможности изменения имени и фамилии.

14. Последний столбец является шаблонным полем, который отображает дату зачисления студента на курс. Измените код разметки для более наглядного отображения:

```

        <asp:TemplateField HeaderText="Enrollment Date"
SortExpression="EnrollmentDate">
            <EditItemTemplate>
                <asp:TextBox ID="EnrollmentDateTextBox" runat="server" Text='<%#
Bind("EnrollmentDate", "{0:d}") %>'></asp:TextBox>
            </EditItemTemplate>
            <ItemTemplate>
                <asp:Label ID="EnrollmentDateLabel" runat="server" Text='<%#
Eval("EnrollmentDate", "{0:d}") %>'></asp:Label>
            </ItemTemplate>
        </asp:TemplateField>

```

Обратите внимание на формат "{0,d}".

#### ***Упражнение 4. Настройка EntityDataSource для улучшения производительности***

В этом упражнении Вы внесете изменения в настройку элемента EntityDataSource для улучшения производительности.

```
ConnectionString="name=SchoolEntities" DefaultContainerName="SchoolEntities"
```

1. В коде разметки элемента EntityDataSource удалите атрибуты ConnectionString и DefaultContainerName и замените их на атрибут ContextTypeName="ContosoUniversity.DAL.SchoolEntities".

Код разметки сейчас должен быть похож на следующий (порядок свойств может быть различным):

```

<asp:EntityDataSource ID="StudentsEntityDataSource" runat="server"
    ContextTypeName="ContosoUniversity.DAL.SchoolEntities"
    EnableFlattening="False"
    EntitySetName="People"
    EnableDelete="True" EnableUpdate="True">
</asp:EntityDataSource>

```

- Запустите страницу в браузере и просмотрите список студентов. Имя и фамилия отображаются в одном поле.

**STUDENT LIST**

	<a href="#">Name</a>	<a href="#">EnrollmentDate</a>
<a href="#">Edit</a> <a href="#">Delete</a>	Abercrombie, Kim	
<a href="#">Edit</a> <a href="#">Delete</a>	Barzdukas, Gytis	9/1/2005
<a href="#">Edit</a> <a href="#">Delete</a>	Justice, Peggy	9/1/2001
<a href="#">Edit</a> <a href="#">Delete</a>	Fakhouri, Fadi	
<a href="#">Edit</a> <a href="#">Delete</a>	Harui, Roger	
<a href="#">Edit</a> <a href="#">Delete</a>	Li, Yan	9/1/2002
<a href="#">Edit</a> <a href="#">Delete</a>	Norman, Laura	9/1/2003
<a href="#">Edit</a> <a href="#">Delete</a>	Olivotto, Nino	9/1/2005
<a href="#">Edit</a> <a href="#">Delete</a>	Tang, Wayne	9/1/2005
<a href="#">Edit</a> <a href="#">Delete</a>	Alonso, Meredith	9/1/2002
<a href="#">1</a> <a href="#">2</a> <a href="#">3</a> <a href="#">4</a>		

- Для сортировки в столбце кликните по имени столбца.
- Кликните для правки данных строки по **Edit**. Отобразятся текстовые поля для изменения имени и фамилии.

**STUDENT LIST**

	<a href="#">Name</a>	<a href="#">EnrollmentDate</a>
<a href="#">Edit</a> <a href="#">Delete</a>	Abercrombie, Kim	
<a href="#">Update</a> <a href="#">Cancel</a>	Barzdukas	Gytis 9/1/2005
<a href="#">Edit</a> <a href="#">Delete</a>	Justice, Peggy	
<a href="#">Edit</a> <a href="#">Delete</a>	Fakhouri, Fadi	
<a href="#">Edit</a> <a href="#">Delete</a>	Harui, Roger	
<a href="#">Edit</a> <a href="#">Delete</a>	Li, Yan	9/1/2002
<a href="#">Edit</a> <a href="#">Delete</a>	Norman, Laura	9/1/2003
<a href="#">Edit</a> <a href="#">Delete</a>	Olivotto, Nino	9/1/2005
<a href="#">Edit</a> <a href="#">Delete</a>	Tang, Wayne	9/1/2005
<a href="#">Edit</a> <a href="#">Delete</a>	Alonso, Meredith	9/1/2002
<a href="#">1</a> <a href="#">2</a> <a href="#">3</a> <a href="#">4</a>		

- Проверьте работу кнопки **Delete**. Удалите любую строку, которая имеет дату зачисления.

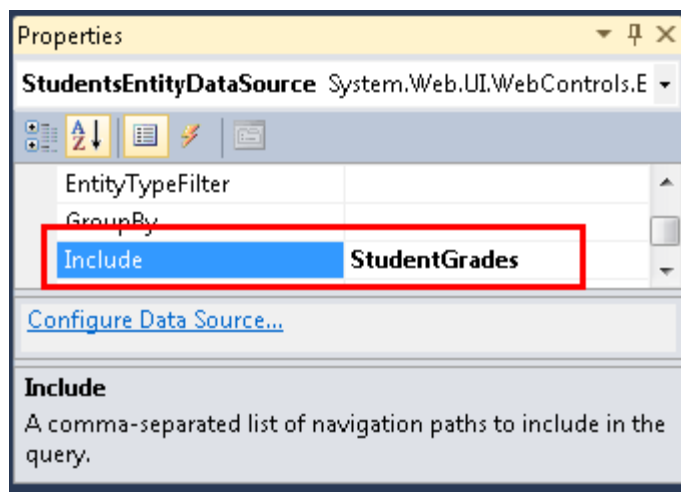
Удаление строк, не содержащих дату зачисления, приведет к ошибке. В дальнейшем эта ошибка будет устранена.

### ***Упражнение 5. Отображение данных с помощью свойства навигации (Navigation Property)***

Свойства навигации предоставляют способ перемещения по ассоциации между типами сущностей. Каждый объект может обладать свойством навигации для каждого отношения, в котором участвует. Свойства навигации позволяют передвигаться по связям и управлять ими в обоих направлениях, а также возвращают ссылочный объект (если кратность равна одному либо нулю или одному) или коллекцию (если кратность больше одного).

В этом упражнении вы добавите возможность вывода количества курсов, на которых обучается студент. Entity Framework предоставляет эту информацию в навигационном свойстве `StudentGrades` сущности `Person`.

1. Откройте файл *Students.aspx* в режиме конструктора.
2. Выделите элемент `StudentsEntityDataSource` и в окне свойств для свойства **Include** установите значение **StudentGrades** (если вы хотите получить несколько свойств навигации, вы можете указать их имена через запятую, например, **StudentGrades, Courses**.)



3. Переключитесь в режим разметки кода. В элементе `StudentsGridView` после последнего элемента `asp:TemplateField` добавьте следующее поле:

```
<asp:TemplateField HeaderText="Number of Courses">
  <ItemTemplate>
    <asp:Label ID="Label1" runat="server" Text="<%#
Eval("StudentGrades.Count") %>"></asp:Label>
  </ItemTemplate>
</asp:TemplateField>
```

В выражении `Eval` используется свойство `StudentGrades` поскольку оно содержит коллекцию, имеющее свойство `Count`, которое можно использовать для отображения количества курсов, в которых студент обучается.

4. Запустите приложение и проверьте, что таблица отображает количество курсов для каждого студента.

STUDENT LIST			
	Name	EnrollmentDate	Number of Courses
<a href="#">Edit</a> <a href="#">Delete</a>	Abercrombie, Kim		0
<a href="#">Edit</a> <a href="#">Delete</a>	Barzdukas, Gytis	9/1/2005	2
<a href="#">Edit</a> <a href="#">Delete</a>	Justice, Peggy	9/1/2001	2

### Упражнение 6. Применение *DetailsView* для вставки данных

В этом упражнении вы создадите страницу с элементом `DetailsView`, который позволит добавлять новых студентов.

1. Добавьте в проект новую веб-страницу, используя главную страницу *Site.Master*. Назовите страницу *StudentsAdd.aspx*.

2. В коде разметки новой страницы добавьте текст Add New Students стиля заголовка h2 в элемент содержимого Content с именем Content2:

```
<asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">
    <h2>
        Add New Students </h2>
</asp:Content>
```

3. Откройте страницу *StudentsAdd.aspx* в обозревателе и убедитесь, что добавленная строка отображается как надо.
4. Откройте страницу *Students.aspx* в коде разметки, скопируйте код элемента *EntityDataSource* и вставьте его в файл *StudentsAdd.asp* и добавьте свойство, разрешающее операцию вставки:

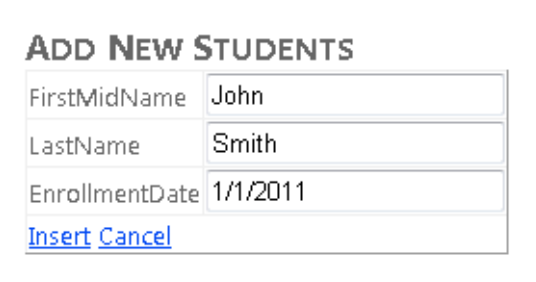
```
EnableInsert="True"
```

5. Добавьте следующий код для использования элемента *DetailsView*:

```
<asp:DetailsView ID="StudentsDetailsView" runat="server"
    DataSourceID="StudentsEntityDataSource" AutoGenerateRows="False"
    DefaultMode="Insert">
    <Fields>
        <asp:BoundField DataField="FirstMidName" HeaderText="First Name"
            SortExpression="FirstMidName" />
        <asp:BoundField DataField="LastName" HeaderText="Last Name"
            SortExpression="LastName" />
        <asp:BoundField DataField="EnrollmentDate" HeaderText="Enrollment Date"
            SortExpression="EnrollmentDate" />
        <asp:CommandField ShowInsertButton="True" />
    </Fields>
</asp:DetailsView>
```

Как и в элементе *GridView* связанные поля элемента *DetailsView* кодируются так, как будто они будут работать для управления данными из базы данных, за исключением того, что они ссылаются на свойства сущностей. В данном случае *DetailsView* используется только для вставки строк, так что включен режим по умолчанию для вставки.

6. Запустите приложение и протестируйте вставку данных студента.



ADD NEW STUDENTS	
FirstMidName	John
LastName	Smith
EnrollmentDate	1/1/2011
<a href="#">Insert</a> <a href="#">Cancel</a>	

7. Откройте список студентов и проверьте, что новый студент успешно добавлен.

### **Упражнение 7. Отображение данных в списке Drop-Down**

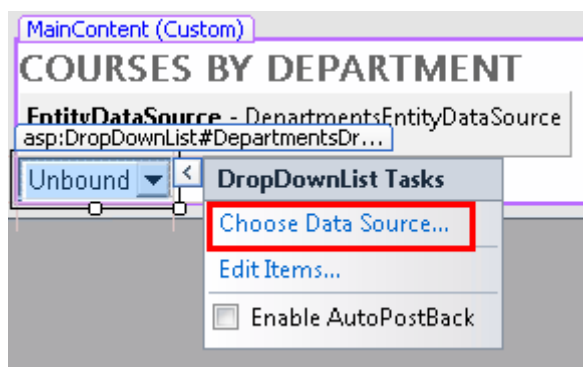
В этом упражнении Вы реализуете связь элемента *DropDownList* с данными с помощью элемента *EntityDataSource*. В следующих упражнениях Вы будете использовать список, чтобы позволить пользователям выбрать кафедру и отобразить курсы, связанные с этой кафедрой.

1. Добавьте в проект новую веб-страницу, используя главную страницу *Site.Master*. Назовите страницу *Courses.aspx*.
2. В коде разметки новой страницы добавьте текст Courses by Department стиля заголовка h2 в элемент содержимого Content с именем Content2:

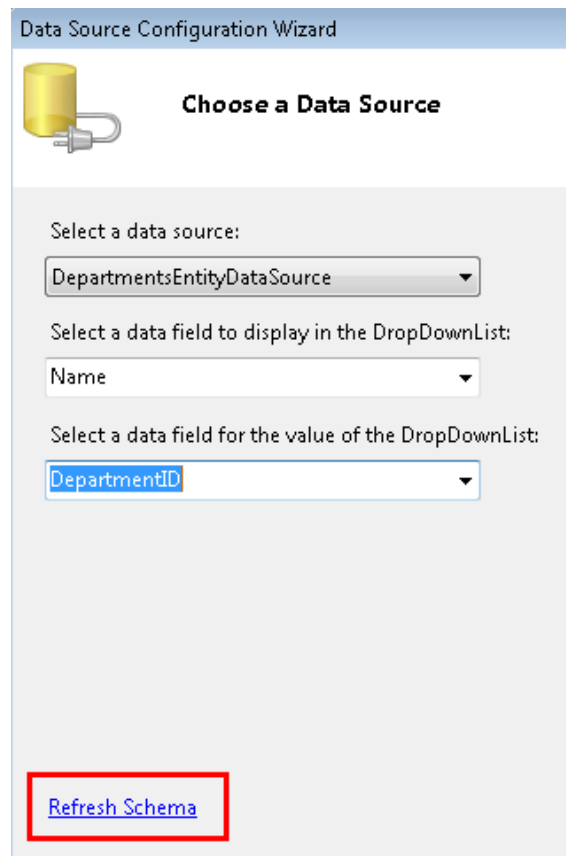
```
<asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">
    <h2>Courses by Department</h2>
</asp:Content>
```

3. В режиме конструктора добавьте элемент `EntityDataSource` и укажите (**ID**) как `DepartmentsEntityDataSource`.
4. Настройте источник данных: на первом шаге мастера **НастройкаObjectContext (ConfigureObjectContext)** выберите для переключателя **Именованное соединение (Named Connection)** значение **SchoolEntities** и для **DefaultContainerName** тоже должно быть значение **SchoolEntities**.
5. На шаге **Настройка выбора данных (Configure Data Selection)** выберите для набора **EntitySetName** значение **DepartmentID**. В поле **Select** укажите только **Departments** и **Name**.

6. Перенесите с панели инструментов элемент `DropDownList`, укажите (**ID**) как `DepartmentsDropDownList`, кликните смарт тег и выберите **Выбрать источник данных (Choose Data Source)** для запуска мастера **DataSource Configuration Wizard**.



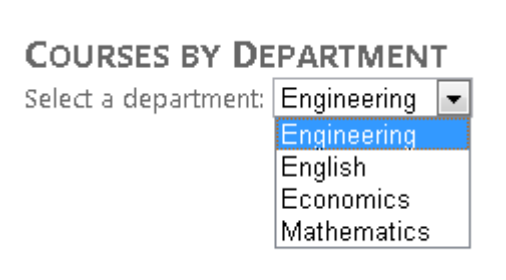
7. На шаге **Выбор источника данных (Choose a Data Source)** выберите **DepartmentsEntityDataSource** как источник данных, кликните **Обновить схему (Refresh Schema)**, и в первом списке укажите поле данных для отображения **Name** и во втором поле для отображения значений **DepartmentID**. Нажмите **OK**.



8. В коде разметки страницы добавьте фразу "Select a department:" прямо перед элементом DropDownList:

```
Select a department:
<asp:DropDownList ID="DepartmentsDropDownList" runat="server"
    DataSourceID="DepartmentsEntityDataSource" DataTextField="Name"
    DataValueField="DepartmentID">
</asp:DropDownList>
```

9. Запустите страницу в обозревателе и протестируйте возможность выбора элемента списка.



В результате выполненных упражнений вы настроили элемент `EntityDataSource`. Работа с этим элементом управления, как правило, мало отличается от работы с другими элементами управления источника данных ASP.NET. Единственное исключение, когда вы хотите получить доступ к навигационным свойствам.

### Lab 3. Фильтрация, упорядочивание и группирование данных

В предыдущей работе вы использовали элемент `EntityDataSource` для отображения и редактирования данных. В этой работе вы реализуете фильтрацию, порядок отображения определенного свойства и группировку данных.



Вы измените страницу *Students.aspx* для фильтрации студентов, сортировки и поиска их по имени, и поиск по имени. Вы также измените страницу *Courses.aspx* для отображения курсов для выбранной кафедры и поиска курсов по названию. Наконец, вы добавите статистику по студентам к странице *About.aspx*.

### STUDENT LIST

	<a href="#">Name</a>	<a href="#">EnrollmentDate</a>	Number of Courses
<a href="#">Edit</a> <a href="#">Delete</a>	Barzdukas, Gytis	9/1/2005	2
<a href="#">Edit</a> <a href="#">Delete</a>	Justice, Peggy	9/1/2001	2
<a href="#">Edit</a> <a href="#">Delete</a>	Li, Yan	9/1/2002	2

### COURSES BY DEPARTMENT

Select a Department

ID	Title	Credits
2021	Composition 3	
2030	Poetry	2
2042	Literature	4

### COURSES BY NAME

Enter a course name

Department	ID	Title	Credits
Economics	4041	Macroeconomics	3
Economics	4022	Microeconomics	3
Economics	4063	new course	5

### STUDENT BODY STATISTICS

Date of Enrollment	Students
9/1/2000	2
9/1/2001	5
9/1/2002	3
1/30/2003	1
9/1/2003	3
9/1/2004	5
9/1/2005	6
1/1/2011	1

### FIND STUDENTS BY NAME

Enter any part of the name

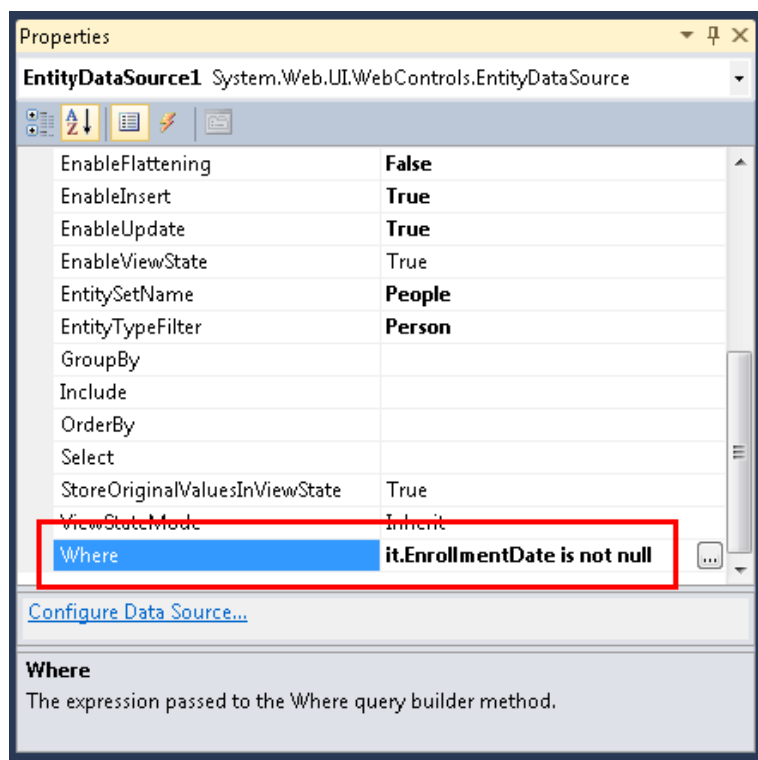
Name	EnrollmentDate
Barzdukas, Gytis	9/1/2005
Justice, Peggy	9/1/2001
Tang, Wayne	9/1/2005

## Упражнение 1. Настройка свойств элемента *EntityDataSource* для более наглядного отображения данных

### Использование свойства "Where" элемента *EntityDataSource* для фильтрации данных

В этой части упражнения вы внесете изменения в элемент *EntityDataSource* для того, чтобы *GridView* мог отображать только студентов, которые зачислены (т.е. имеют даты зачисления).

1. Откройте в режиме конструктора страницу *Students.asp*, созданную в предыдущей работе.
2. Выделите *EntityDataSource*. В окне свойств установите для свойства *Where* значение *it.EnrollmentDate is not null*.



Синтаксис, используемый в свойстве *Where* элемента *EntityDataSource*, соответствует Entity SQL. В выражении *it.EnrollmentDate IS NOT NULL*, слово *it* представляет собой ссылку на сущность, возвращаемую запросом. Таким образом, *it.EnrollmentDate* относится к свойству *EnrollmentDate* человека лица, что возвращает управление *EntityDataSource*.

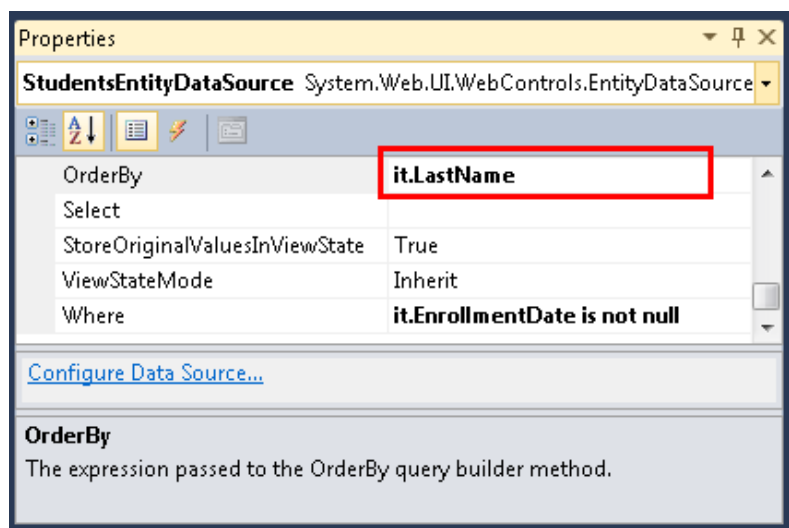
3. Откройте страницу в обозревателе. Будут отображены только те студенты, которые имеют дату зачисления хотя бы на один курс.

#### STUDENT LIST

	Name	EnrollmentDate	Number of Courses
<a href="#">Edit</a> <a href="#">Delete</a>	Barzdukas, Gytis	9/1/2005	2
<a href="#">Edit</a> <a href="#">Delete</a>	Justice, Peggy	9/1/2001	2
<a href="#">Edit</a> <a href="#">Delete</a>	Li, Yan	9/1/2002	2

### Использование свойства "OrderBy" элемента EntityDataSource для определения порядка отображения данных

4. Определите порядок отображения студентов в списке, например, сначала фамилия, потом имя: для свойства **OrderBy** элемента EntityDataSource установите значение `it.LastName`.

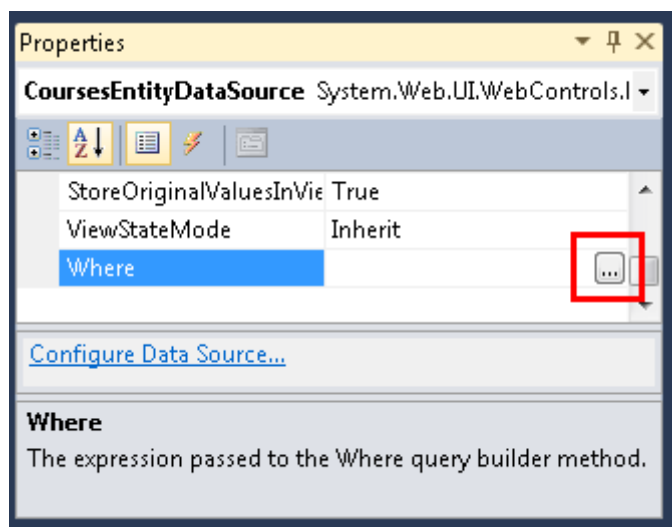


5. Откройте страницу в обозревателе. Проверьте реализованную функциональность.

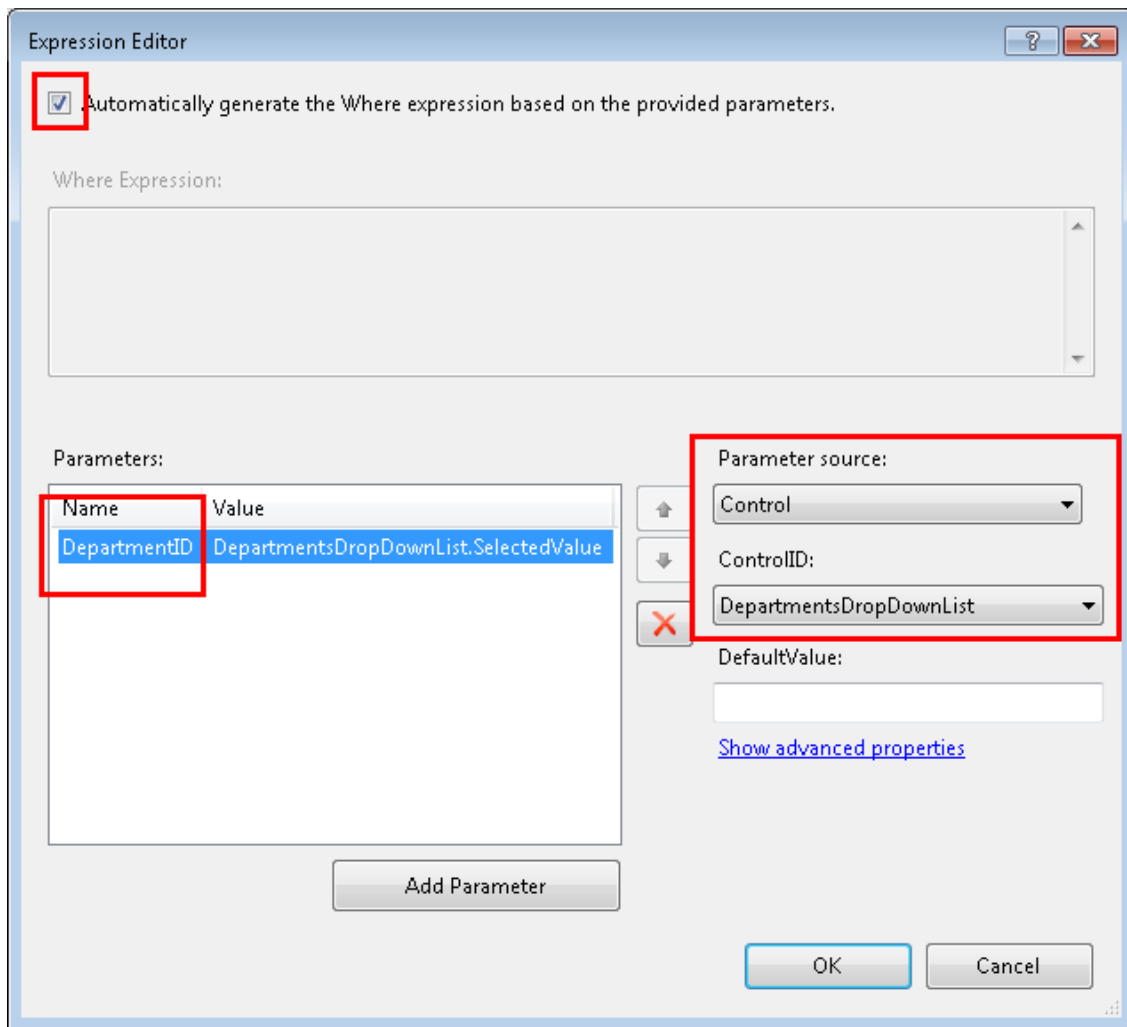
### Использование параметров элемента управления в свойстве "Where"

Можно передать значения параметров в свойство `Where`. На странице *Courses.aspx*, вы можете это использовать для отображения курсов, связанных с кафедрой, которую пользователь выбирает из выпадающего списка.

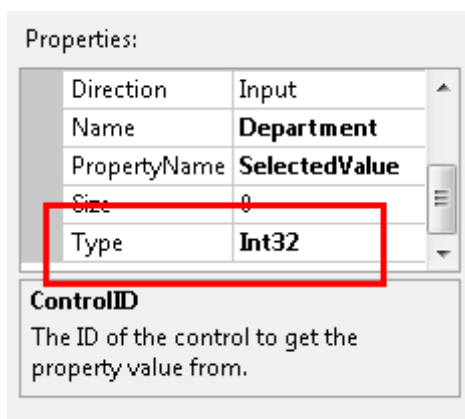
6. Откройте *Courses.aspx* в режиме конструктора.
7. Добавьте на страницу второй элемент EntityDataSource, установите свойству (**ID**) имя `CoursesEntityDataSource`. Настройте источник данных: установите соединение с моделью `SchoolEntities` и выберите в качестве набора **EntitySetName** значение `Courses`.
8. В окне свойств элемента `CoursesEntityDataSource` кликните на три точки напротив свойства **Where**.



9. В окне Редактор выражений (Expression Editor) выберите Автоматически создавать выражение Where на основе параметров (Automatically generate the Where ...) и кликните Добавить параметр (Add Parameter).
10. Укажите имя параметру – DepartmentID, в списке Источник параметров (Parameter source) выберите Control, в списке ControlID – DepartmentsDropDownList.



11. Кликните Показать дополнительные (Show advanced properties) и в окне свойств редактора выражений для свойства Type укажите Int32.



12. Нажмите ОК.

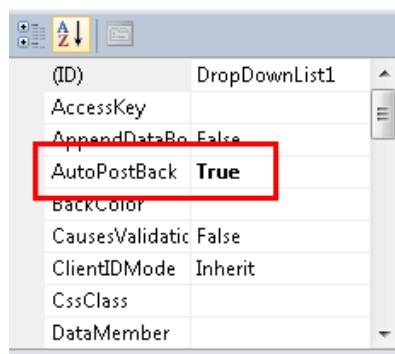
13. Ниже выпадающего списка добавьте элемента GridView и укажите (ID) как CoursesGridView.

14. Для нового элемента укажите источник данных – CoursesEntityDataSource, кликните **Обновить схему (Refresh Schema)**, далее кликните **Правка столбцов (Edit Columns)**, удалите столбец DepartmentID. Код разметки элемента GridView должен быть следующим.

```
<asp:GridView ID="CoursesGridView" runat="server" AutoGenerateColumns="False"
    DataKeyNames="CourseID" DataSourceID="CoursesEntityDataSource">
    <Columns>
        <asp:BoundField DataField="CourseID" HeaderText="CourseID"
ReadOnly="True"
            SortExpression="CourseID" />
        <asp:BoundField DataField="Title" HeaderText="Title"
SortExpression="Title" />
        <asp:BoundField DataField="Credits" HeaderText="Credits"
            SortExpression="Credits" />
    </Columns>
</asp:GridView>
```

Когда пользователь выберет название кафедры (department) в выпадающем списке, список курсов должен изменяться автоматически соответственно кафедре.

15. Для реализации этого выделите элемент выпадающий список и в окне свойств для свойства AutoPostBack установите значение True.



16. Перейдите в режим разметки кода и замените свойства ConnectionString и DefaultContainer элемента CoursesEntityDataSource на ContextTypeName="ContosoUniversity.DAL.SchoolEntities". Код разметки должен быть следующим.

```
<asp:EntityDataSource ID="CoursesEntityDataSource" runat="server"
    ContextTypeName="ContosoUniversity.DAL.SchoolEntities"
EnableFlattening="false"
    EntitySetName="Courses"
    AutoGenerateWhereClause="true" Where="">
    <WhereParameters>
        <asp:ControlParameter ControlID="DepartmentsDropDownList"
Type="Int32"
            Name="DepartmentID" PropertyName="SelectedValue" />
    </WhereParameters>
</asp:EntityDataSource>
```

17. Запустите страницу Courses.aspx в обозревателе. В выпадающем списке выберите любой элемент и проверьте изменение списка курсов в элементе GridView.

**COURSES BY DEPARTMENT**

Select a Department

ID	Title	Credits
2021	Composition 3	
2030	Poetry	2
2042	Literature	4

### Использование свойства "GroupBy" элемента EntityDataSource для группирования данных

18. Откройте страницу *About.aspx* в режиме разметки, замените существующее содержимое элемента `BodyContent` на "Student Body Statistics" между тегами `h2`, а между `<p>` удалите:

```
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
    <h2>
        Student Body Statistics
    </h2>
</asp:Content>
```

19. После заголовка `h2` добавьте элемент `EntityDataSource`, укажите ID, равным `StudentStatisticsEntityDataSource`.

20. В режиме конструктора настройте источник данных: соединение `SchoolEntities`, набор данных `EntitySetName` – **People**, остальные параметры оставьте по умолчанию.

21. Установите в окне свойств:

- для фильтрации только студентов свойству `Where` значение `it.EnrollmentDate is not null`.
- для группирования по дате зачисления свойству `GroupBy` установите значение `it.EnrollmentDate`.
- для отбора по дате и подсчета количества студентов свойству `Select` установите `it.EnrollmentDate, Count(it.EnrollmentDate) AS NumberOfStudents`.
- для определения порядка вывода результатов по дате свойству `OrderBy` установите `it.EnrollmentDate`.

Код разметки элемента `EntityDataSource` должен быть следующим.

```
<asp:EntityDataSource ID="StudentStatisticsEntityDataSource" runat="server"
    ConnectionString="name=SchoolEntities" DefaultContainerName="SchoolEntities"
    EnableFlattening="False" EntitySetName="People" GroupBy="it.EnrollmentDate"
    OrderBy="it.EnrollmentDate"
    Select="it.EnrollmentDate, Count(it.EnrollmentDate) AS NumberOfStudents"
    Where="it.EnrollmentDate is not null">
</asp:EntityDataSource>
```

22. Добавьте следующий код для создания элемента `GridView` для отображения данных:

```
<asp:GridView ID="StudentStatisticsGridView" runat="server"
    AutoGenerateColumns="False"
    DataSourceID="StudentStatisticsEntityDataSource">
    <Columns>
        <asp:BoundField DataField="EnrollmentDate" DataFormatString="{0:d}"
            HeaderText="Date of Enrollment"
            ReadOnly="True" SortExpression="EnrollmentDate" />
    </Columns>
</asp:GridView>
```

```

        <asp:BoundField DataField="NumberOfStudents" HeaderText="Students"
            ReadOnly="True" SortExpression="NumberOfStudents" />
    </Columns>
</asp:GridView>

```

23. Запустите страницу *About.aspx* в обозревателе и просмотрите результаты статистики по датам зачисления.

Date of Enrollment	Students
9/1/2000	2
9/1/2001	5
9/1/2002	3
1/30/2003	1
9/1/2003	3
9/1/2004	5
9/1/2005	6
1/1/2011	1

## Упражнение 2. Реализация поиска данных

### Применение элемента QueryExtender для фильтрации и упорядочивания

Элемент `QueryExtender` обеспечивает способ определения фильтрации и сортировки в разметке. В этой части упражнения вы будете использовать элемент `QueryExtender` для фильтрации и упорядочивания данных, используя навигационное свойство.

1. Откройте страницу *Courses.aspx* в коде разметки и ниже кода `GridView` вставьте следующий код для создания заголовка, текстового поля для ввода строки поиска и кнопку поиска, а также добавьте код настройки элемента `EntityDataSource`:

```

        <h2>Courses by Name</h2>
        Enter a course name
        <asp:TextBox ID="SearchTextBox" runat="server"></asp:TextBox>
        <asp:Button ID="SearchButton" runat="server" Text="Search" />
        <br /><br />
        <asp:EntityDataSource ID="SearchEntityDataSource" runat="server"
            ConnectionString="name=SchoolEntities"
            DefaultContainerName="SchoolEntities" EnableFlattening="False"
            EntitySetName="Courses"
            Include="Department" >
        </asp:EntityDataSource>

```

Обратите внимание, что свойство `Include` элемента управления `EntityDataSource` устанавливается как `Department`. В базе данных таблица `Course` не содержит название кафедры, она содержит столбец внешнего ключа `DepartmentID`. Если бы вы выполняли запросы к базе данных напрямую для получения названия кафедры вместе с данными курса, вы должны были соединить таблицы `Course` курс и `Department`.

Установив свойству `Include` значение `Department`, вы указали, что `Entity Framework` должен обеспечить получение соответствующего объекта `Department`, когда он получает объект `Course`. Объект `Department` затем хранится в навигационном свойстве объекта `Course`.

2. После элемента `EntityDataSource` вставьте следующий код разметки для создания элемента `QueryExtender`, связанного с `EntityDataSource`:

```
<asp:QueryExtender ID="SearchQueryExtender" runat="server"
    TargetControlID="SearchEntityDataSource" >
    <asp:SearchExpression SearchType="StartsWith" DataFields="Title">
        <asp:ControlParameter ControlID="SearchTextBox" />
    </asp:SearchExpression>
    <asp:OrderByExpression DataField="Department.Name" Direction="Ascending">
        <asp:ThenBy DataField="Title" Direction="Ascending" />
    </asp:OrderByExpression>
</asp:QueryExtender>
```

Элемент `SearchExpression` определяет выражение поиска курса. Свойство `SearchType` определяется как `StartsWith`. Это означает, что поиск должен определить, совпадает ли строка поиска с какой-либо строкой в начале поля.

Элемент `OrderByExpression` указывает, что набор результатов будет упорядочен по названию курса в названии кафедры. Обратите внимание, как указано название отдела: `Department.Name`. Поскольку связь между объектом `Course` и `Department` является один-к-одному, свойство навигации `Department` содержит сущность `Department`. Если бы это было отношения один-ко-многим, свойство содержало бы коллекцию. Чтобы получить названия кафедры, необходимо указать свойство `Name` сущности `Department`.

3. Добавьте элемент `GridView` для отображения результатов требуемого запроса:

```
<asp:GridView ID="SearchGridView" runat="server"
    AutoGenerateColumns="False"
    DataKeyNames="CourseID" DataSourceID="SearchEntityDataSource"
    AllowPaging="true">
    <Columns>
        <asp:TemplateField HeaderText="Department">
            <ItemTemplate>
                <asp:Label ID="Label2" runat="server" Text='<%#
Eval("Department.Name") %>'></asp:Label>
            </ItemTemplate>
        </asp:TemplateField>
        <asp:BoundField DataField="CourseID" HeaderText="ID"/>
        <asp:BoundField DataField="Title" HeaderText="Title" />
        <asp:BoundField DataField="Credits" HeaderText="Credits" />
    </Columns>
</asp:GridView>
```

Первым столбцом является шаблонное поле, которое отображает название кафедры. Выражение привязки данных определяет `Department.Name`, как вы видели в элементе `QueryExtender`.

4. Запустите страницу в обозревателе. Введите название курса и просмотрите результаты.



### COURSES BY DEPARTMENT

Select a Department

ID	Title	Credits
2021	Composition	3
2030	Poetry	2
2042	Literature	4

### COURSES BY NAME

Enter a course name

Department	ID	Title	Credits
Economics	4041	Macroeconomics	3
Economics	4022	Microeconomics	3
Economics	4063	new course	5

5. Введите букву "m" и кликните кнопку **Search** для просмотра всех курсов начинающихся с этой буквы.

### COURSES BY DEPARTMENT

Select a Department

ID	Title	Credits
2021	Composition	3
2030	Poetry	2
2042	Literature	4

### COURSES BY NAME

Enter a course name

Department	ID	Title	Credits
Economics	4041	Macroeconomics	3
Economics	4022	Microeconomics	3

### Применение оператора "Like" Operator to Filter Data

Вы можете достичь эффекта, аналогичного тому, который предоставляется элементом QueryExtender со свойствами `StartsWith`, `Contains` и `EndsWith`, путем использования оператора `Like` в свойстве `Where` элемента управления `EntityDataSource`. В этой части упражнения вы увидите, как использовать оператор `Like` для поиска студента по имени.

6. Откройте файл *Students.aspx* в режиме разметки кода. После элемента `GridView` добавьте следующую разметку:

```
<h2>Find Students by Name</h2>
Enter any part of the name
<asp:TextBox ID="SearchTextBox" runat="server"
AutoPostBack="true"></asp:TextBox>
<asp:Button ID="SearchButton" runat="server" Text="Search" />
<br />
<br />
```

```

        <asp:EntityDataSource ID="SearchEntityDataSource" runat="server"
            ConnectionString="name=SchoolEntities"
            DefaultContainerName="SchoolEntities" EnableFlattening="False"
            EntitySetName="People"
            Where="it.EnrollmentDate is not null and (it.FirstMidName Like '%' +
@StudentName + '%' or it.LastName Like '%' + @StudentName + '%') " >
            <WhereParameters>
                <asp:ControlParameter ControlID="SearchTextBox" Name="StudentName"
PropertyName="Text"
                Type="String" DefaultValue="%" />
            </WhereParameters>
        </asp:EntityDataSource>
        <asp:GridView ID="SearchGridView" runat="server" AutoGenerateColumns="False"
DataKeyNames="PersonID"
            DataSourceID="SearchEntityDataSource" AllowPaging="true">
            <Columns>
                <asp:TemplateField HeaderText="Name" SortExpression="LastName,
FirstMidName">
                    <ItemTemplate>
                        <asp:Label ID="LastNameFoundLabel" runat="server" Text='<#
Eval("LastName") %>'></asp:Label>,
                        <asp:Label ID="FirstNameFoundLabel" runat="server" Text='<#
Eval("FirstMidName") %>'></asp:Label>
                    </ItemTemplate>
                </asp:TemplateField>
                <asp:TemplateField HeaderText="Enrollment Date"
SortExpression="EnrollmentDate">
                    <ItemTemplate>
                        <asp:Label ID="EnrollmentDateFoundLabel" runat="server"
Text='<# Eval("EnrollmentDate", "{0:d}") %>'></asp:Label>
                    </ItemTemplate>
                </asp:TemplateField>
            </Columns>
        </asp:GridView>

```

7. Запустите страницу в обозревателе. Initially you see all of the students because the default value for the StudentName parameter is "%".

**FIND STUDENTS BY NAME**

Enter any part of the name

Name	EnrollmentDate
Barzdukas, Gytis	9/1/2005
Justice, Peggy	9/1/2001
Li, Yan	9/1/2002

8. Введите букву "g" в текстовое поле и кликните **Search**. Вы увидите список студентов, которые имеют "g" в имени или фамилии.

**FIND STUDENTS BY NAME**

Enter any part of the name

Name	EnrollmentDate
Barzdukas, Gytis	9/1/2005
Justice, Peggy	9/1/2001
Tang, Wayne	9/1/2005

## Lab 4. Работа со связанными данными

В прошлой работе Вы настроили элемент `EntityDataSource` для фильтрации, сортировки и группирования данных. В этой работе Вы создадите страницу *Инструкторы*, которая отобразит список преподавателей (инструкторов). При выборе инструктора, вы увидите список курсов, преподаваемых этим инструктором. При выборе курса вы увидите подробности курса и список студентов, обучающихся на курсе.

### INSTRUCTORS

	Name	Hire Date	Office Assignment
<a href="#">Edit</a> <a href="#">Select</a>	Abercrombie, Kim	3/11/1995	Smith 18
<a href="#">Edit</a> <a href="#">Select</a>	Fakhouri, Fadi	8/6/2002	29 Adams
<a href="#">Edit</a> <a href="#">Select</a>	Harui, Roger	7/1/1998	37 Williams
<a href="#">Edit</a> <a href="#">Select</a>	Zheng, Roger	2/12/2004	143 Smith
<a href="#">Edit</a> <a href="#">Select</a>	Kapoor, Candace	1/15/2001	57 Adams
<a href="#">Edit</a> <a href="#">Select</a>	Serrano, Stacy	6/1/1999	271 Williams
<a href="#">Edit</a> <a href="#">Select</a>	Stewart, Jasmine	10/12/1997	131 Smith
<a href="#">Edit</a> <a href="#">Select</a>	Xu, Kristen	7/23/2001	203 Williams
<a href="#">Edit</a> <a href="#">Select</a>	Van Houten, Roger	12/7/2000	213 Smith

### COURSE DETAILS

ID	2030
Title	Poetry
Credits	2
Department	English
Location	
URL	http://www.fineartschool.n

### COURSES TAUGHT

	ID	Title	Department
<a href="#">Select</a>	2030	Poetry	English

### STUDENT GRADES

Name	Grade
Barzdukas, Gytis	3.50
Justice, Peggy	4.00

### Упражнение 1. Отображение и обновление связанных данных в элементе GridView

1. Создайте новую страницу *Instructors.aspx*, основанная на странице *Site.Master* и добавьте следующую разметку в элемент Content с именем Content2:

```
<h2>Instructors</h2>
<div>
  <asp:EntityDataSource ID="InstructorsEntityDataSource" runat="server"
    ConnectionString="name=SchoolEntities" DefaultContainerName="SchoolEntities"
    EnableFlattening="False"
    EntitySetName="People"
    Where="it.HireDate is not null" Include="OfficeAssignment"
    EnableUpdate="True">
  </asp:EntityDataSource>
</div>
```

Эта разметка создает элемент `EntityDataSource`, который выбирает инструкторов и позволяет выполнять обновления.

2. Между разметкой `EntityDataSource` и закрывающим тегом `</div>` добавьте код разметки для создания элементов `GridView` и `Label` (этот элемент будет отображать сообщения об ошибках):

```
<asp:GridView ID="InstructorsGridView" runat="server"
  AllowPaging="True" AllowSorting="True"
  AutoGenerateColumns="False" DataKeyNames="PersonID"
  DataSourceID="InstructorsEntityDataSource"
  OnSelectedIndexChanged="InstructorsGridView_SelectedIndexChanged"
  SelectedRowStyle-BackColor="LightGray"
  onrowupdating="InstructorsGridView_RowUpdating">
```

```

        <Columns>
            <asp:CommandField ShowSelectButton="True" ShowEditButton="True" />
            <asp:TemplateField HeaderText="Name" SortExpression="LastName">
                <ItemTemplate>
                    <asp:Label ID="InstructorLastNameLabel" runat="server"
Text='<%# Eval("LastName") %>'></asp:Label>,
                    <asp:Label ID="InstructorFirstNameLabel" runat="server"
Text='<%# Eval("FirstMidName") %>'></asp:Label>
                </ItemTemplate>
                <EditItemTemplate>
                    <asp:TextBox ID="InstructorLastNameTextBox" runat="server"
Text='<%# Bind("FirstMidName") %>' Width="7em"></asp:TextBox>
                    <asp:TextBox ID="InstructorFirstNameTextBox" runat="server"
Text='<%# Bind("LastName") %>' Width="7em"></asp:TextBox>
                </EditItemTemplate>
            </asp:TemplateField>
            <asp:TemplateField HeaderText="Hire Date"
SortExpression="HireDate">
                <ItemTemplate>
                    <asp:Label ID="InstructorHireDateLabel" runat="server"
Text='<%# Eval("HireDate", "{0:d}") %>'></asp:Label>
                </ItemTemplate>
                <EditItemTemplate>
                    <asp:TextBox ID="InstructorHireDateTextBox" runat="server"
Text='<%# Bind("HireDate", "{0:d}") %>' Width="7em"></asp:TextBox>
                </EditItemTemplate>
            </asp:TemplateField>
            <asp:TemplateField HeaderText="Office Assignment"
SortExpression="OfficeAssignment.Location">
                <ItemTemplate>
                    <asp:Label ID="InstructorOfficeLabel" runat="server"
Text='<%# Eval("OfficeAssignment.Location") %>'></asp:Label>
                </ItemTemplate>
                <EditItemTemplate>
                    <asp:TextBox ID="InstructorOfficeTextBox" runat="server"
Text='<%# Eval("OfficeAssignment.Location") %>' Width="7em"
oninit="InstructorOfficeTextBox_Init"></asp:TextBox>
                </EditItemTemplate>
            </asp:TemplateField>
        </Columns>
        <SelectedRowStyle BackColor="LightGray"></SelectedRowStyle>
    </asp:GridView>
    <asp:Label ID="ErrorMessageLabel" runat="server" Text="" Visible="false"
ViewStateMode="Disabled"></asp:Label>

```

Элемент GridView позволяет выбрать строку, подчеркивает выбранную строку с легким серым фоном, и определяет обработчики событий (которые вы создадите позже) SelectedIndexChanged и Updating. Он также указывает PersonId для свойства DataKeyNames, так что значение ключа выбранной строки могут быть переданы другому элементу управления, что вы будете добавлять позже.

В последнем столбце содержится значение офиса инструктора, который хранится в навигационном свойстве сущности Person, потому что он приходит от соответствующего экземпляра. Обратите внимание, что элемент EditItemTemplate определяет Eval вместо Bind, потому что GridView управления не может непосредственно связываться с навигационными свойствами для того, чтобы обновить их. Вы обновить назначение офис в коде. Чтобы сделать это, вам нужно ссылка элемент TextBox, и вы получите и сохраните, что в случае события Init элемента управления TextBox.

После элемента GridView добавлен элемент Label, который используется для сообщений об ошибках. Свойство Visible изначально установлено в false и состояние просмотра отключено, так что сообщение появится только тогда, когда произойдет ошибка.

3. Откройте файл *Instructors.aspx.cs* и добавьте выражение using:

```
using ContosoUniversity.DAL;
```

4. Добавьте ссылку на текстовое поле непосредственно после объявления класса:

```
private TextBox instructorOfficeTextBox;
```

5. Добавьте заглушку обработчика события SelectedIndexChanged (код будет добавлен позже).

```
protected void InstructorsGridView_SelectedIndexChanged(object sender, EventArgs e)
{
}
```

6. Добавьте обработчик события Init элемента управления InstructorOfficeTextBox, в котором сохраните ссылку на элемент TextBox. Вы будете использовать эту ссылку, чтобы получить значение, введенное пользователем при обновлении.

```
protected void InstructorOfficeTextBox_Init(object sender, EventArgs e)
{
    instructorOfficeTextBox = sender as TextBox;
}
```

Вы будете использовать событие Updating элемента GridView, чтобы обновить свойство Location, связанное с сущностью OfficeAssignment.

7. Добавьте следующий обработчик для события Updating:

```
protected void InstructorsGridView_RowUpdating(object sender, GridViewUpdateEventArgs e)
{
    using (var context = new SchoolEntities())
    {
        var instructorBeingUpdated = Convert.ToInt32(e.Keys[0]);
        var officeAssignment = (from o in context.OfficeAssignments
                                where o.InstructorID == instructorBeingUpdated
                                select o).FirstOrDefault();

        try
        {
            if (String.IsNullOrEmpty(instructorOfficeTextBox.Text) == false)
            {
                if (officeAssignment == null)
                {
                    context.OfficeAssignments.AddObject(OfficeAssignment.CreateOfficeAssignment(instructorBeingUpdated, instructorOfficeTextBox.Text, null));
                }
                else
                {
                    officeAssignment.Location = instructorOfficeTextBox.Text;
                }
            }
        }
    }
}
```

```

    }
    else
    {
        if (officeAssignment != null)
        {
            context.DeleteObject(officeAssignment);
        }
    }
    context.SaveChanges();
}
catch (Exception)
{
    e.Cancel = true;
    ErrorMessageLabel.Visible = true;
    ErrorMessageLabel.Text = "Update failed.";
    //Add code to log the error.
}
}
}

```

Этот код запускается, когда пользователь нажимает **Update** в строке элемента GridView. Код использует LINQ to Entities, чтобы получить сущность OfficeAssignment, которая связана с текущей сущностью Person, используя PersonId выбранной строки из аргумента события.

8. Запустите страницу *Instructors.aspx* в обозревателе.

#### INSTRUCTORS

	Name	Hire Date	Office Assignment
<a href="#">Edit</a> <a href="#">Select</a>	Abercrombie, Kim	3/11/1995	Smith 17
<a href="#">Edit</a> <a href="#">Select</a>	Fakhouri, Fadi	8/6/2002	29 Adams
<a href="#">Edit</a> <a href="#">Select</a>	Harui, Roger	7/1/1998	37 Williams
<a href="#">Edit</a> <a href="#">Select</a>	Zheng, Roger	2/12/2004	143 Smith
<a href="#">Edit</a> <a href="#">Select</a>	Kapoor, Candace	1/15/2001	57 Adams
<a href="#">Edit</a> <a href="#">Select</a>	Serrano, Stacy	6/1/1999	271 Williams
<a href="#">Edit</a> <a href="#">Select</a>	Stewart, Jasmine	10/12/1997	131 Smith
<a href="#">Edit</a> <a href="#">Select</a>	Xu, Kristen	7/23/2001	203 Williams
<a href="#">Edit</a> <a href="#">Select</a>	Van Houten, Roger	12/7/2000	213 Smith

9. Кликните **Edit (Правка)**, отобразятся текстовые поля.

#### INSTRUCTORS

	Name		Hire Date	Office Assignment
<a href="#">Update</a> <a href="#">Cancel</a>	Kim	Abercrombie	3/11/1995	Smith 17
<a href="#">Edit</a> <a href="#">Select</a>	Fakhouri, Fadi		8/6/2002	29 Adams
<a href="#">Edit</a> <a href="#">Select</a>	Harui, Roger		7/1/1998	37 Williams
<a href="#">Edit</a> <a href="#">Select</a>	Zheng, Roger		2/12/2004	143 Smith
<a href="#">Edit</a> <a href="#">Select</a>	Kapoor, Candace		1/15/2001	57 Adams
<a href="#">Edit</a> <a href="#">Select</a>	Serrano, Stacy		6/1/1999	271 Williams
<a href="#">Edit</a> <a href="#">Select</a>	Stewart, Jasmine		10/12/1997	131 Smith
<a href="#">Edit</a> <a href="#">Select</a>	Xu, Kristen		7/23/2001	203 Williams
<a href="#">Edit</a> <a href="#">Select</a>	Van Houten, Roger		12/7/2000	213 Smith

10. Измените все значения, в том числе и поле **Office Assignment**. Кликните **Update (Обновить)** и вы увидите изменения, отраженные в списке.

## **Упражнение 2. Отображение связанных данных в отдельном элементе управления**

Каждый преподаватель может быть назначен на один или на несколько курсов. В этом упражнении Вы добавите элементы `EntityDataSource` и `GridView` для перечисления курсов, связанных с инструктором, выбранном в элементе `GridView`, отображающем инструкторов.

1. Для создания заголовка (подписи) и элемента `EntityDataSource` для курсов добавьте следующий код разметки между элементом `Label` (сообщающий о возможных ошибках) и закрывающим тегом `</div>`:

```
<h3>Courses Taught</h3>
<asp:EntityDataSource ID="CoursesEntityDataSource" runat="server"
    ConnectionString="name=SchoolEntities"
    DefaultContainerName="SchoolEntities" EnableFlattening="False"
    EntitySetName="Courses"
    Where="@PersonID IN (SELECT VALUE instructor.PersonID FROM it.People AS
instructor)">
    <WhereParameters>
        <asp:ControlParameter ControlID="InstructorsGridView" Type="Int32"
Name="PersonID" PropertyName="SelectedValue" />
    </WhereParameters>
</asp:EntityDataSource>
```

Параметр `Where` содержит значение `PersonID` инструктора выбираемой строки в элементе `InstructorsGridView`.

2. Для создания элемента `GridView` добавьте следующий код разметки сразу после элемента `CoursesEntityDataSource` (до тега `</div>`):

```
<asp:GridView ID="CoursesGridView" runat="server"
    DataSourceID="CoursesEntityDataSource"
    AllowSorting="True" AutoGenerateColumns="False"
    SelectedRowStyle-BackColor="LightGray"
    DataKeyNames="CourseID">
    <EmptyDataTemplate>
        <p>No courses found.</p>
    </EmptyDataTemplate>
    <Columns>
        <asp:CommandField ShowSelectButton="True" />
        <asp:BoundField DataField="CourseID" HeaderText="ID"
ReadOnly="True" SortExpression="CourseID" />
        <asp:BoundField DataField="Title" HeaderText="Title"
SortExpression="Title" />
        <asp:TemplateField HeaderText="Department"
SortExpression="DepartmentID">
            <ItemTemplate>
                <asp:Label ID="GridViewDepartmentLabel" runat="server"
Text='<%# Eval("Department.Name") %>'></asp:Label>
            </ItemTemplate>
        </asp:TemplateField>
    </Columns>
</asp:GridView>
```

3. Откройте страницу в обозревателе.

## INSTRUCTORS

	<a href="#">Name</a>	<a href="#">Hire Date</a>	<a href="#">Office Assignment</a>
<a href="#">Edit</a> <a href="#">Select</a>	Abercrombie, Kim	3/11/1995	Smith 17
<a href="#">Edit</a> <a href="#">Select</a>	Fakhouri, Fadi	8/6/2002	29 Adams
<a href="#">Edit</a> <a href="#">Select</a>	Harui, Roger	7/1/1998	37 Williams
<a href="#">Edit</a> <a href="#">Select</a>	Zheng, Roger	2/12/2004	143 Smith
<a href="#">Edit</a> <a href="#">Select</a>	Kapoor, Candace	1/15/2001	57 Adams
<a href="#">Edit</a> <a href="#">Select</a>	Serrano, Stacy	6/1/1999	271 Williams
<a href="#">Edit</a> <a href="#">Select</a>	Stewart, Jasmine	10/12/1997	131 Smith
<a href="#">Edit</a> <a href="#">Select</a>	Xu, Kristen	7/23/2001	203 Williams
<a href="#">Edit</a> <a href="#">Select</a>	Van Houten, Roger	12/7/2000	213 Smith

## COURSES TAUGHT

No courses found.

4. Выделите любую строку с инструктором и проверьте, что внизу отображается информация о читаемом им курсе.

## INSTRUCTORS

	<a href="#">Name</a>	<a href="#">Hire Date</a>	<a href="#">Office Assignment</a>
<a href="#">Edit</a> <a href="#">Select</a>	Abercrombie, Kim	3/11/1995	Smith 17
<a href="#">Edit</a> <a href="#">Select</a>	Fakhouri, Fadi	8/6/2002	29 Adams
<a href="#">Edit</a> <a href="#">Select</a>	Harui, Roger	7/1/1998	37 Williams
<a href="#">Edit</a> <a href="#">Select</a>	Zheng, Roger	2/12/2004	143 Smith
<a href="#">Edit</a> <a href="#">Select</a>	Kapoor, Candace	1/15/2001	57 Adams
<a href="#">Edit</a> <a href="#">Select</a>	Serrano, Stacy	6/1/1999	271 Williams
<a href="#">Edit</a> <a href="#">Select</a>	Stewart, Jasmine	10/12/1997	131 Smith
<a href="#">Edit</a> <a href="#">Select</a>	Xu, Kristen	7/23/2001	203 Williams
<a href="#">Edit</a> <a href="#">Select</a>	Van Houten, Roger	12/7/2000	213 Smith

## COURSES TAUGHT

	<a href="#">ID</a>	<a href="#">Title</a>	<a href="#">Department</a>
<a href="#">Select</a>	2030	Poetry	English

Элемент `CoursesGridView` показывает только несколько полей о курсе. Для отображения всей детальной информации о курсе можно использовать элемент `DetailsView` для курса, который выбрал пользователь.

5. Добавьте следующий код разметки после закрывающего тега `</div>`:

```
<div>
<h3>Course Details</h3>
<asp:EntityDataSource ID="CourseDetailsEntityDataSource" runat="server"
    ConnectionString="name=SchoolEntities"
    DefaultContainerName="SchoolEntities"
    EnableFlattening="False"
    EntitySetName="Courses"
    AutoGenerateWhereClause="False" Where="it.CourseID = @CourseID"
    Include="Department,OnlineCourse,OnsiteCourse,StudentGrades.Person"
    OnSelected="CourseDetailsEntityDataSource_Selected">
    <WhereParameters>
```



```

        <asp:ControlParameter ControlID="CoursesGridView" Type="Int32"
Name="CourseID" PropertyName="SelectedValue" />
    </WhereParameters>
</asp:EntityDataSource>
<asp:DetailsView ID="CourseDetailsView" runat="server"
AutoGenerateRows="False"
DataSourceID="CourseDetailsEntityDataSource">
    <EmptyDataTemplate>
        <p>
            No course selected.</p>
    </EmptyDataTemplate>
    <Fields>
        <asp:BoundField DataField="CourseID" HeaderText="ID"
ReadOnly="True" SortExpression="CourseID" />
        <asp:BoundField DataField="Title" HeaderText="Title"
SortExpression="Title" />
        <asp:BoundField DataField="Credits" HeaderText="Credits"
SortExpression="Credits" />
        <asp:TemplateField HeaderText="Department">
            <ItemTemplate>
                <asp:Label ID="DetailsViewDepartmentLabel" runat="server"
Text='<%# Eval("Department.Name") %>'></asp:Label>
            </ItemTemplate>
        </asp:TemplateField>
        <asp:TemplateField HeaderText="Location">
            <ItemTemplate>
                <asp:Label ID="LocationLabel" runat="server" Text='<%#
Eval("OnsiteCourse.Location") %>'></asp:Label>
            </ItemTemplate>
        </asp:TemplateField>
        <asp:TemplateField HeaderText="URL">
            <ItemTemplate>
                <asp:Label ID="URLLabel" runat="server" Text='<%#
Eval("OnlineCourse.URL") %>'></asp:Label>
            </ItemTemplate>
        </asp:TemplateField>
    </Fields>
</asp:DetailsView>
</div>

```

Этот код создает элемент EntityDataSource, который привязан к набору Courses. Свойство Where выбирает курс, используя значение CourseID выбранной строки в элементе GridView, отображающий курсы.

Разметка также определяет обработчик для выбранного события, которое вы будете использовать в дальнейшем для отображения оценки студентов.

6. В файле *Instructors.aspx.cs* создайте заглушеу для метода CourseDetailsEntityDataSource\_Selected

```

protected void CourseDetailsEntityDataSource_Selected(object
sender, EventArgs e)
{
}

```

7. Откройте страницу в обозревателе.

## INSTRUCTORS

	Name	Hire Date	Office Assignment
<a href="#">Edit</a> <a href="#">Select</a>	Abercrombie, Kim	3/11/1995	Smith 17
<a href="#">Edit</a> <a href="#">Select</a>	Fakhouri, Fadi	8/6/2002	29 Adams
<a href="#">Edit</a> <a href="#">Select</a>	Harui, Roger	7/1/1998	37 Williams
<a href="#">Edit</a> <a href="#">Select</a>	Zheng, Roger	2/12/2004	143 Smith
<a href="#">Edit</a> <a href="#">Select</a>	Kapoor, Candace	1/15/2001	57 Adams
<a href="#">Edit</a> <a href="#">Select</a>	Serrano, Stacy	6/1/1999	271 Williams
<a href="#">Edit</a> <a href="#">Select</a>	Stewart, Jasmine	10/12/1997	131 Smith
<a href="#">Edit</a> <a href="#">Select</a>	Xu, Kristen	7/23/2001	203 Williams
<a href="#">Edit</a> <a href="#">Select</a>	Van Houten, Roger	12/7/2000	213 Smith

## COURSE DETAILS

No course selected.

## COURSES TAUGHT

No courses found.

- Выберите инструктора и затем выберите курс для просмотра подробной о нем информации.

## INSTRUCTORS

	Name	Hire Date	Office Assignment
<a href="#">Edit</a> <a href="#">Select</a>	Abercrombie, Kim	3/11/1995	17 Smith
<a href="#">Edit</a> <a href="#">Select</a>	Fakhouri, Fadi	8/6/2002	29 Adams
<a href="#">Edit</a> <a href="#">Select</a>	Harui, Roger	7/1/1998	37 Williams
<a href="#">Edit</a> <a href="#">Select</a>	Zheng, Roger	2/12/2004	143 Smith
<a href="#">Edit</a> <a href="#">Select</a>	Kapoor, Candace	1/15/2001	57 Adams
<a href="#">Edit</a> <a href="#">Select</a>	Serrano, Stacy	6/1/1999	271 Williams
<a href="#">Edit</a> <a href="#">Select</a>	Stewart, Jasmine	10/12/1997	131 Smith
<a href="#">Edit</a> <a href="#">Select</a>	Xu, Kristen	7/23/2001	203 Williams
<a href="#">Edit</a> <a href="#">Select</a>	Van Houten, Roger	12/7/2000	213 Smith

## COURSE DETAILS

ID	2030
Title	Poetry
Credits	2
Department	English
Location	
URL	http://www.finearts

## COURSES TAUGHT

	ID	Title	Department
<a href="#">Select</a>	2030	Poetry	English

### Упражнение 3. Применение события "Selected" EntityDataSource для отображения связанных данных

В этом упражнении Вы отобразите всех зачисленных студентов для определенного курса. Чтобы сделать это, вы будете использовать событие Selected элемента управления EntityDataSource связанного с курсом в DetailsView.

- В файл *Instructors.aspx* добавьте код разметки после элемента DetailsView:

```
<h3>Student Grades</h3>
<asp:ListView ID="GradesListView" runat="server">
  <EmptyDataTemplate>
    <p>No student grades found.</p>
  </EmptyDataTemplate>
</LayoutTemplate>
```

```

        <table border="1" runat="server" id="itemPlaceholderContainer">
            <tr id="Tr1" runat="server">
                <th id="Th1" runat="server">
                    Name
                </th>
                <th id="Th2" runat="server">
                    Grade
                </th>
            </tr>
            <tr id="itemPlaceholder" runat="server">
            </tr>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td>
                <asp:Label ID="StudentLastNameLabel" runat="server"
Text='<# Eval("Person.LastName") %>' />,
                <asp:Label ID="StudentFirstNameLabel" runat="server"
Text='<# Eval("Person.FirstName") %>' />
            </td>
            <td>
                <asp:Label ID="StudentGradeLabel" runat="server" Text='<#
Eval("Grade") %>' />
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>

```

Этот код разметки создает элемент `ListView`, который отображает список студентов и их оценки для выбранного курса. Источник данных не задан, потому что вы выполните привязку в коде. Элемент `EmptyDataTemplate` обеспечивает сообщение для отображения, когда ничего не выбрано. Элемент `LayoutTemplate` создает HTML таблицу для отображения списка, а `ItemTemplate` определяет столбцы для отображения. ID студента и оценка студента определяются в сущности `StudentGrade`, имя студента выбирается из сущности `Person`, которую `Entity Framework` делает доступным в свойстве навигации `Person` сущности `StudentGrade`.

2. В файле *Instructors.aspx.cs* реализуйте метод `CourseDetailsEntityDataSource_Selected` следующим образом:

```

protected void CourseDetailsEntityDataSource_Selected(object
sender, EventArgs e)
{
    var course = e.Results.Cast<Course>().FirstOrDefault();
    if (course != null)
    {
        var studentGrades = course.StudentGrades.ToList();
        GradesListView.DataSource = studentGrades;
        GradesListView.DataBind();
    }
}

```

Аргумент этого события обеспечивает выбранные данные в виде коллекции, которая будет иметь нулевые элементы, если ничего не выбрано, или один пункт, если выбран объект `Course`. Если выбран объект `Course`, код использует первый метод, чтобы преобразовать коллекцию к одному объекту. Затем она получает `StudentGrade` объекты из

свойства навигации, преобразует их в коллекции, и связывает элемент `GradesListView` в коллекцию.

3. Для очистки отображения в классе `Instructors` создайте следующий метод:

```
private void ClearStudentGradesDataSource()  
{  
    var emptyStudentGradesList = new List<StudentGrade>();  
    GradesListView.DataSource = emptyStudentGradesList;  
    GradesListView.DataBind();  
}
```

4. Вызовите этот метод в обработчике события `Page_Load` страницы *Instructors.aspx* для отображения пустого шаблона при первой ее загрузке.

```
protected void Page_Load(object sender, EventArgs e)  
{  
    if (!IsPostBack)  
    {  
        ClearStudentGradesDataSource();  
    }  
}
```

5. Также вызовите этот метод в обработчике события `InstructorsGridView_SelectedIndexChanged` – это событие происходит при выборе инструктора:

```
protected void InstructorsGridView_SelectedIndexChanged(object  
sender, EventArgs e)  
{  
    ClearStudentGradesDataSource();  
}
```

6. Откройте страницу в обозревателе.

## INSTRUCTORS

	Name	Hire Date	Office Assignment
<a href="#">Edit</a> <a href="#">Select</a>	Abercrombie, Kim	3/11/1995	Smith 17
<a href="#">Edit</a> <a href="#">Select</a>	Fakhouri, Fadi	8/6/2002	29 Adams
<a href="#">Edit</a> <a href="#">Select</a>	Harui, Roger	7/1/1998	37 Williams
<a href="#">Edit</a> <a href="#">Select</a>	Zheng, Roger	2/12/2004	143 Smith
<a href="#">Edit</a> <a href="#">Select</a>	Kapoor, Candace	1/15/2001	57 Adams
<a href="#">Edit</a> <a href="#">Select</a>	Serrano, Stacy	6/1/1999	271 Williams
<a href="#">Edit</a> <a href="#">Select</a>	Stewart, Jasmine	10/12/1997	131 Smith
<a href="#">Edit</a> <a href="#">Select</a>	Xu, Kristen	7/23/2001	203 Williams
<a href="#">Edit</a> <a href="#">Select</a>	Van Houten, Roger	12/7/2000	213 Smith

## COURSES TAUGHT

No courses found.

## COURSE DETAILS

No course selected.

## STUDENT GRADES

No student grades found.

7. Выберите инструктора, который назначен на курс, а затем выберите курс.

## INSTRUCTORS

	Name	Hire Date	Office Assignment
<a href="#">Edit</a> <a href="#">Select</a>	Abercrombie, Kim	3/11/1995	17 Smith
<a href="#">Edit</a> <a href="#">Select</a>	Fakhouri, Fadi	8/6/2002	29 Adams
<a href="#">Edit</a> <a href="#">Select</a>	Harui, Roger	7/1/1998	37 Williams
<a href="#">Edit</a> <a href="#">Select</a>	Zheng, Roger	2/12/2004	143 Smith
<a href="#">Edit</a> <a href="#">Select</a>	Kapoor, Candace	1/15/2001	57 Adams
<a href="#">Edit</a> <a href="#">Select</a>	Serrano, Stacy	6/1/1999	271 Williams
<a href="#">Edit</a> <a href="#">Select</a>	Stewart, Jasmine	10/12/1997	131 Smith
<a href="#">Edit</a> <a href="#">Select</a>	Xu, Kristen	7/23/2001	203 Williams
<a href="#">Edit</a> <a href="#">Select</a>	Van Houten, Roger	12/7/2000	213 Smith

## COURSES TAUGHT

	ID	Title	Department
<a href="#">Select</a>	2030	Poetry	English

## COURSE DETAILS

ID	2030
Title	Poetry
Credits	2
Department	English
Location	
URL	http://www.finearts.

## STUDENT GRADES

Name	Grade
Barzdukas, Gytis	3.50
Justice, Peggy	4.00

## Lab 5. Работа со связанными данными (продолжение)

В предыдущей работе вы начали использовать элемент `EntityDataSource` для работы со связанными данными. В этой работе вы продолжите работать со связанными данными путем добавления и удаления отношения и добавления нового объекта, который имеет отношение к существующему объекту.

Вы создадите страницу, которая добавляет курсы, распределенные по кафедрам. Кафедры уже существуют, и, когда вы создаете новый курс, в то же время вы будете устанавливать отношения между ним и существующей кафедрой.

## ADD COURSES

ID	<input type="text"/>
Title	<input type="text"/>
Credits	<input type="text"/>
Department	Engineering <input type="button" value="v"/>
<a href="#">Insert</a> <a href="#">Cancel</a>	

Вы также создадите страницу, которая работает с отношением многие-ко-многим, назначая инструктора курса (добавление отношения между двумя сущностями, которые вы выберете) или удаление инструктора из курса (удаление отношения между двумя сущностями). В базе данных, добавив, отношения между преподавателем и курсом в новой строке добавляется к таблице ассоциации `CourseInstructor`; удаление отношения включает в себя удаление строки из таблицы ассоциации `CourseInstructor`. В `Entity Framework`, настроив свойства навигации, это делается без явного обращения к таблице `CourseInstructor`.

## ASSIGN INSTRUCTORS TO COURSES OR REMOVE FROM COURSES

Select an Instructor:

### ASSIGN A COURSE

Select a Course:

### REMOVE A COURSE

Select a Course:

### Упражнение 1. Добавление сущности с отношением к другой сущности

1. Создайте новую страницу *CoursesAdd.aspx*, которая основана на мастер странице *Site.Master* и добавьте код разметки элемента Content с именем Content2:

```
<h2>Add Courses</h2>
<asp:EntityDataSource ID="CoursesEntityDataSource" runat="server"
    ConnectionString="name=SchoolEntities"
    DefaultContainerName="SchoolEntities"
    EnableFlattening="False"
    EntitySetName="Courses"
    EnableInsert="True" EnableDelete="True" >
</asp:EntityDataSource>
<asp:DetailsView ID="CoursesDetailsView" runat="server"
    AutoGenerateRows="False"
    DataSourceID="CoursesEntityDataSource" DataKeyNames="CourseID"
    DefaultMode="Insert" oniteminserting="CoursesDetailsView_ItemInserting">
    <Fields>
        <asp:BoundField DataField="CourseID" HeaderText="ID" />
        <asp:BoundField DataField="Title" HeaderText="Title" />
        <asp:BoundField DataField="Credits" HeaderText="Credits" />
        <asp:TemplateField HeaderText="Department">
            <InsertItemTemplate>
                <asp:EntityDataSource ID="DepartmentsEntityDataSource"
                    runat="server" ConnectionString="name=SchoolEntities"
                    DefaultContainerName="SchoolEntities" EnableDelete="True"
                    EnableFlattening="False"
                    EntitySetName="Departments" EntityTypeFilter="Department">
                </asp:EntityDataSource>
                <asp:DropDownList ID="DepartmentsDropDownList" runat="server"
                    DataSourceID="DepartmentsEntityDataSource"
                    DataTextField="Name" DataValueField="DepartmentID"
                    oninit="DepartmentsDropDownList_Init">
                </asp:DropDownList>
            </InsertItemTemplate>
        </asp:TemplateField>
        <asp:CommandField ShowInsertButton="True" />
    </Fields>
</asp:DetailsView>
```

Этот код разметки создает элемент `EntityDataSource`, который выбирает курсы и позволяет добавить строки. Вы будете использовать обработчик события вставки, чтобы обновить свойство навигации `Department`, когда новый курс будет создан.

Далее создается элемент `DetailsView` для добавления новых объектов `Course`. Используются связанные поля для свойств `Course`. Вы будете должны ввести значение `CourseID`, потому что это поле не генерируется системой.

2. В файле `CoursesAdd.aspx.cs` после объявления класса добавьте поле класса, ссылающееся на элемент `DepartmentsDropDownList`:

```
private DropDownList departmentDropDownList;
```

3. Далее добавьте обработчик события `Init` элемента `DepartmentsDropDownList` в котором сохраните ссылку на элемент управления.

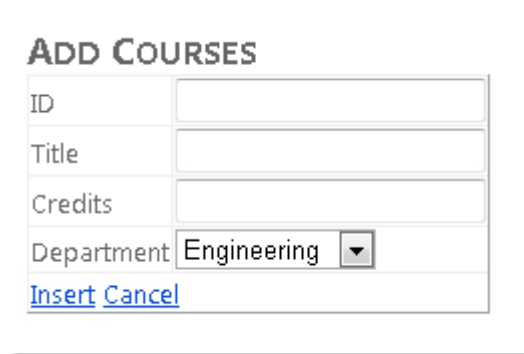
```
protected void DepartmentsDropDownList_Init(object sender,
EventArgs e)
{
    departmentDropDownList = sender as DropDownList;
}
```

4. Добавьте обработчик события `Inserting` элемента `DetailsView`:

```
protected void CoursesDetailsView_ItemInserting(object sender,
DetailsViewInsertEventArgs e)
{
    var departmentID =
Convert.ToInt32(departmentDropDownList.SelectedValue);
    e.Values["DepartmentID"] = departmentID;
}
```

Когда пользователь нажимает `Insert`, то событие `Inserting` вызывается, прежде чем будет вставлена новая запись. Обработчик получает `DepartmentID` от элемента `DropDownList` и использует его, чтобы установить значение, которое будет использоваться для свойства `DepartmentID` сущности `Course`.

5. Запустите страницу `CoursesAdd.aspx` в обозревателе.



ADD COURSES

ID	<input type="text"/>
Title	<input type="text"/>
Credits	<input type="text"/>
Department	Engineering ▼
<a href="#">Insert</a> <a href="#">Cancel</a>	

6. Введите ID, название и количество кредита и выберите кафедру, затем кликните **Insert (Вставка)**.
7. Откройте страницу `Courses.aspx` и выберите кафедру, ведущую новый курс. Проверьте, что новый курс успешно добавлен.

COURSES BY DEPARTMENT		
Select a Department <input type="text" value="Engineering"/>		
ID	Title	Credits
1050	Chemistry	4
1061	Physics	4
4062	New engineering course 5	

## Упражнение 2. Применение отношения многое ко многим

Отношения между Courses и People многие-ко-многим. В этом упражнении Вы будете добавлять и удалять отношения между Person Человек и Course курс лиц, обновляя свойства навигации соответствующих сущностей.

1. Создайте новую страницу *InstructorsCourses.aspx* которая основана на мастер странице Site.Master и добавьте код разметки элемента Content с именем Content2:

```

    <h2>Assign Instructors to Courses or Remove from Courses</h2>
    <br />
    <asp:EntityDataSource ID="InstructorsEntityDataSource" runat="server"
        ConnectionString="name=SchoolEntities"
DefaultContainerName="SchoolEntities"
        EnableFlattening="False"
        EntitySetName="People"
        Where="it.HireDate is not null" Select="it.LastName + ', ' +
it.FirstMidName AS Name, it.PersonID">
    </asp:EntityDataSource>
    Select an Instructor:
    <asp:DropDownList ID="InstructorsDropDownList" runat="server"
DataSourceID="InstructorsEntityDataSource"
        AutoPostBack="true" DataTextField="Name" DataValueField="PersonID"
        OnSelectedIndexChanged="InstructorsDropDownList_SelectedIndexChanged"
        OnDataBound="InstructorsDropDownList_DataBound">
    </asp:DropDownList>
    <h3>
        Assign a Course</h3>
    <br />
    Select a Course:
    <asp:DropDownList ID="UnassignedCoursesDropDownList" runat="server"
        DataTextField="Title" DataValueField="CourseID">
    </asp:DropDownList>
    <br />
    <asp:Button ID="AssignCourseButton" runat="server" Text="Assign"
OnClick="AssignCourseButton_Click" />
    <br />
    <asp:Label ID="CourseAssignedLabel" runat="server" Visible="false"
Text="Assignment successful"></asp:Label>
    <br />
    <h3>
        Remove a Course</h3>
    <br />
    Select a Course:
    <asp:DropDownList ID="AssignedCoursesDropDownList" runat="server"
        DataTextField="title" DataValueField="courseid">
    </asp:DropDownList>
    <br />
    <asp:Button ID="RemoveCourseButton" runat="server" Text="Remove"
OnClick="RemoveCourseButton_Click" />
    <br />

```



```
<asp:Label ID="CourseRemovedLabel" runat="server" Visible="false" Text="Removal
successful"></asp:Label>
```

Это код разметки создает элемент `EntityDataSource`, который извлекает имя и `PersonId` объекта `Person` для инструкторов. Элемент `DropDownList` связан с элементом `EntityDataSource`. `DropDownList` определяет обработчик для события `DataBound` с привязкой к данным. Вы будете использовать этот обработчик привязки двух раскрывающихся списков, которые отображают курсы.

Разметка также создает следующую группу элементов управления, чтобы использовать их для присвоения курса к выбранному инструктору:

- `DropDownList` для выбора назначаемого курса. Этот элемент будет заполнен курсами, которые в настоящее время не назначены выбранному инструктору.
- `Button` для начала выполнения действия.
- `Label` для отображения сообщения об ошибке.

Наконец, разметка также создает группу элементов управления, чтобы использовать удаление курса для выбранного инструктора.

2. В файл `InstructorsCourses.aspx.cs` добавьте выражение `using`:

```
using ContosoUniversity.DAL;
```

3. Добавьте метод для заполнения двух выпадающих списков, отображающих курсы:

```
private void PopulateDropDownLists()
{
    using (var context = new SchoolEntities())
    {
        var allCourses = (from c in context.Courses
                        select c).ToList();

        var instructorID =
Convert.ToInt32(InstructorsDropDownList.SelectedValue);
        var instructor = (from p in
context.People.Include("Courses")
                        where p.PersonID == instructorID
                        select p).First();

        var assignedCourses = instructor.Courses.ToList();
        var unassignedCourses =
allCourses.Except(assignedCourses.AsEnumerable()).ToList();

        UnassignedCoursesDropDownList.DataSource =
unassignedCourses;
        UnassignedCoursesDropDownList.DataBind();
        UnassignedCoursesDropDownList.Visible = true;

        AssignedCoursesDropDownList.DataSource =
assignedCourses;
        AssignedCoursesDropDownList.DataBind();
        AssignedCoursesDropDownList.Visible = true;
    }
}
```

В методе выполняется запрос получения всех курсов. Затем определяется, какие курсы назначаются для инструктора и заполняются соответствующие раскрывающиеся списки,

4. Добавьте обработчик события `Click` для кнопки `Assign`:

```

protected void AssignCourseButton_Click(object sender, EventArgs e)
{
    using (var context = new SchoolEntities())
    {
        var instructorID =
Convert.ToInt32(InstructorsDropDownList.SelectedValue);
        var instructor = (from p in context.People
                           where p.PersonID == instructorID
                           select p).First();

        var courseID =
Convert.ToInt32(UnassignedCoursesDropDownList.SelectedValue);
        var course = (from c in context.Courses
                       where c.CourseID == courseID
                       select c).First();
        instructor.Courses.Add(course);
        try
        {
            context.SaveChanges();
            PopulateDropDownLists();
            CourseAssignedLabel.Text = "Assignment successful.";
        }
        catch (Exception)
        {
            CourseAssignedLabel.Text = "Assignment
unsuccessful.";
            //Add code to log the error.
        }
        CourseAssignedLabel.Visible = true;
    }
}

```

В методе реализовывается получение объекта `Person` для выбранного инструктора, получение объекта `Course` для выбранного курса, и добавление выбранного курса для свойства навигации `Courses` для сущности `Person`. Затем сохраняются изменения в базу данных и повторно заполняются раскрывающиеся списки, так что результаты можно увидеть сразу.

5. Добавьте обработчик события `Click` для кнопки `Remove`:

```

protected void RemoveCourseButton_Click(object sender, EventArgs
e)
{
    using (var context = new SchoolEntities())
    {
        var instructorID =
Convert.ToInt32(InstructorsDropDownList.SelectedValue);
        var instructor = (from p in context.People
                           where p.PersonID == instructorID
                           select p).First();

        var courseID =
Convert.ToInt32(AssignedCoursesDropDownList.SelectedValue);
        var courses = instructor.Courses;
        var courseToRemove = new Course();
        foreach (Course c in courses)
        {
            if (c.CourseID == courseID)
            {
                courseToRemove = c;
                break;
            }
        }
        try
        {

```

```

        courses.Remove(courseToRemove);
        context.SaveChanges();
        PopulateDropDownLists();
        CourseRemovedLabel.Text = "Removal successful.";
    }
    catch (Exception)
    {
        CourseRemovedLabel.Text = "Removal unsuccessful.";
        //Add code to log the error.
    }
    CourseRemovedLabel.Visible = true;
}
}

```

В методе реализовывается получение объекта `Person` для выбранного инструктора, получение объекта `Course` для выбранного курса, и удаление выбранного курса для свойства навигации `Courses` для сущности `Person`. Затем сохраняются изменения в базу данных и повторно заполняются раскрывающиеся списки, так что результаты можно увидеть сразу.

6. Добавьте реализацию обработчика события загрузки страницы `Page_Load`, в котором реализуется скрывание сообщений об ошибках в случае их отсутствия

```

protected void Page_Load(object sender, EventArgs e)
{
    CourseAssignedLabel.Visible = false;
    CourseRemovedLabel.Visible = false;
}

```

7. Добавьте обработчики событий `DataBound` и `SelectedIndexChanged` выпадающего списка инструкторов of the instructors drop-down list to populate the courses drop-down lists:

```

protected void InstructorsDropDownList_DataBound(object sender,
EventArgs e)
{
    PopulateDropDownLists();
}

protected void InstructorsDropDownList_SelectedIndexChanged(object
sender, EventArgs e)
{
    PopulateDropDownLists();
}

```

8. Запустите страницу в обозревателе.

## ASSIGN INSTRUCTORS TO COURSES OR REMOVE FROM COURSES

Select an Instructor:

### ASSIGN A COURSE

Select a Course:

### REMOVE A COURSE

Select a Course:

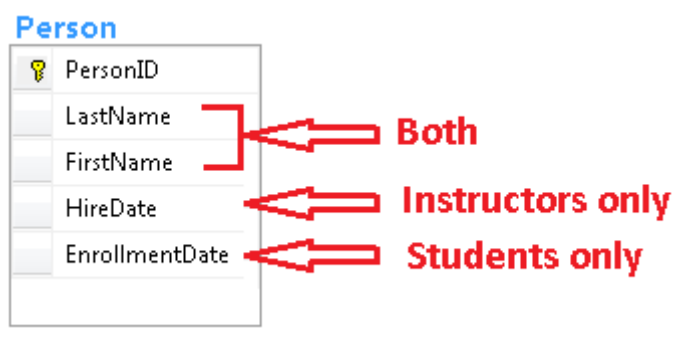
9. Выберите инструктора. В выпадающем списке **Assign a Course** выберите курс и нажмите кнопку **Assign**.
10. Перейдите на страницу списка инструкторов. Выберите инструктора и убедитесь, что курс добавился для этого инструктора.

## Lab 6. Реализация наследования Table-per-Hierarchy

В этой работе будет показано, как реализуется наследование в модели данных путем создания наследуемых сущностей, хранящихся в одной таблице базы данных. Вы добавите в модель данных производные сущности и внесете изменения в существующие страницы для их применения.

### Варианты наследования *Table-per-Hierarchy* и *Table-per-Type*

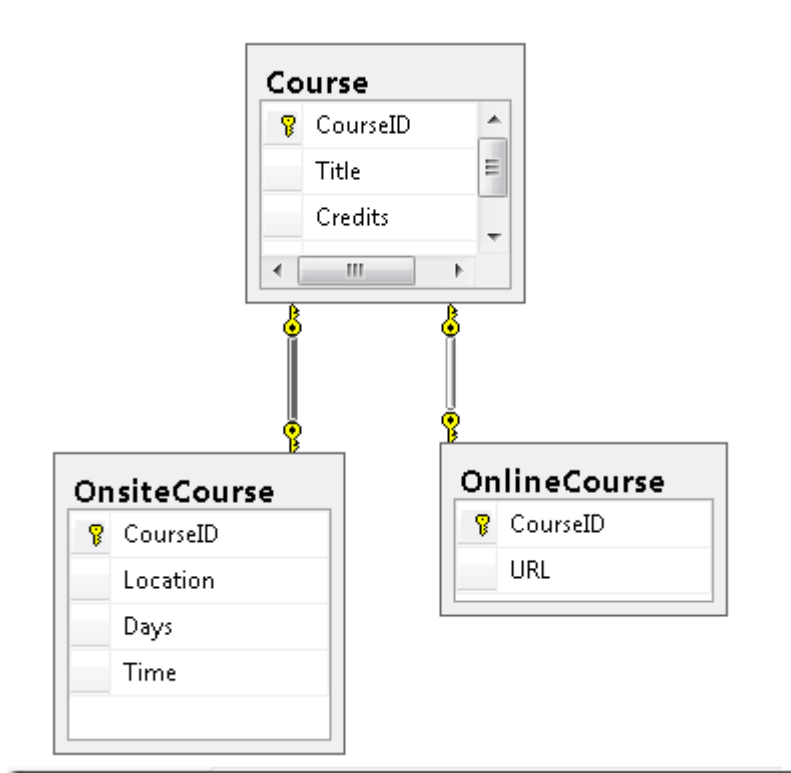
База данных может хранить информацию о связанных объектах в одной таблице или в нескольких таблицах. Например, в базе данных *School*, таблица *Person* включает в себя информацию о студентах и преподавателях в одной таблице. Некоторые из столбцов применяются только к инструкторам (*HireDate*), некоторые только к студентам (*EnrollmentDate*), а некоторые как (*LastName*, *FirstName*) применяются для обеих сущностей.



Вы можете создать новые сущности *Instructor* и *Student*, которые будут наследоваться от сущности *Person*. Этот паттерн создания наследуемых сущностей,

хранящихся в одной таблице базы данных называется *table-per-hierarchy* (TPH) наследование.

Для курсов база *School* применяет другой паттерн. Online курсы и onsite курсы хранятся в разных таблицах, каждая из которых имеет внешний ключ, связанный с ключом таблицы *Course*.



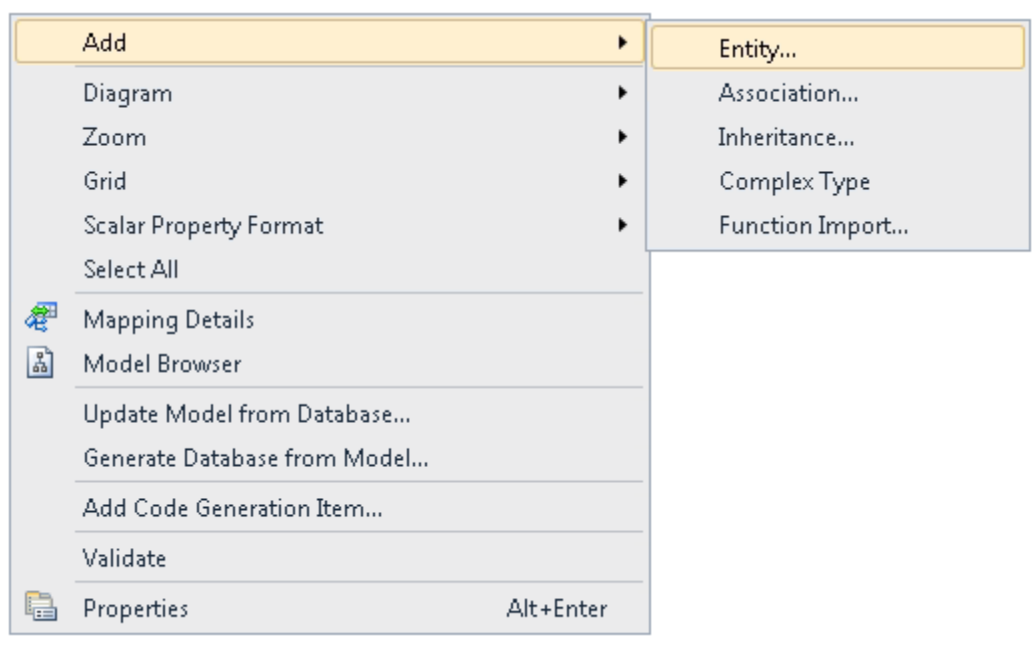
Вы можете создать в модели данных сущности *OnlineCourse* и *OnsiteCourse*, наследуемые от сущности *Course*. Этот паттерн создает наследуемые сущности, находящихся в разных таблицах, а в базовой сущности хранятся общие данные. Такое паттерн называется *table per type* (TPT) наследование.

Паттерн TPH обеспечивает, как правило, более высокую производительность в Entity Framework, чем TPT, потому что TPT может привести к сложным запросам слияния. Для реализации наследования TPH следует выполнить следующие действия:

- Создать типы сущностей *Instructor* и *Student* которые наследуются от *Person*.
- Переместить свойства из базовой сущности, которые относятся к производным сущностям в производные сущности.
- Установить ограничения на свойства в производных типов.
- Сделать базовую сущность абстрактной.
- Сопоставить каждый полученный объект в таблицу *Person* с условием, которое определяет, как определить, представляет строка объект *Person* или производный тип.

### ***Упражнение 1. Добавление производных сущностей *Instructor* и *Student****

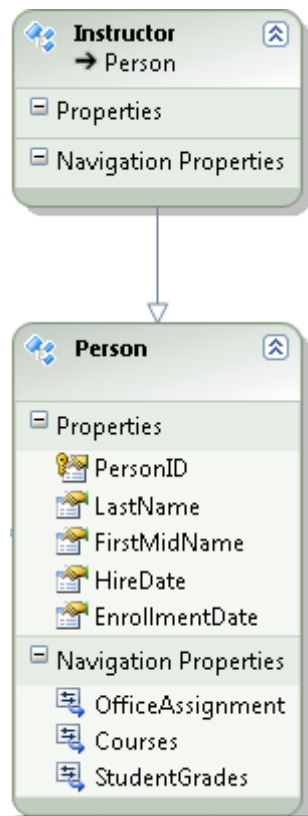
1. Откройте файл *SchoolModel.edmx* и на пустом месте дизайнера кликните правую кнопку мыши, выберите **Add (Добавить)**, далее – **Entity (Сущность)**.



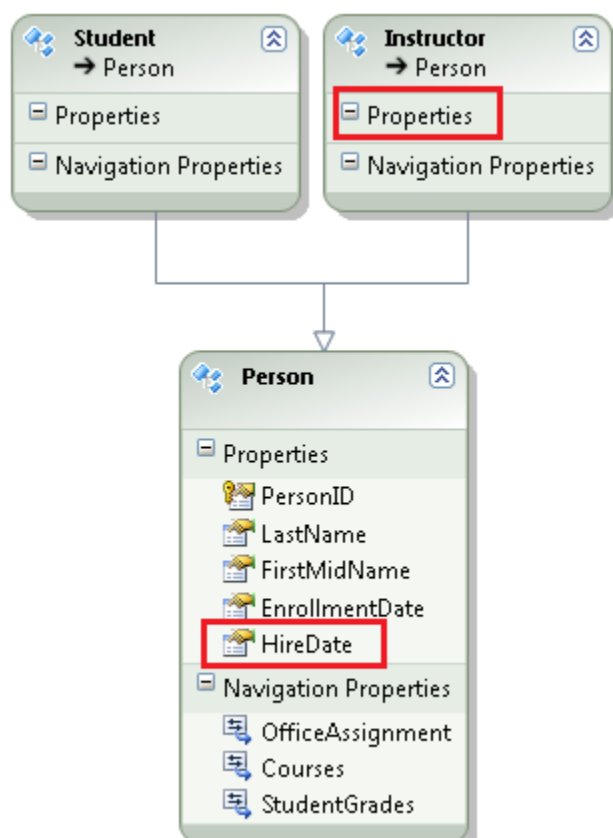
2. В окне **Add Entity (Добавление сущности)** укажите имя – `Instructor` и базовый тип (**Base type**) `Person`. Нажмите **OK**.

A screenshot of the 'Add Entity' dialog box. The dialog has a title bar with a question mark and a close button. It contains two main sections: 'Properties' and 'Key Property'. In the 'Properties' section, there are three fields: 'Entity name' with the value 'Instructor', 'Base type' with a dropdown menu showing 'Person', and 'Entity Set' with the value 'People'. In the 'Key Property' section, there is a checkbox labeled 'Create key property' which is checked, a 'Property name' field with the value 'Id', and a 'Property type' dropdown menu showing 'Int32'. At the bottom right, there are two buttons: 'OK' (highlighted with a blue border) and 'Cancel'.

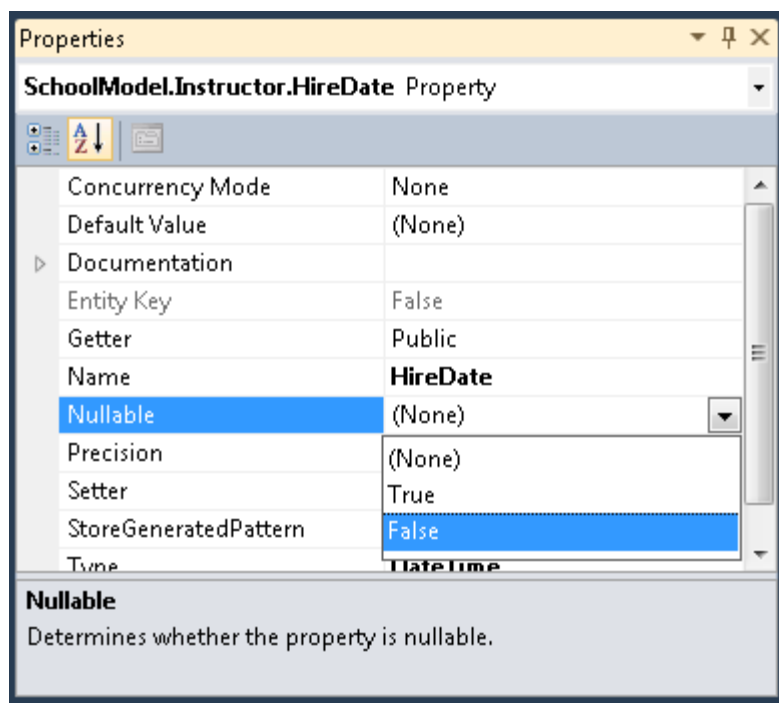
Дизайнер создаст сущность `Instructor`, которая наследуется от сущности `Person`. Новая сущность не имеет свойств.



3. Повторите аналогичные действия для создания сущности `Student`, которая также наследуется от `Person`.
4. В данной задаче только инструкторы имеют дату приема на работу, поэтому перенесите свойство `HireDate` из сущности `Person` в сущность `Instructor`, используя операции **Cut и Paste**.



5. Дата приема на работу для сущности **Instructor** не может быть пустой (null), поэтому установите свойству **HireDate** сущности **Instructor**, свойство **Nullable** (Допускает Null) значение **False**.

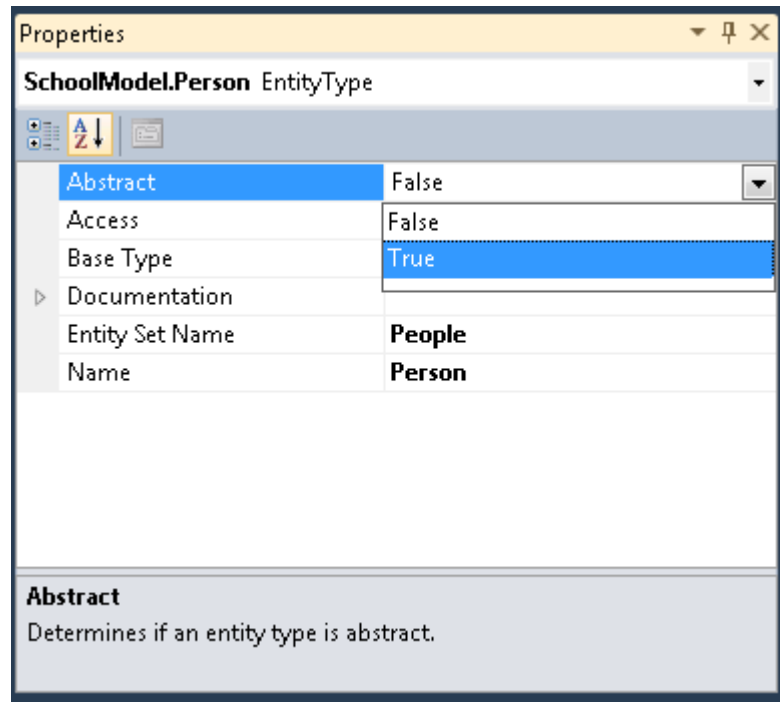


6. Переместите свойство **EnrollmentDate** из сущности **Person** в сущность **Student**.  
 7. Для свойства **Nullable** (Допускает Null) установите **False** свойству **EnrollmentDate**.



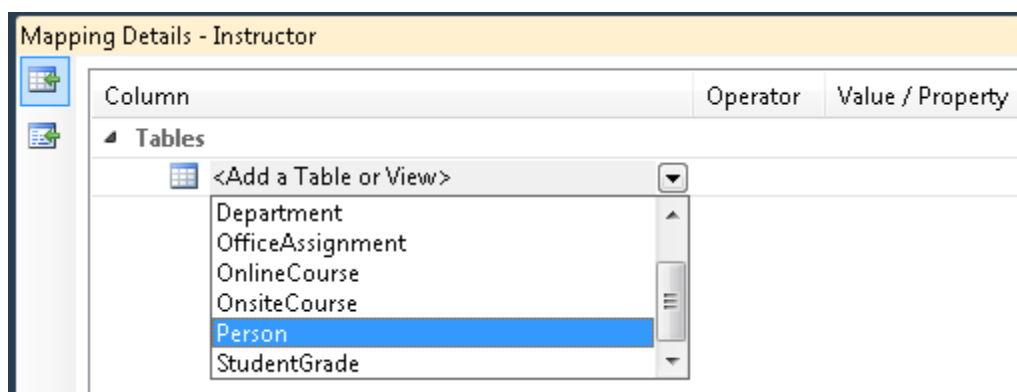
Теперь, когда сущность `Person` имеет только свойства, которые являются общими для инструкторов и студентов, сущность `Person` может быть использована только в качестве базового объекта в структуре наследования. Таким образом, необходимо гарантировать, что эта базовая сущность никогда не будет рассматривается как самостоятельный субъект, т.е. сделать ее абстрактной.

8. В окне свойств сущности `Person` для свойства **Abstract** (Абстрактный) установите **True**.

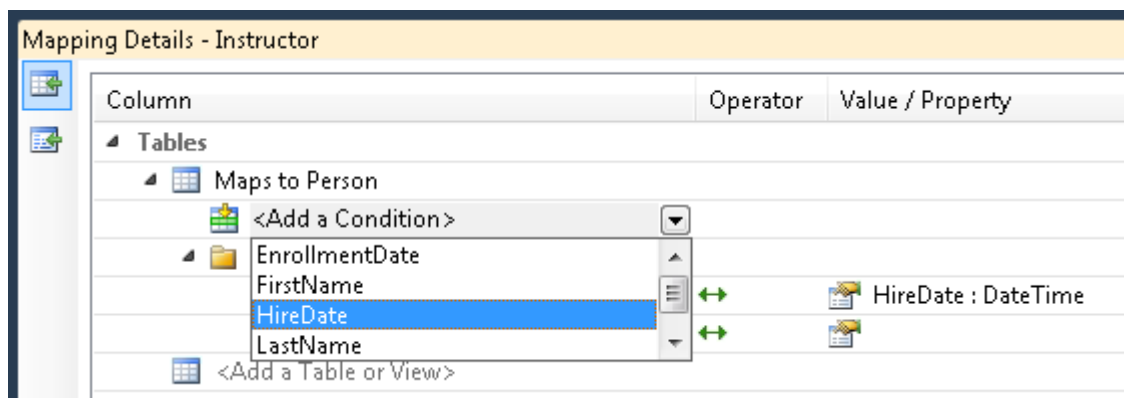


Теперь вы должны показать Entity Framework, как отличить сущности `Instructor` и `Student` в базе данных.

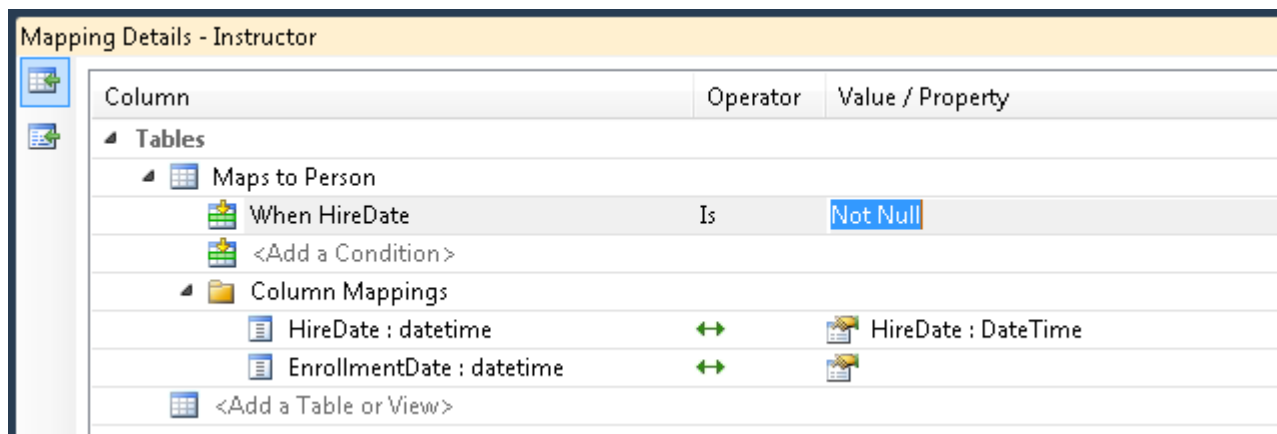
9. В контекстном меню сущности `Instructor` выберите команду **Table Mapping** (Таблица сопоставления).
10. В окне **Mapping Details** (Сведения о сопоставлении) кликните **Add a Table or View** (Добавление таблицы или представления) и выберите **Person**.



11. Кликните **Add a Condition** и выберите **HireDate**.



12. Измените **Operator** на **Is** и **Value / Property** установите **Not Null**.



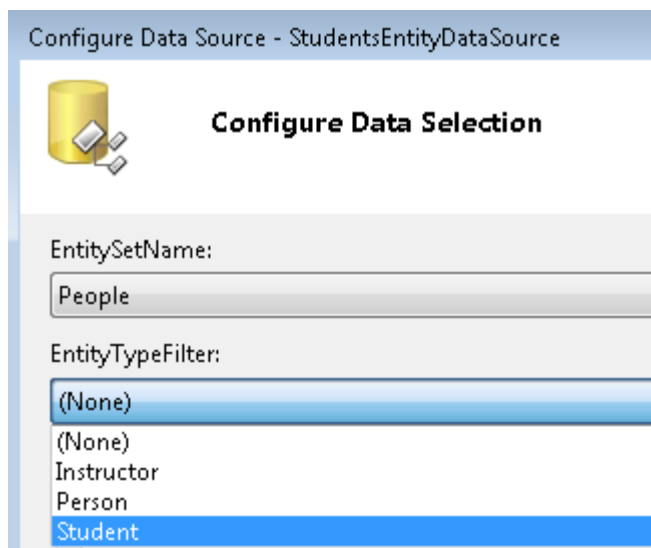
13. Повторите процедуру для сущности `Students`, указав, что эта сущность отображает в таблице `Person`, когда `EnrollmentDate` не ноль.
14. Сохраните и закройте модель данных.
15. Постройте проект, чтобы создать новые объекты и сделать их доступными в конструкторе.

## Упражнение 2. Применение сущностей *Instructor* и *Student*

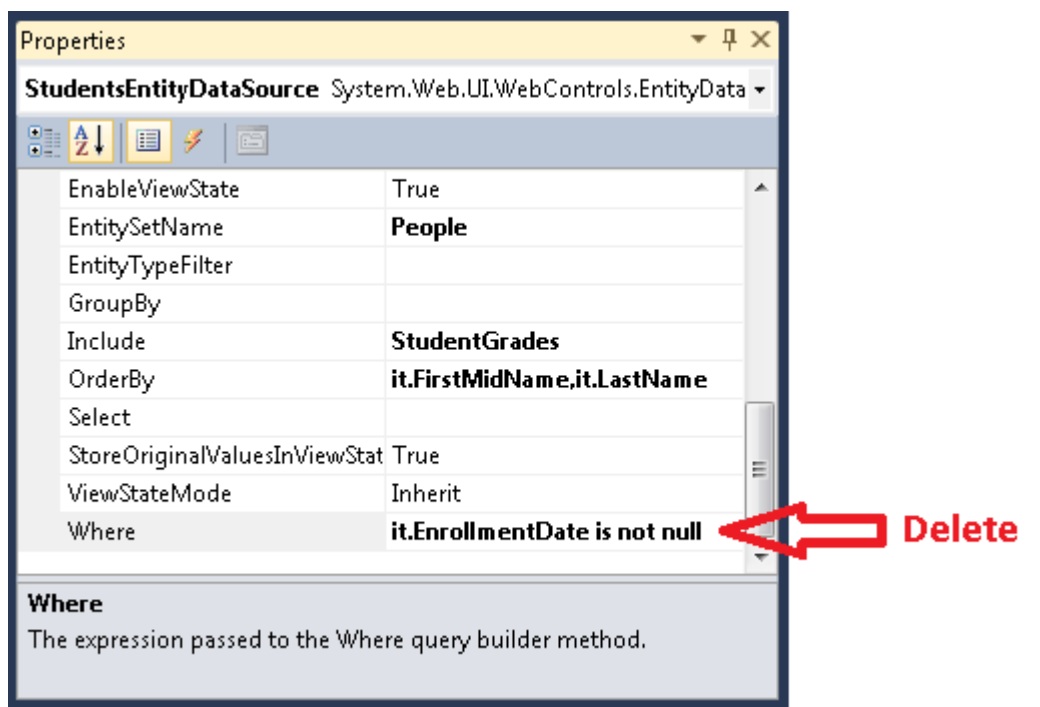
Когда вы создали веб-страницы, которые работают с данными о студентах и инструкторах, вы привязали данные сущности `Person` и отфильтровали свойство `HireDate` или `EnrollmentDate` для ограничения возвращаемых данных для студентов или преподавателей.

Тем не менее, теперь, когда связали каждый элемент источника данных с сущностью `Person`, вы можете указать, что только типы объектов `Student` или `Instructor` должны быть выбраны. Поскольку `Entity Framework` знает, как отличить студентов и преподавателей в наборе `Person`, вы можете удалить настройки свойства `Where`.

Вы можете это сделать в мастере `Visual Studio Designer`, указать тип объекта в элементе `EntityDataSource`, как показано в следующем примере



В окне свойств **Properties** вы можете удалить значение *Where*, как показано ниже в примере:



Далее необходимые изменения вы выполните в коде разметки.

1. Откройте страницу *Students.aspx* в режиме разметки.
2. В элементе *StudentsEntityDataSource* удалите атрибут *Where* и добавьте атрибут *EntityTypeFilter="Student"*. Код разметки должен быть следующим:

```
<asp:EntityDataSource ID="StudentsEntityDataSource" runat="server"
    ConnectionString="name=SchoolEntities"
    DefaultContainerName="SchoolEntities"
    EnableDelete="True" EnableFlattening="False" EnableUpdate="True"
    EntitySetName="People" Include="StudentGrades"
    OrderBy="it.LastName" EntityTypeFilter="Student">
</asp:EntityDataSource>
```

Установка атрибута `EntityTypeFilter` гарантирует, что `EntityDataSource` будет выбирать только указанный тип. Если вы хотите, чтобы получить оба типа `Student` и `Instructor`, вы должны не устанавливать этот атрибут.

- Для элемента `SearchEntityDataSource` добавьте атрибут `EntityTypeFilter="Student"` и измените атрибут `Where` выбирает студентов. Код разметки должен быть следующим:

```
<asp:EntityDataSource ID="SearchEntityDataSource" runat="server"
    ConnectionString="name=SchoolEntities"
    DefaultContainerName="SchoolEntities" EnableFlattening="False"
    EntitySetName="People" EntityTypeFilter="Student"
    Where="it.FirstMidName Like '%' + @StudentName + '%' or it.LastName Like
    '%' + @StudentName + '%'" >
    <WhereParameters>
        <asp:ControlParameter ControlID="SearchTextBox" Name="StudentName"
        PropertyName="Text"
        Type="String" DefaultValue="" />
    </WhereParameters>
</asp:EntityDataSource>
```

- Запустите страницу, чтобы проверить, что все работает как раньше.

### STUDENT LIST

	ID	Name	Enrollment Date	Number of Courses
<a href="#">Edit</a> <a href="#">Delete</a>	14	Walker, Alexandra	9/1/2000	2
<a href="#">Edit</a> <a href="#">Delete</a>	30	Shan, Alicia	9/1/2003	2
<a href="#">Edit</a> <a href="#">Delete</a>	28	White, Anthony	9/1/2001	2
<a href="#">Edit</a> <a href="#">Delete</a>	13	Anand, Arturo	9/1/2003	2
<a href="#">Edit</a> <a href="#">Delete</a>	22	Alexander, Carson	9/1/2005	3
<a href="#">Edit</a> <a href="#">Delete</a>	19	Bryant, Carson	9/1/2001	1
<a href="#">Edit</a> <a href="#">Delete</a>	15	Powell, Carson	9/1/2004	1
<a href="#">Edit</a> <a href="#">Delete</a>	26	Rogers, Cody	9/1/2002	2
<a href="#">Edit</a> <a href="#">Delete</a>	16	Jai, Damien	9/1/2001	1
<a href="#">Edit</a> <a href="#">Delete</a>	33	Gao, Erica	1/30/2003	0
1 2 3				

### FIND STUDENTS BY NAME

Enter any part of the name

Name	Enrollment Date
Barzdukas, Gytis	9/1/2005
Justice, Peggy	9/1/2001
Li, Yan	9/1/2002
Norman, Laura	9/1/2003
Olivotto, Nino	9/1/2005
Tang, Wayne	9/1/2005
Alonso, Meredith	9/1/2002
Lopez, Sophia	9/1/2004
Browning, Meredith	9/1/2000
Anand, Arturo	9/1/2003
1 2 3	

5. Обновите следующие страницы, которые вы создали в предыдущих упражнениях, так чтобы они использовали новые сущности *Student* и *Instructor* вместо сущности *Person*, а затем запустите их, чтобы убедиться, что они работают как раньше:

- На странице *StudentsAdd.aspx* в элемент *StudentsEntityDataSource* добавьте *EntityTypeFilter="Student"*:

```
<asp:EntityDataSource ID="StudentsEntityDataSource" runat="server"
    ConnectionString="name=SchoolEntities" DefaultContainerName="SchoolEntities"
    EnableDelete="True" EnableFlattening="False" EnableUpdate="True"
    EnableInsert="True" EntityTypeFilter="Student"
    EntitySetName="People" Include="StudentGrades">
</asp:EntityDataSource>
</asp:EntityDataSource>
```

- На странице *About.aspx* в элемент *StudentStatisticsEntityDataSource* добавьте *EntityTypeFilter="Student"* и удалите *Where="it.EnrollmentDate is not null"*. Код разметки должен быть таким:

```
<asp:EntityDataSource ID="StudentStatisticsEntityDataSource" runat="server"
    ConnectionString="name=SchoolEntities" DefaultContainerName="SchoolEntities"
    EnableFlattening="False" EntitySetName="People" EntityTypeFilter="Student"
    GroupBy="it.EnrollmentDate"
    OrderBy="it.EnrollmentDate"
    Select="it.EnrollmentDate, Count(it.EnrollmentDate) AS NumberOfStudents">
</asp:EntityDataSource>
```

Date of Enrollment	Students
9/1/2000	2
9/1/2001	5
9/1/2002	3
1/30/2003	1
9/1/2003	3
9/1/2004	5
9/1/2005	6
1/1/2011	1

- На страницах *Instructors.aspx* и *InstructorsCourses.aspx* в элемент *InstructorsEntityDataSource* добавьте *EntityTypeFilter="Instructor"* и удалите *Where="it.HireDate is not null"*.

Код страницы *Instructors.aspx*:

```
<asp:EntityDataSource ID="InstructorsEntityDataSource" runat="server"
```

```

        ConnectionString="name=SchoolEntities"
DefaultContainerName="SchoolEntities" EnableFlattening="False"
        EntitySetName="People" EntityTypeFilter="Instructor"
Include="OfficeAssignment">
    </asp:EntityDataSource>

```

## INSTRUCTORS

	ID	Name	Hire Date	Office Assignment
<a href="#">Edit</a> <a href="#">Select</a>	1	Abercrombie, Kim	3/11/1995	17 Smith
<a href="#">Edit</a> <a href="#">Select</a>	4	Fakhouri, Fadi	8/6/2002	29 Adams
<a href="#">Edit</a> <a href="#">Select</a>	5	Harui, Roger	7/1/1998	37 Williams
<a href="#">Edit</a> <a href="#">Select</a>	18	Zheng, Roger	2/12/2004	143 Smith
<a href="#">Edit</a> <a href="#">Select</a>	25	Kapoor, Candace	1/15/2001	57 Adams
<a href="#">Edit</a> <a href="#">Select</a>	27	Serrano, Stacy	6/1/1999	271 Williams
<a href="#">Edit</a> <a href="#">Select</a>	31	Stewart, Jasmine	10/12/1997	131 Smith
<a href="#">Edit</a> <a href="#">Select</a>	32	Xu, Kristen	7/23/2001	203 Williams
<a href="#">Edit</a> <a href="#">Select</a>	34	Van Houten, Roger	12/7/2000	213 Smith

## COURSES TAUGHT

	ID	Title	Department
<a href="#">Select</a>	2042	Literature	English
<a href="#">Select</a>	3141	Trigonometry	Mathematics
<a href="#">Select</a>	4022	Microeconomics	Economics
<a href="#">Select</a>	4041	Macroeconomics	Economics
<a href="#">Select</a>	4061	Quantitative	Economics
<a href="#">Select</a>	4062	New engineering course	Engineering
<a href="#">Select</a>	4063	new course	Economics

## COURSE DETAILS

ID	2042
Title	Literature
Credits	4
Department	English
Location	225 Adams
URL	

## STUDENT GRADES

ID	Name	Grade
6	Li, Yan	3.50
7	Norman, Laura	4.00
8	Olivotto, Nino	3.00

Код разметки *InstructorsCourses.aspx*:

```

<asp:EntityDataSource ID="InstructorsEntityDataSource" runat="server"
    ConnectionString="name=SchoolEntities" DefaultContainerName="SchoolEntities"
    EnableFlattening="False"
    EntitySetName="People"
    EntityTypeFilter="Instructor"
    Select="it.LastName + ', ' + it.FirstMidName AS Name, it.PersonID">
</asp:EntityDataSource>

```

## ASSIGN INSTRUCTORS TO COURSES OR REMOVE FROM COURSES

Select an Instructor:

### ASSIGN A COURSE

Select a Course:

### REMOVE A COURSE

Select a Course:

В результате этих изменений, вы улучшили ремонтпригодность приложения Contoso в несколькими способами. Вы перенесли выборку данных и логику проверки из слоя интерфейса (.aspx разметки) и сделали его неотъемлемой частью слоя доступа к данным. Это помогает изолировать код приложения от изменений, которые вы могли бы сделать в будущем в схеме базы данных или модели данных. Например, вы могли бы решить, что студенты могут быть наняты в качестве помощников преподавателей и, следовательно, получить дату найма. Затем можно добавить новое свойство определяющее студентов-инструкторов и обновить модель данных. Код веб-приложения не нужно будет изменять исключением случаев, когда вы хотите, чтобы показать дату проката для студентов. Еще одно преимущество добавления классов инструкторов и студентов состоит в том, что код становится более понятным, чем когда использовались только объекты Person, которые на самом деле студенты или преподаватели.

## Lab 7. Использование хранимых процедур

В этой работе вы узнаете, как использовать хранимые процедуры, чтобы получить больше контроля над доступом к базе данных.

Платформа Entity Framework поддерживает возможность использования хранимых процедур для доступа к базе данных. Для любой сущности вы можете указать хранимую процедуру, чтобы использовать ее для создания, обновления или удаления объектов этого типа. Тогда в модели данных вы можете добавить ссылки на хранимые процедуры, которые можно использовать для выполнения таких задач, как получение наборов данных.

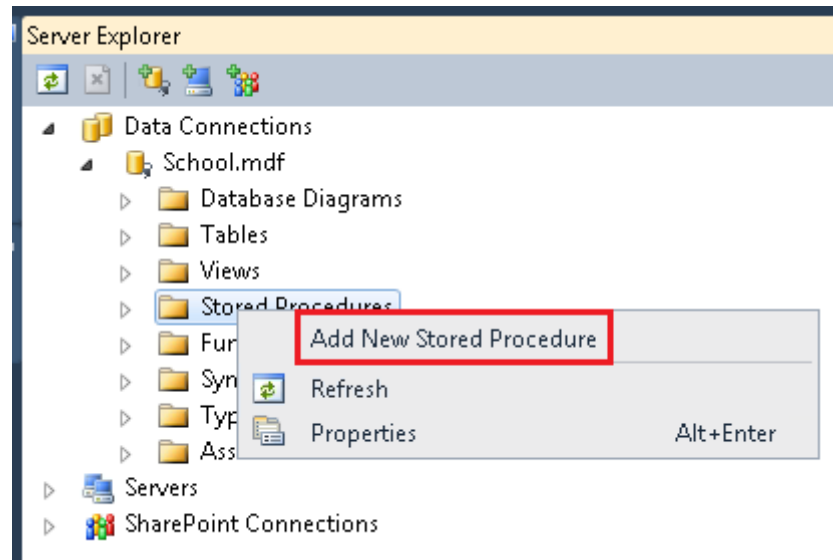
Использование хранимых процедур является общим требованием для доступа к базе данных. В некоторых случаях администратор базы данных может потребовать, чтобы весь доступ к базе данных проходил через хранимые процедуры в целях безопасности. В других случаях вы можете построить бизнес-логику в некоторых процессах, которые Entity Framework использует при обновлении базы данных. Например, всякий раз, когда объект удаляется, возможно, вы хотите скопировать его в архивной базе данных. Или, когда ряд обновляется вы можете написать строку таблицы регистрации, которая записывает, кто сделал изменения. Вы можете выполнить эти виды задач в хранимой процедуре, которая вызывается, когда Entity Framework удаляет или обновляет объект.

В этой работе Вы измените способ обращения к базе данных для некоторых уже созданных страниц. Вы создадите хранимые процедуры в базе данных для вставки объектов Student и Instructor. Вы добавите их в модели данных, укажите, что Entity Framework должен использовать их для добавления объектов Student и Instructor в базу данных. Вы также создадите хранимую процедуру, которую можно использовать для извлечения объектов Course

### ***Упражнение 1. Создание хранимых процедур в базе данных***

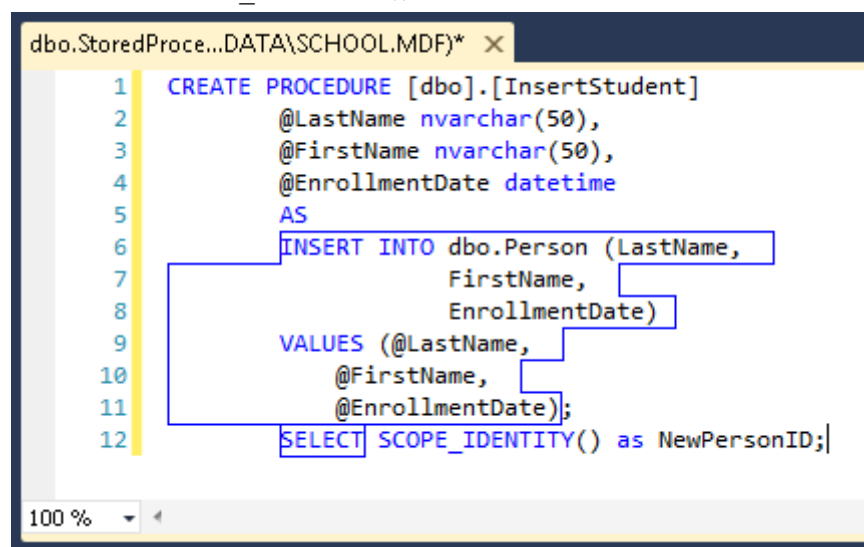
1. В **Server Explorer** (Обозревателе серверов) раскройте узел *School.mdf*, далее раскройте узел **Stored Procedures** (Хранимые процедуры) и просмотрите список уже существующих процедур в базе данных *School*.

2. Выделите папку **Stored Procedures (Хранимые процедуры)** и в контекстном меню выберите команду **Add New Stored Procedure (Добавить новую хранимую процедуру)**.



3. Скопируйте следующий SQL выражения и вставьте в окно редактирования хранимой процедуры, заменив шаблон:

```
CREATE PROCEDURE [dbo].[InsertStudent]
    @LastName nvarchar(50),
    @FirstName nvarchar(50),
    @EnrollmentDate datetime
AS
    INSERT INTO dbo.Person (LastName,
                            FirstName,
                            EnrollmentDate)
    VALUES (@LastName,
            @FirstName,
            @EnrollmentDate);
    SELECT SCOPE_IDENTITY() as NewPersonID;
```



Student имеет четыре свойства: PersonID, LastName, FirstName и EnrollmentDate. В базе данных генерируется значение идентификатора автоматически, и хранимая



процедура принимает параметры для трех других свойств. Хранимая процедура возвращает значение ключа записи новой строки так, чтобы платформа Entity Framework могла отслеживать версию объекта в памяти.

4. Сохраните и закройте хранимую процедуру.
5. Создайте хранимую процедуру `InsertInstructor`, используя следующее SQL выражение:

```
CREATE PROCEDURE [dbo].[InsertInstructor]
    @LastName nvarchar(50),
    @FirstName nvarchar(50),
    @HireDate datetime
AS
INSERT INTO dbo.Person (LastName,
                        FirstName,
                        HireDate)
VALUES (@LastName,
        @FirstName,
        @HireDate);
SELECT SCOPE_IDENTITY() as NewPersonID;
```

6. Создайте две хранимые процедуры обновления объектов `Student` и `Instructor`. (база данных имеет хранимую процедуру удаления `DeletePerson`, которая может применяться и для `Instructor` и для `Student`).

```
CREATE PROCEDURE [dbo].[UpdateStudent]
    @PersonID int,
    @LastName nvarchar(50),
    @FirstName nvarchar(50),
    @EnrollmentDate datetime
AS
UPDATE Person SET LastName=@LastName,
                FirstName=@FirstName,
                EnrollmentDate=@EnrollmentDate
WHERE PersonID=@PersonID;
```

```
CREATE PROCEDURE [dbo].[UpdateInstructor]
    @PersonID int,
    @LastName nvarchar(50),
    @FirstName nvarchar(50),
    @HireDate datetime
AS
UPDATE Person SET LastName=@LastName,
                FirstName=@FirstName,
                HireDate=@HireDate
WHERE PersonID=@PersonID;
```

7. Создайте хранимую процедуру, которая будет считывать данные о курсах:

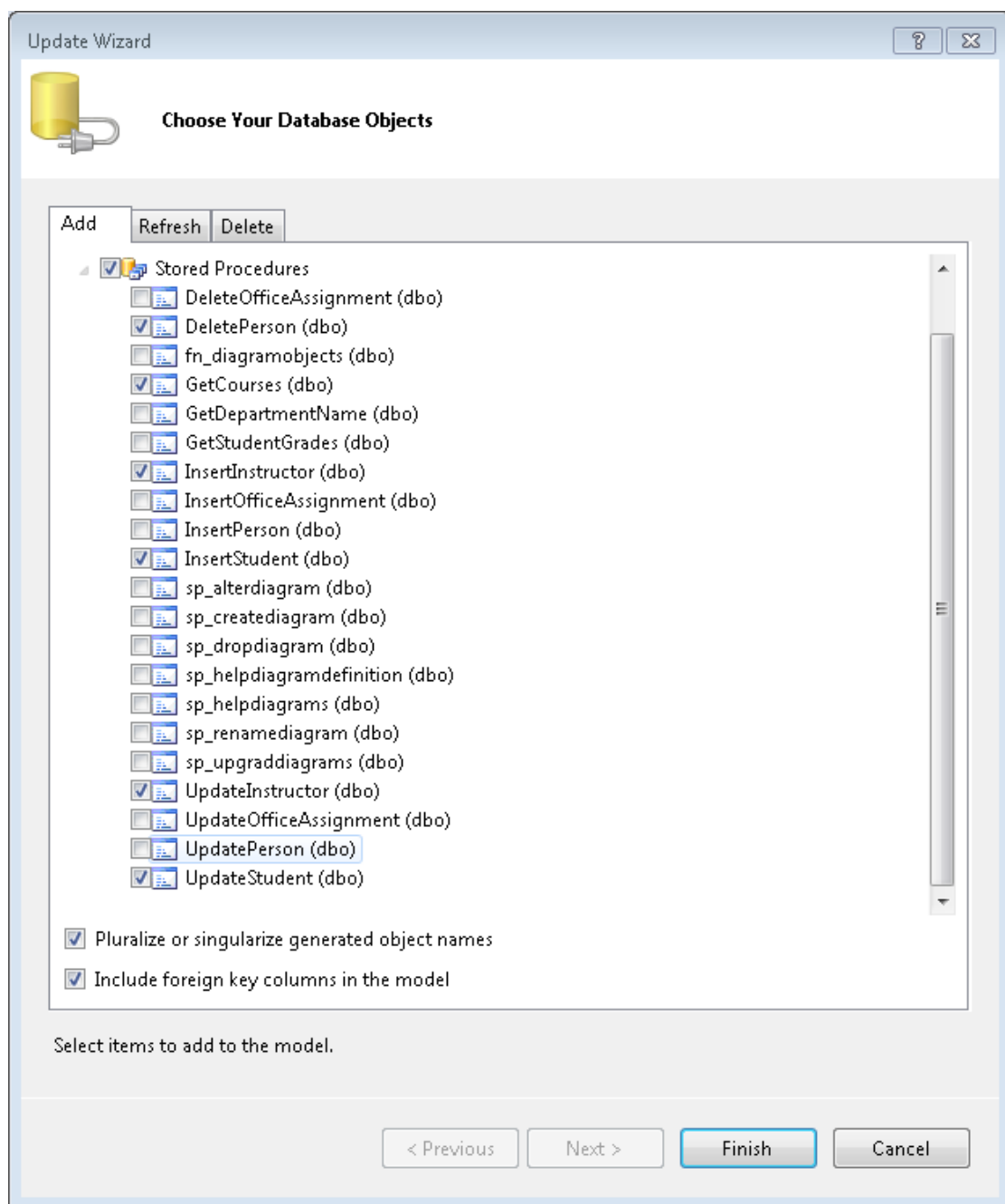
```
CREATE PROCEDURE [dbo].[GetCourses]
AS
SELECT CourseID, Title, Credits, DepartmentID FROM dbo.Course
```

## Упражнение 2. Размещение хранимых процедур в модели данных

### Добавление хранимых процедур в модель данных

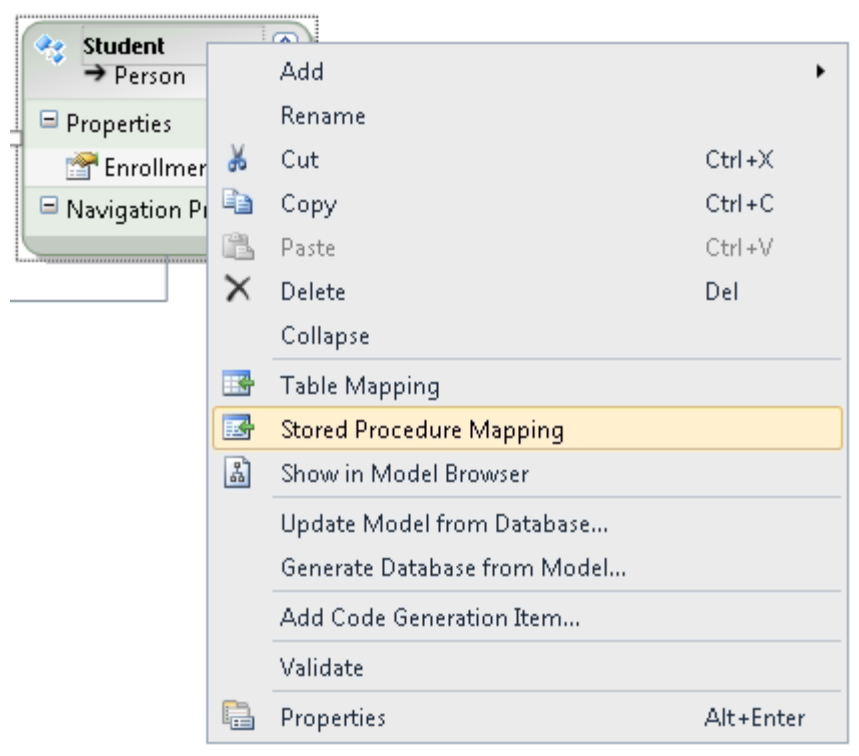
Хранимые процедуры, которые определены в базе данных, должны быть добавлены к модели данных, чтобы сделать их доступными для Entity Framework.

1. Откройте модель *SchoolModel.edmx* в режиме дизайнера.
2. В контекстном меню поверхности модели выберите **Update Model from Database (Обновить модель из базы данных)**.
3. На вкладке **Add(Добавить)** окна **Choose Your Database Objects (Выбор объектов базы данных)** раскройте **Stored Procedures (Хранимые процедуры)**, выберите вновь добавленные процедуры и процедуру *DeletePerson*.
4. Включите оба флажка генерации имен и включения внешних ключей, нажмите **Finish (Готово)**.

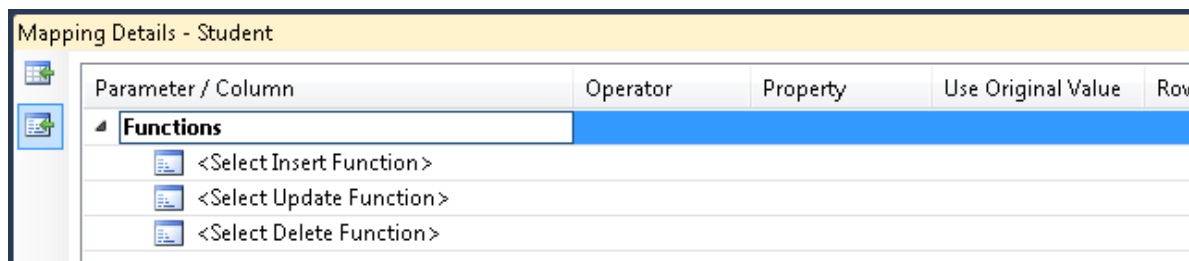


## Сопоставление хранимых процедур

5. В дизайнере модели в контекстном меню `Student` выберите **Stored Procedure Mapping** (Сопоставление хранимых процедур).



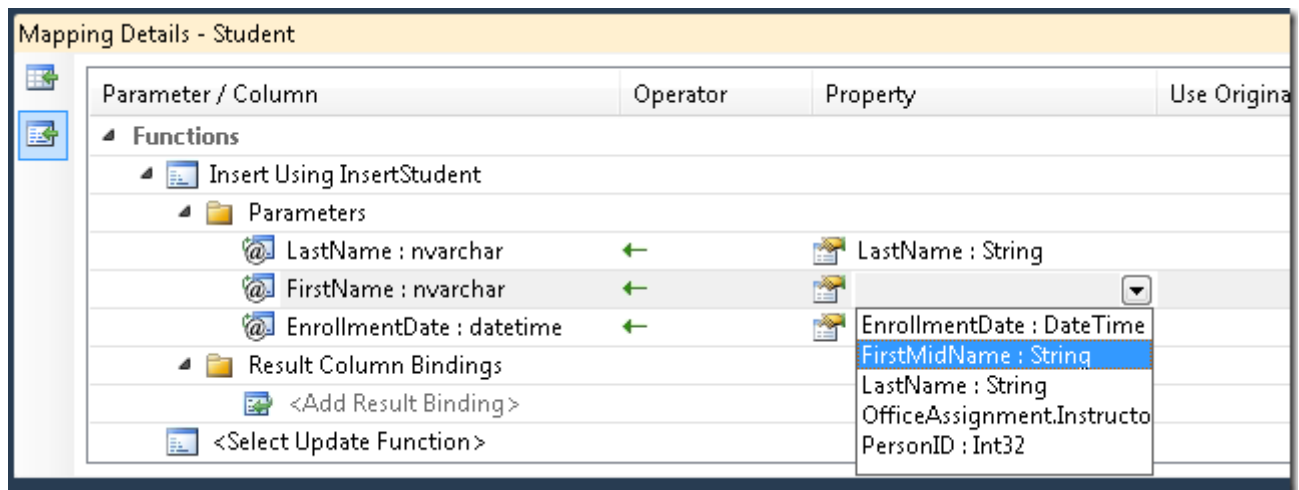
Появится окно **Mapping Details (Сведения о сопоставлении)**, в котором можно указать хранимые процедуры, которые Entity Framework должен использовать для вставки, обновления и удаления объектов данного типа.



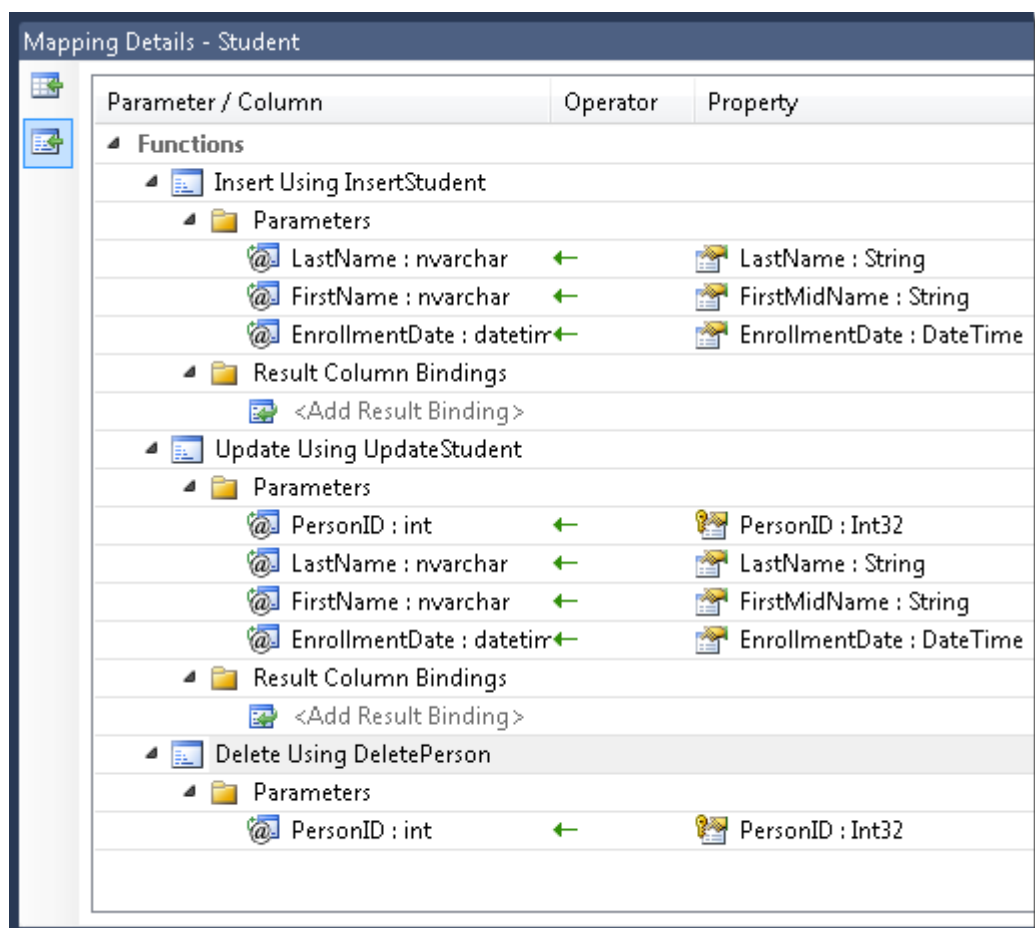
6. Выберите в списке **Insert (Вставка)** функцию **InsertStudent**.

Окно показывает список параметров хранимой процедуры, каждый из которых должен быть сопоставлен со свойствами сущности. `LastName` и `EnrollmentDate` отображаются автоматически, потому что имена совпадают, а вот свойства `FirstName` нет (потому, что в вы изменили имя свойства `FirstName` на `FirstMidName`).

7. Выберите `FirstMidName` из выпадающего списка напротив свойства `FirstName`, который показывает доступные свойства сущностей.



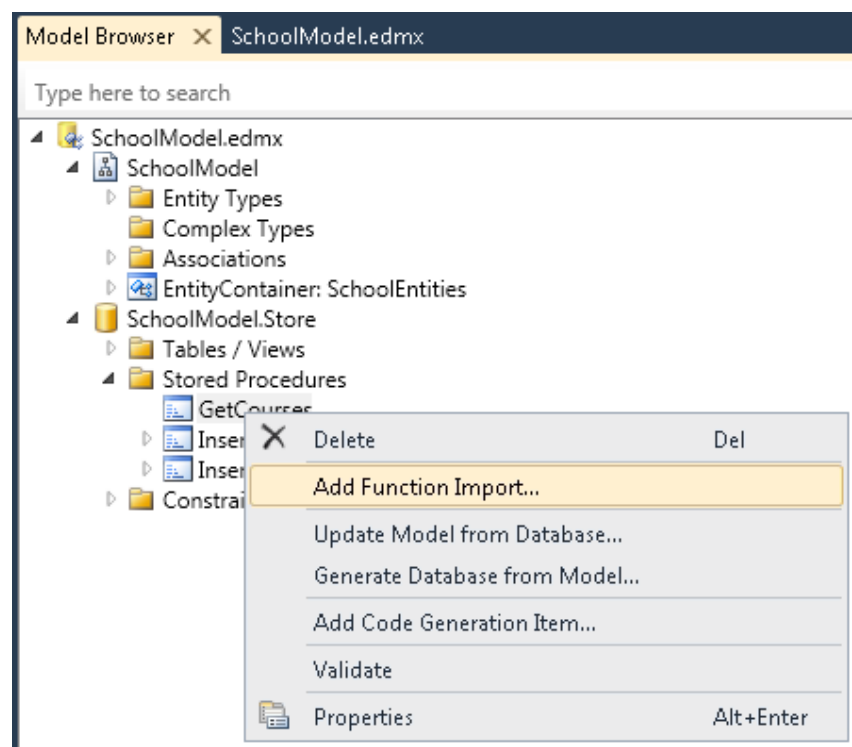
8. Ниже в списке выбора функции обновления укажите хранимую процедуру UpdateStudent и выберите FirstMidName из выпадающего списка напротив свойства FirstName.
9. И для функции удаления укажите процедуру DeletePerson.
10. Проверьте параметры в соответствии с рисунком.



11. В контекстном меню Instructor выберите **Stored Procedure Mapping (Сопоставление хранимых процедур)** и укажите соответствующие хранимые процедуры вставки, обновления и удаления для сопоставления с моделью согласно рисунка.

Mapping Details - Instructor		
Parameter / Column	Operator	Property
Functions		
Insert Using InsertInstructor		
Parameters		
@ LastName : nvarchar	←	LastName : String
@ FirstName : nvarchar	←	FirstMidName : String
@ HireDate : datetime	←	HireDate : DateTime
Result Column Bindings		
<Add Result Binding>		
Update Using UpdateInstructor		
Parameters		
@ PersonID : int	←	PersonID : Int32
@ LastName : nvarchar	←	LastName : String
@ FirstName : nvarchar	←	FirstMidName : String
@ HireDate : datetime	←	HireDate : DateTime
Result Column Bindings		
<Add Result Binding>		
Delete Using DeletePerson		
Parameters		
@ PersonID : int	←	PersonID : Int32

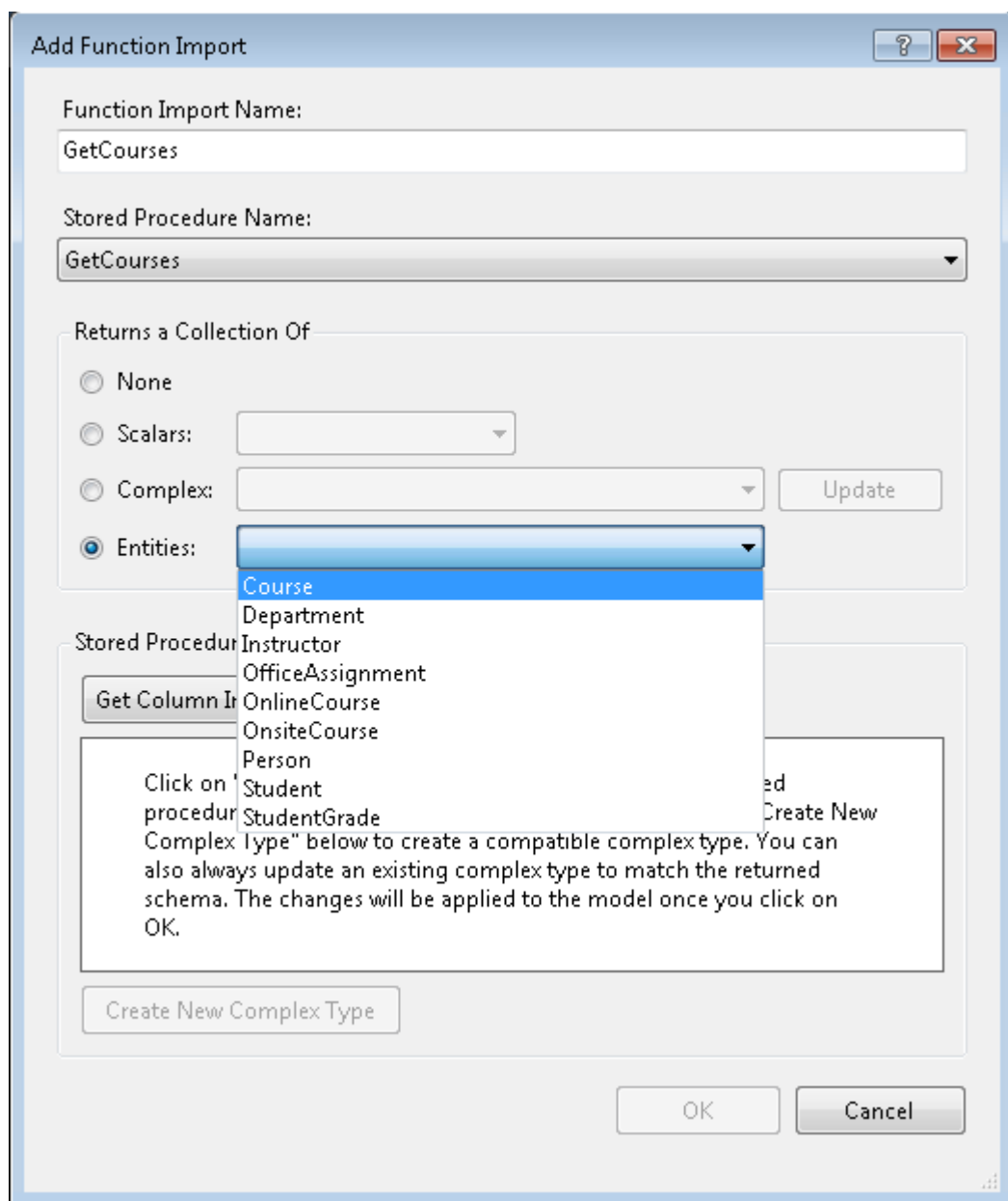
12. Откройте **Model Browser (Обозреватель моделей)**, раскройте последовательно узлы **SchoolModel.Store** и **Stored Procedures (Хранимые процедуры)**.
13. В контекстном меню процедуры **GetCourses** выберите команду **Add Function Import (Добавить импорт функций)**.



14. В окне добавления импорта **Add Function Import** для группы **Returns a Collection Of (Возвращает коллекцию)** выберите переключатель **Entities**

(Сущности), и затем выберите `Course` в качестве типа возвращаемой сущности. Нажмите **ОК**, сохраните и закройте `.edmx` файл.

15. Постройте приложение.



### Упражнение 3. Применение хранимых процедур

#### Применение процедур Insert, Update, и Delete

1. Откройте страницу `StudentsAdd.aspx` в обозревателе и добавьте нового студента, запустится хранимая процедура `InsertStudent` и запись добавится в таблицу `Student`.

## ADD NEW STUDENTS

First Name	<input type="text" value="New"/>
Last Name	<input type="text" value="Student"/>
Enrollment Date	<input type="text" value="09/10/2010"/>
<a href="#">Insert</a> <a href="#">Cancel</a>	

- Откройте страницу *Students.aspx* в обозревателе и проверьте, что новый студент добавлен в список.

## STUDENT LIST

	ID	Name	Enrollment Date	Number of Courses
<a href="#">Edit</a> <a href="#">Delete</a>	26	Rogers, Cody	9/1/2002	2
<a href="#">Edit</a> <a href="#">Delete</a>	28	White, Anthony	9/1/2001	2
<a href="#">Edit</a> <a href="#">Delete</a>	29	Griffin, Rachel	9/1/2004	1
<a href="#">Edit</a> <a href="#">Delete</a>	30	Shan, Alicia	9/1/2003	2
<a href="#">Edit</a> <a href="#">Delete</a>	33	Gao, Erica	1/30/2003	0
<a href="#">Edit</a> <a href="#">Delete</a>	35	Smith, John	1/1/2011	0
<a href="#">Edit</a> <a href="#">Delete</a>	38	Student, New	9/10/2010	0
<a href="#">1</a> <a href="#">2</a> <a href="#">3</a>				

- Измените имя для проверки функции обновления, а затем удалите студента для проверки функции удаления.

## STUDENT LIST

	ID	Name	Enrollment Date	Number of Courses
<a href="#">Edit</a> <a href="#">Delete</a>	26	Rogers, Cody	9/1/2002	2
<a href="#">Edit</a> <a href="#">Delete</a>	28	White, Anthony	9/1/2001	2
<a href="#">Edit</a> <a href="#">Delete</a>	29	Griffin, Rachel	9/1/2004	1
<a href="#">Edit</a> <a href="#">Delete</a>	30	Shan, Alicia	9/1/2003	2
<a href="#">Edit</a> <a href="#">Delete</a>	33	Gao, Erica	1/30/2003	0
<a href="#">Edit</a> <a href="#">Delete</a>	35	Smith, John	1/1/2011	0
<a href="#">Update</a> <a href="#">Cancel</a>	38	<input type="text" value="Student"/>	<input type="text" value="ToBeDeleted"/>	0
<a href="#">1</a> <a href="#">2</a> <a href="#">3</a>				

## Применение хранимой процедуры Select

Entity Framework не может автоматически запустить такую хранимую процедуру как `GetCourses`, и в этом случае нет возможности использовать элемент `EntityDataSource`. Запуск процедуры вы должны выполнить непосредственно в коде.

- Откройте файл *InstructorsCourses.aspx.cs*.
- Найдите метод `PopulateDropDownLists`, который применяет запрос LINQ-to-Entities для получения всех курсов `query to retrieve all course entities so that it can loop through the list and determine which ones an instructor is assigned to and which ones are unassigned:`

```
var allCourses = (from c in context.Courses
                  select c).ToList();
```

6. Замените этот код следующим:

```
var allCourses = context.GetCourses();
```

Теперь страница использует хранимую процедуру `GetCourses` для получения списка курсов.

7. Откройте данную страницу в обозревателе и проверьте правильность ее работы.

## Lab 8. Применение функциональности **Dynamic Data** для форматирования и валидации данных

В этой работе вы изучите возможности динамических данных (**Dynamic Data**) для обеспечения следующих преимуществ:

- поля автоматически форматируются при отображении согласно типу,
- поля автоматически проверяются на соответствие типа,
- вы можете добавить в модель данных дополнительные возможности по форматированию и проверке данных.

В этой работе вы внесете изменения в элементы управления для отображения и редактирования полей на странице *Students.aspx*, и добавите форматирование и проверку полей имени и даты объектов `Student`.

**STUDENT LIST**

	<u>Name</u>	<u>Enrollment Date</u>	Number of Courses
<a href="#">Edit</a> <a href="#">Delete</a>	Alexander, Carson	9/1/2055	3
<a href="#">Edit</a> <a href="#">Delete</a>	Alonso, Meredith	9/1/2002	1
<a href="#">Edit</a> <a href="#">Delete</a>	Anand, Arturo	9/1/2003	2
<a href="#">Edit</a> <a href="#">Delete</a>	Barzdukas, Gytis	9/1/2005	2
<a href="#">Edit</a> <a href="#">Delete</a>	Browning, Meredith	9/1/2000	2
<a href="#">Edit</a> <a href="#">Delete</a>	Bryant, Carson	9/1/2001	1
<a href="#">Edit</a> <a href="#">Delete</a>	Carlson, Robyn	9/1/2005	1
<a href="#">Edit</a> <a href="#">Delete</a>	Gao, Erica	1/30/2003	0
<a href="#">Edit</a> <a href="#">Delete</a>	Griffin, Rachel	9/1/2004	1
<a href="#">Edit</a> <a href="#">Delete</a>	Holt, Roger	9/1/2004	1

[1](#) [2](#) [3](#)

**FIND STUDENTS BY NAME**

Enter any part of the name

Name	Enrollment Date
Barzdukas, Gytis	9/1/2005

### Упражнение 1. Применение элементов *DynamicField* и *DynamicControl*

1. Откройте страницу *Students.aspx* и в элементе `StudentsGridView` замените элементы `TemplateField` с именами **Name** и **Enrollment Date** следующим кодом:

```
<asp:TemplateField HeaderText="Name" SortExpression="LastName">  
    <EditItemTemplate>
```



```

        <asp:DynamicControl ID="LastNameTextBox" runat="server"
DataField="LastName" Mode="Edit" />
        <asp:DynamicControl ID="FirstNameTextBox" runat="server"
DataField="FirstMidName" Mode="Edit" />
    </EditItemTemplate>
    <ItemTemplate>
        <asp:DynamicControl ID="LastNameLabel" runat="server"
DataField="LastName" Mode="ReadOnly" />,
        <asp:DynamicControl ID="FirstNameLabel" runat="server"
DataField="FirstMidName" Mode="ReadOnly" />
    </ItemTemplate>
</asp:TemplateField>
    <asp:DynamicField DataField="EnrollmentDate" HeaderText="Enrollment
Date" SortExpression="EnrollmentDate" />

```

Этот код использует элементы `DynamicControl` в элементах управления `TextBox` и `Label` в шаблоне поля отображения имени студента, а также применяет элемент `DynamicField` для поля даты (enrollment date).

2. Добавьте элемент `ValidationSummary` после компонента `StudentsGridView`:

```

    <asp:ValidationSummary ID="StudentsValidationSummary"
runat="server" ShowSummary="true"
    DisplayMode="BulletList" Style="color: Red" />

```

3. В элементе `SearchGridView` замените код столбцов **Name** and **Enrollment Date** как это было сделано в элементе `StudentsGridView`. Элементы `Columns` компонента `SearchGridView` должны выглядеть следующим образом:

```

    <asp:TemplateField HeaderText="Name" SortExpression="LastName">
    <ItemTemplate>
        <asp:DynamicControl ID="LastNameLabel" runat="server"
DataField="LastName" Mode="ReadOnly" />,
        <asp:DynamicControl ID="FirstNameLabel" runat="server"
DataField="FirstMidName" Mode="ReadOnly" />
    </ItemTemplate>
    </asp:TemplateField>
    <asp:DynamicField DataField="EnrollmentDate" HeaderText="Enrollment
Date" SortExpression="EnrollmentDate" />

```

4. Откройте файл `Students.aspx.cs` и добавьте выражение `using`:

```
using ContosoUniversity.DAL;
```

5. Добавьте обработчик события `Init` этой страницы:

```

protected void Page_Init(object sender, EventArgs e)
{
    StudentsGridView.EnableDynamicData(typeof(Student));
    SearchGridView.EnableDynamicData(typeof(Student));
}

```

Это код определяет, как `Dynamic Data` будут обеспечивать форматирование и проверку полей элементов объекта `Student`.

6. Откройте страницу в обозревателе.

## STUDENT LIST

	Name	Enrollment Date	Number of Courses
<a href="#">Edit</a> <a href="#">Delete</a>	Alexander, Carson	9/1/2055 12:00:00 AM	3
<a href="#">Edit</a> <a href="#">Delete</a>	Alonso, Meredith	9/1/2002 12:00:00 AM	1
<a href="#">Edit</a> <a href="#">Delete</a>	Anand, Arturo	9/1/2003 12:00:00 AM	2
<a href="#">Edit</a> <a href="#">Delete</a>	Barzdukas, Gytis	9/1/2005 12:00:00 AM	2
<a href="#">Edit</a> <a href="#">Delete</a>	Browning, Meredith	9/1/2000 12:00:00 AM	2
<a href="#">Edit</a> <a href="#">Delete</a>	Bryant, Carson	9/1/2001 12:00:00 AM	1
<a href="#">Edit</a> <a href="#">Delete</a>	Carlson, Robyn	9/1/2005 12:00:00 AM	1
<a href="#">Edit</a> <a href="#">Delete</a>	Gao, Erica	1/30/2003 12:00:00 AM	0
<a href="#">Edit</a> <a href="#">Delete</a>	Griffin, Rachel	9/1/2004 12:00:00 AM	1
<a href="#">Edit</a> <a href="#">Delete</a>	Holt, Roger	9/1/2004 12:00:00 AM	1
1 2 3			

## FIND STUDENTS BY NAME

Enter any part of the name

Name	Enrollment Date
Barzdukas, Gytis	9/1/2005 12:00:00 AM
Justice, Peggy	9/1/2001 12:00:00 AM
Li, Yan	9/1/2002 12:00:00 AM
Norman, Laura	9/1/2003 12:00:00 AM
Olivotto, Nino	9/1/2005 12:00:00 AM

7. В столбце **Enrollment Date** время отображается в длинном формате в соответствии со свойством `DateTime`.

Обратите внимание как `Dynamic Data` автоматически обеспечивает валидацию данных.

8. Например, кликните **Edit (Правка)** для первого студента, очистите поле даты и кликните **Update (Обновить)**, проверьте, что `Dynamic Data` автоматически пометил, что требуется данные в это поле, так как модель не допускает, чтобы это поле было пустым. На странице должно отобразиться сообщение об этом в элементе `ValidationSummary`:

## STUDENT LIST

	Name	Enrollment Date
<a href="#">Update</a> <a href="#">Cancel</a>	Alexander , Carson	*
<a href="#">Edit</a> <a href="#">Delete</a>	Alonso, Meredith	9/1/2002
<a href="#">Edit</a> <a href="#">Delete</a>	Anand, Arturo	9/1/2003
<a href="#">Edit</a> <a href="#">Delete</a>	Barzdukas, Gytis	9/1/2005
<a href="#">Edit</a> <a href="#">Delete</a>	Browning, Meredith	9/1/2000
<a href="#">Edit</a> <a href="#">Delete</a>	Bryant, Carson	9/1/2001
<a href="#">Edit</a> <a href="#">Delete</a>	Carlson, Robyn	9/1/2005
<a href="#">Edit</a> <a href="#">Delete</a>	Gao, Erica	1/30/2003
<a href="#">Edit</a> <a href="#">Delete</a>	Griffin, Rachel	9/1/2004
<a href="#">Edit</a> <a href="#">Delete</a>	Holt, Roger	9/1/2004
1 2 3		

- The `EnrollmentDate` field is required.

9. Проверьте наличие всплывающей подсказки при наведении мыши на метку ошибки поля даты:

Enrollment Date	Number of Courses
<input type="text"/>	* 3
9/1/2002	1
9/1/2003	2

The EnrollmentDate field is required.

10. Введите неправильную дату, например 1/32/2010, проверьте, что Dynamic Data также среагировал на эту ошибку:

#### STUDENT LIST

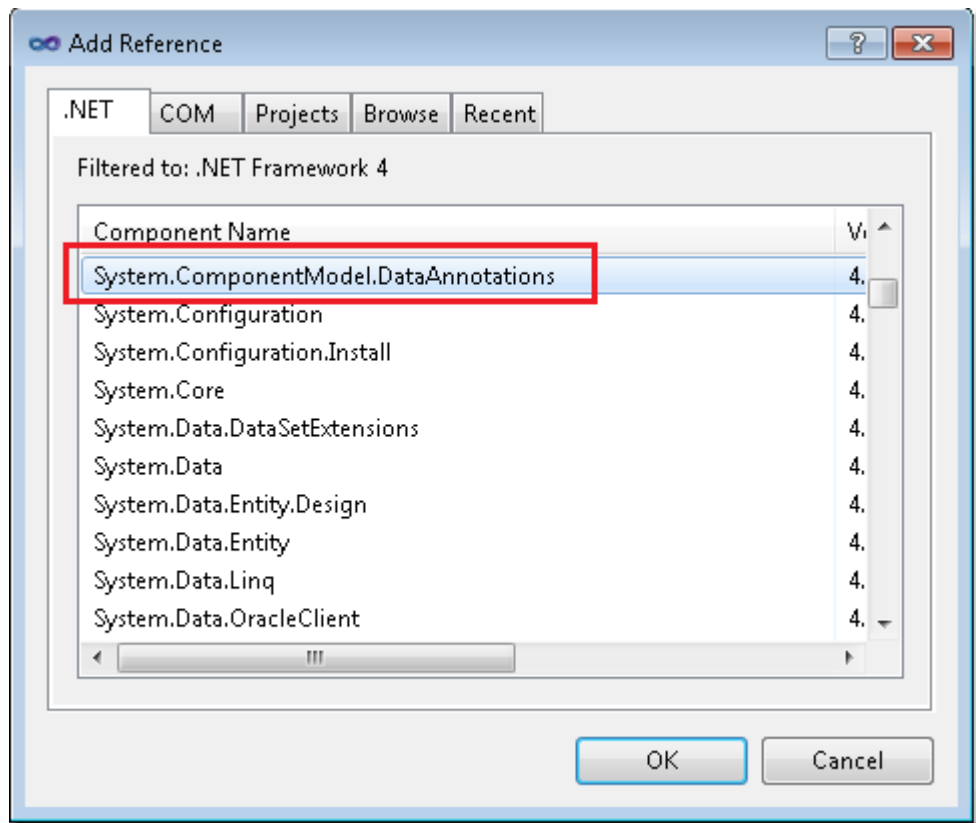
	Name		Enrollment Date
<a href="#">Update</a> <a href="#">Cancel</a>	<input type="text" value="Alexander"/>	, <input type="text" value="Carson"/>	1/32/2010
<a href="#">Edit</a> <a href="#">Delete</a>	Alonso, Meredith		9/1/2002 12:00:00 AM
<a href="#">Edit</a> <a href="#">Delete</a>	Anand, Arturo		9/1/2003 12:00:00 AM
<a href="#">Edit</a> <a href="#">Delete</a>	Barzdukas, Gytis		9/1/2005 12:00:00 AM
<a href="#">Edit</a> <a href="#">Delete</a>	Browning, Meredith		9/1/2000 12:00:00 AM
<a href="#">Edit</a> <a href="#">Delete</a>	Bryant, Carson		9/1/2001 12:00:00 AM
<a href="#">Edit</a> <a href="#">Delete</a>	Carlson, Robyn		9/1/2005 12:00:00 AM
<a href="#">Edit</a> <a href="#">Delete</a>	Gao, Erica		1/30/2003 12:00:00 AM
<a href="#">Edit</a> <a href="#">Delete</a>	Griffin, Rachel		9/1/2004 12:00:00 AM
<a href="#">Edit</a> <a href="#">Delete</a>	Holt, Roger		9/1/2004 12:00:00 AM
1 2 3			

▪ The value is not valid.

### Упражнение 2. Добавление правил валидации и форматирования

В этом упражнении вы реализуете дополнительные возможности по форматированию и проверки данных к тем которые предоставляются механизмом Dynamic Data.

- В контекстном **ContosoUniversity** выберите команду **Add Reference (Добавить ссылку)** и в окне добавления ссылки на вкладке **.NET** найдите компонент `System.ComponentModel.DataAnnotations` и нажмите **ОК**.



2. В папку *DAL* добавьте новый файл – класс *Student.cs*
3. Добавьте класс метаданных с указанием требуемых свойств класса и атрибутов валидации:

```
using System;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;

namespace ContosoUniversity.DAL
{
    [MetadataType(typeof(StudentMetadata))]
    public partial class Student
    {
    }

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

        [StringLength(25, ErrorMessage = "First name must be 25 characters or
less in length.")]
        [Required(ErrorMessage = "First name is required.")]
        public String FirstMidName { get; set; }

        [StringLength(25, ErrorMessage = "Last name must be 25 characters or
less in length.")]
        [Required(ErrorMessage = "Last name is required.")]
        public String LastName { get; set; }
    }
}
```

В этом коде создается частичный (partial) класс для сущности `Student`. Атрибут `MetadataType`, добавленный к этому частичному классу определяет класс как с возможностью применять специфические метаданные. Класс метаданных может иметь любое имя, но лучше применять суффикс "Metadata" как часть имени.

Атрибуты применяются для свойств в классе метаданных, определяющих форматирование, валидацию, правила и сообщения об ошибках

Атрибуты обеспечивают следующие результаты:

- `EnrollmentDate` будет отображаться как дата без времени,
- оба поля имени должны быть не больше 25 символов и сообщение об ошибке,
- оба поля имени требуют обязательного ввода и отображают сообщение об соответствующей ошибке.

4. Откройте страницу `Students.aspx` в обозревателе:

STUDENT LIST			
	Name	Enrollment Date	Number of Courses
<a href="#">Edit</a> <a href="#">Delete</a>	Alexander, Carson	9/1/2055	3
<a href="#">Edit</a> <a href="#">Delete</a>	Alonso, Meredith	9/1/2002	1
<a href="#">Edit</a> <a href="#">Delete</a>	Anand, Arturo	9/1/2003	2

5. Выполните правку строки: удалите содержимое поля имени и обновите строку. Проверьте, что отобразились предупреждения об ошибке.

STUDENT LIST						
	Name					
<a href="#">Update</a> <a href="#">Cancel</a>	<input type="text"/>	*	<input type="text"/>			
<a href="#">Edit</a> <a href="#">Delete</a>	Alonso, Meredith					
<a href="#">Edit</a> <a href="#">Delete</a>	Anand, Arturo					
<a href="#">Edit</a> <a href="#">Delete</a>	Barzdukas, Gytis					
<a href="#">Edit</a> <a href="#">Delete</a>	Browning, Meredith					
<a href="#">Edit</a> <a href="#">Delete</a>	Bryant, Carson					
<a href="#">Edit</a> <a href="#">Delete</a>	Carlson, Robyn					
<a href="#">Edit</a> <a href="#">Delete</a>	Gao, Erica					
<a href="#">Edit</a> <a href="#">Delete</a>	Griffin, Rachel					
<a href="#">Edit</a> <a href="#">Delete</a>	Holt, Roger					
1 2 3						
<ul style="list-style-type: none"> <li>▪ Last name is required.</li> <li>▪ First name is required.</li> </ul>						

6. Введите в поле имени более 25 символов, проверьте появление сообщения об ошибке.

## STUDENT LIST

	<a href="#">Name</a>
<a href="#">Update</a> <a href="#">Cancel</a>	Alexander more than 25 c *, Carson more than 25 char *
<a href="#">Edit</a> <a href="#">Delete</a>	Alonso, Meredith
<a href="#">Edit</a> <a href="#">Delete</a>	Anand, Arturo
<a href="#">Edit</a> <a href="#">Delete</a>	Barzdukas, Gytis
<a href="#">Edit</a> <a href="#">Delete</a>	Browning, Meredith
<a href="#">Edit</a> <a href="#">Delete</a>	Bryant, Carson
<a href="#">Edit</a> <a href="#">Delete</a>	Carlson, Robyn
<a href="#">Edit</a> <a href="#">Delete</a>	Gao, Erica
<a href="#">Edit</a> <a href="#">Delete</a>	Griffin, Rachel
<a href="#">Edit</a> <a href="#">Delete</a>	Holt, Roger
1 <a href="#">2</a> <a href="#">3</a>	

- Last name must be 25 characters or less in length.
- First name must be 25 characters or less in length.