**Лабораторная работа № 2**

**Знакомство с Entity Framework и Bootstrap**

**Цель работы**

Освоить механизмы подключения источника данных к веб-приложению с использованием Entity Framework и отображения модели c использованием фреймворка Bootstrap.

**После выполнения ЛР студенты должны уметь**

1. Подключать к проекту источник данных с использованием Entity Framework;
2. Настраивать переходы между представлениями;
3. Оформлять представления с помощью фреймворка Bootstrap.

**Ход работы**

1. Запустить Microsoft SQL Server Management Studio и создать таблицы для дальнейшей работы.
2. Открыть предыдущий или создать новый проект ASP .NET MVC в среде Visual Studio.
3. Установить Entity Framework.
4. Добавить в проект модель данных, создав подключение к базе данных.
5. Создать метод, в котором осуществляется формирование списка объектов из записей одной из таблиц.
6. Создать представление для отображения списка полученных объектов на веб-странице
7. Создать метод с параметром, где осуществляется получение одного объекта по идентификатору.
8. Создать представление для отображения основной информации по объекту.

**Порядок выполнения работы**

* 1. **Создание таблиц в базе данных**

Ранее были рассмотрены примеры работы со списком строковых объектов, в рамках же данной лабораторной работы будет продемонстрирован пример получения и вывода объектов собственных классов.

***ВНИМАНИЕ!*** *В качестве источника данных настоятельно рекомендуется использовать базу данных, созданную в рамках выполнения ИДЗ по дисциплине «Технологии проектирования баз данных», однако допускается выполнение и с предметной областью, описанной ниже*.

В качестве тестовой предметной области возьмем проведение опросов. Нам понадобятся следующие сущности: [Участник опроса]–[Ответ участника на вопрос]–[Вопрос]. Создадим для них таблицы в базе данных.

Для участника опроса (Person) будем хранить:

* его уникальный идентификатор,
* ФИО,
* возраст,
* пол,
* флаг наличия места работы.

Характеристиками вопроса (Question) будут:

* его номер,
* текстовое содержание.

В ответе участника на вопрос (Answer) сохраним:

* идентификатор участника (внешний ключ на таблицу Person),
* номер вопроса (внешний ключ на таблицу Question),
* дату и время ответа на вопрос,
* текст ответа.

На рисунке 1 приведен скрипт для создания вышеописанных таблиц и связей между ними.

В случае если Вы будете использовать свою предметную область, то необходимо выбрать такой набор таблиц в вашей БД, чтобы он включал две основные таблицы и связывающую их таблицу с несколькими собственными атрибутами. Возможные варианты:

* [Студент] – [Студент в группе] – [Группа]
* [Пациент] – [Прием пациента у врача] – [Врач]
* [Покупатель] – [Заказ товара покупателем] – [Товар]
* [Преподаватель] – [Опыт преподавателя по дисциплине] – [Дисциплина]
* [Музыкант] – [Роль музыканта в группе] – [Группа]
* [Ингредиент] – [Ингредиент в блюде] – [Блюдо]
* [Удар] – [Свойства удара у персонажа] – [Персонаж]
* прочие аналогичные.

Важно, чтобы в промежуточной таблице был хотя бы один собственный атрибут, например:

* студент в группе – способ оплаты
* прием пациента у врача – время приема
* заказ товара покупателем – количество
* и т.д.

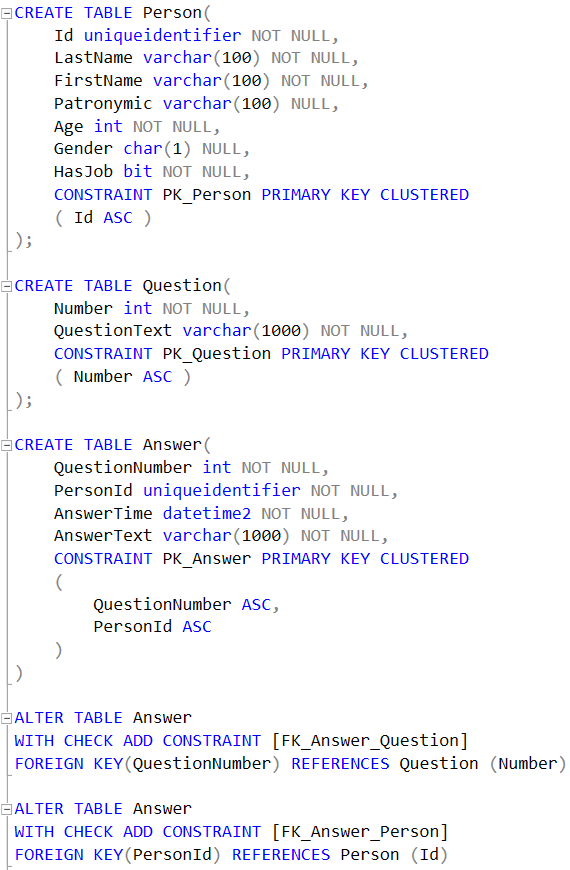


Рисунок 1 – Скрипт для создания таблиц

* 1. **Добавление модели в проект веб-приложения**

Откройте предыдущий или создайте новый проект ASP .NET MVC в среде Visual Studio. Для того, чтобы получить возможность работать с данными из базы, необходимо добавить в проект пакет Entity Framework.

ORM ADO.NET Entity Framework – фреймворк, который используется для возможности подключения данных с сервера БД к среде Visual Studio. Данная технология позволяет оперировать данными из базы, производя не только чтение, но и возможность вносить необходимые изменения в базу данных при необходимости непосредственно из кода программы. При разработке с использованием ORM Entity Framework существует три основных подхода (см. ссылку №1 в списке источников):

1. Code First (сначала код) – подход, предполагающий создание классов внутри проекта в Visual Studio, которые впоследствии будут преобразованы и импортированы в таблицы базы данных.
2. Model First (сначала модель) – подход, предполагающий создание графической модели сущностей в проекте Visual Studio, по которым будут сгенерированы классы, а затем сформированы таблицы базы данных.
3. Database First (сначала база данных) – подход, предполагающий автоматическую генерацию сущностей и классов в проекте Visual Studio на основе таблиц из ранее созданной базы данных. Данный подход и будет использоваться в рамках выполнения лабораторных работ.

Для того, чтобы добавить в проект пакет Entity Framework, необходимо нажать правой кнопкой мыши по имени проекта, затем выбрать пункт «Управление пакетами NuGet…» (рисунок 2).

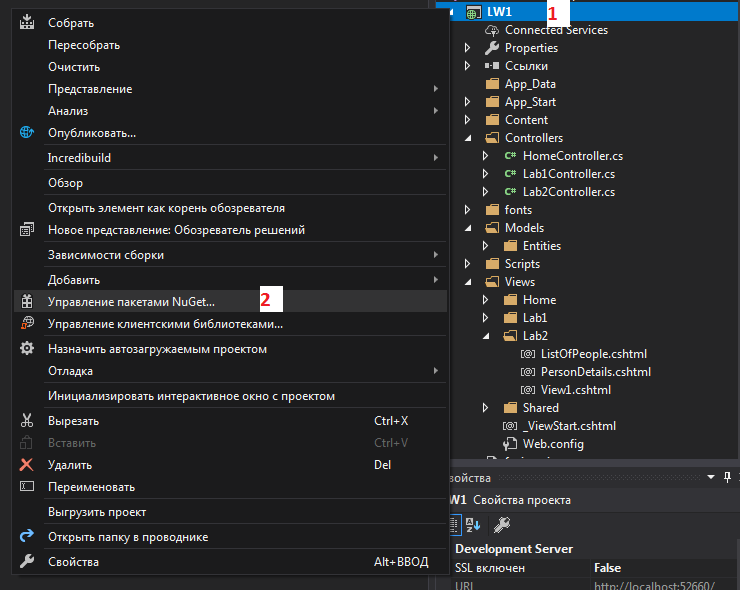


Рисунок 2 – Порядок открытия NuGet

Далее откроется диспетчер пакетов NuGet. Во вкладке «Обзор» в графе поиска необходимо начать вводить название необходимого пакета, затем выбрать подходящий и справа нажать «Установить» (рисунок 3).

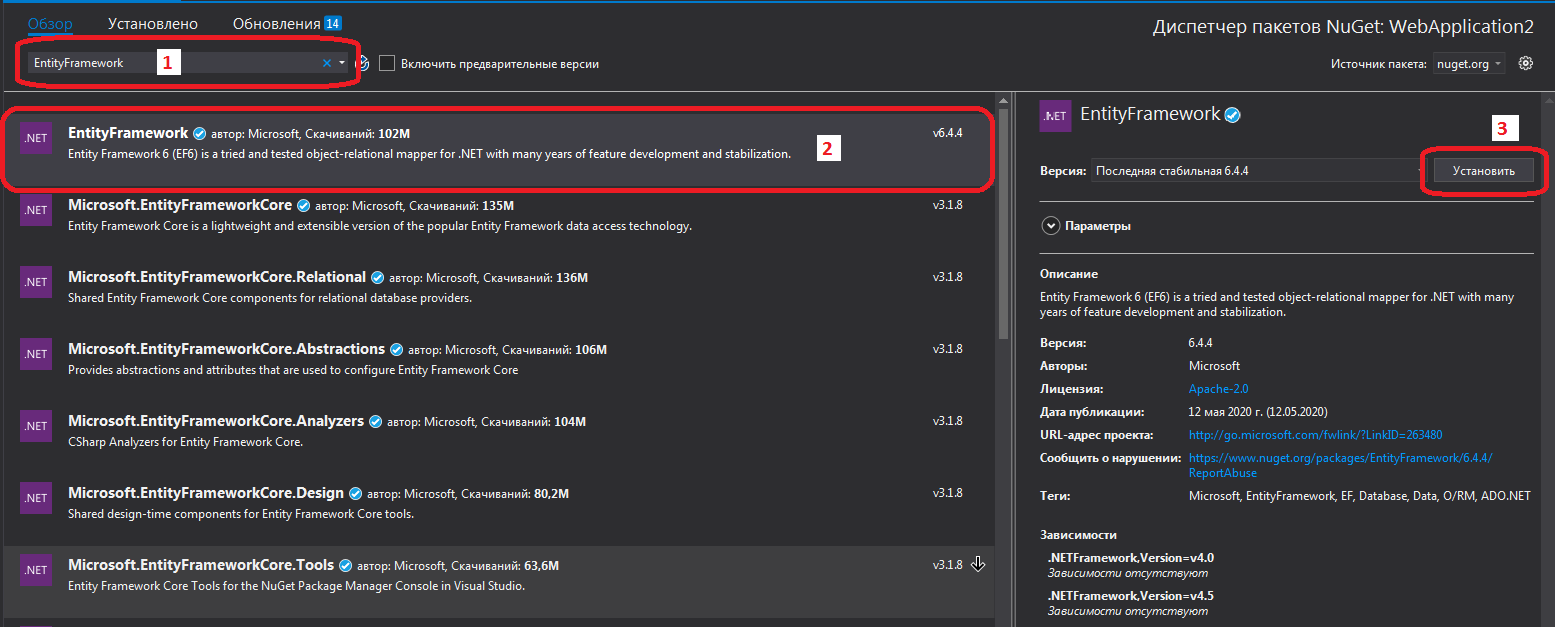


Рисунок 3 – Порядок установки Entity Framework

Далее требуется подтвердить установку и нажать «Ок» (рисунок 4).

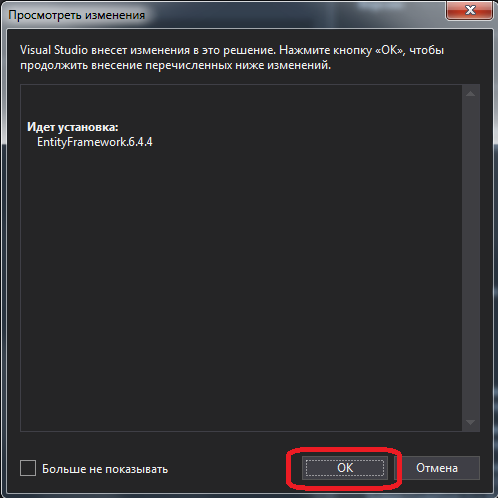


Рисунок 4 – Подтверждение установки

А затем «Я принимаю» (рисунок 5).

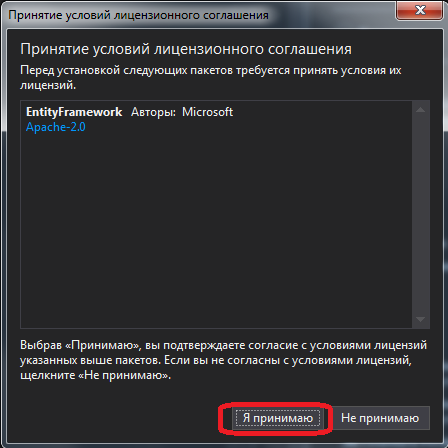


Рисунок 5 – Принятие лицензии

Для добавления модели (или других файлов, например, классов, скриптов и пр.) в проект необходимо нажать правой кнопкой мыши по папке и проделать действия, как продемонстрировано на рисунке 6.

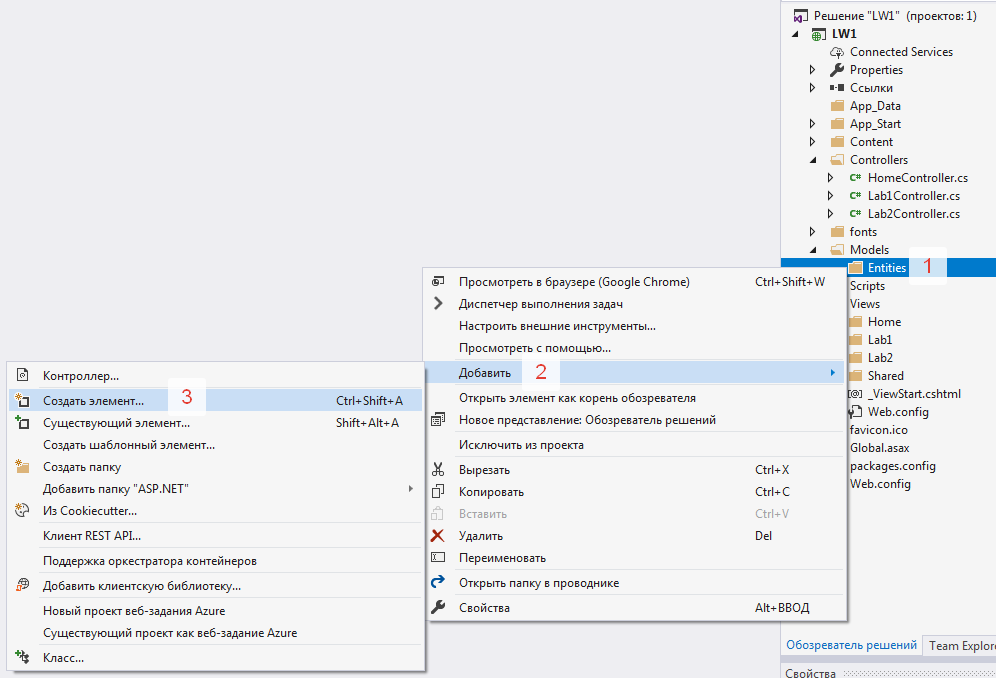


Рисунок 6 – Добавление новых элементов в проект

Поскольку модель работает с данными, то и в списке элементов ее следует искать среди группы Data (рисунок 7). Выберите пункт «Модель ADO.NET EDM», введите имя будущей модели и нажмите «Далее».

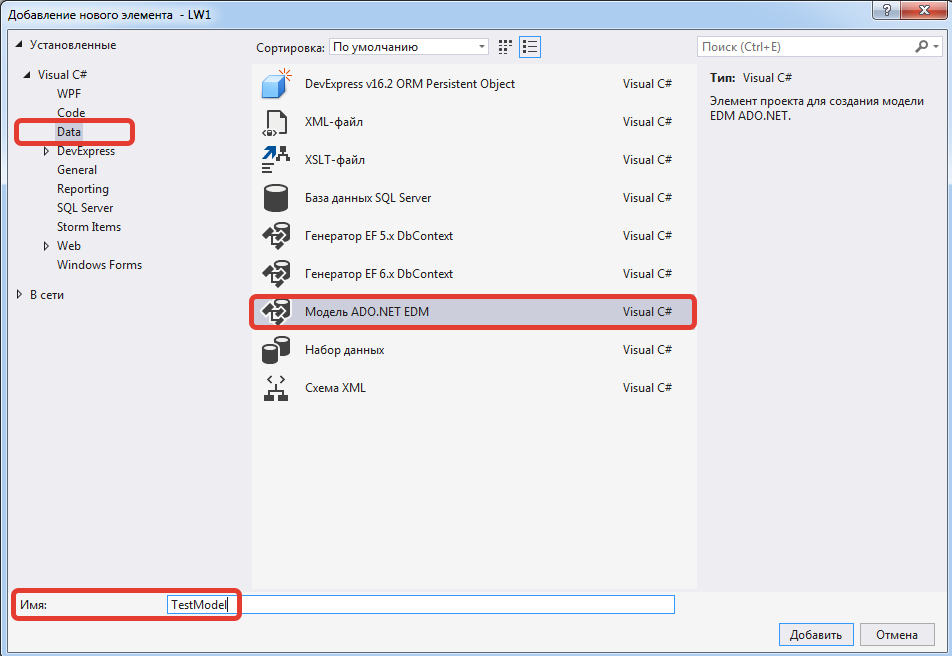


Рисунок 7 – Добавление модели в проект

Это приведет к запуску мастера создания модели сущностных данных. На первом шаге (рисунок 8) мастер предоставляет выбор: генерировать EDM из существующей базы данных, либо определить пустую модель (для разработки в стиле «сначала модель»).

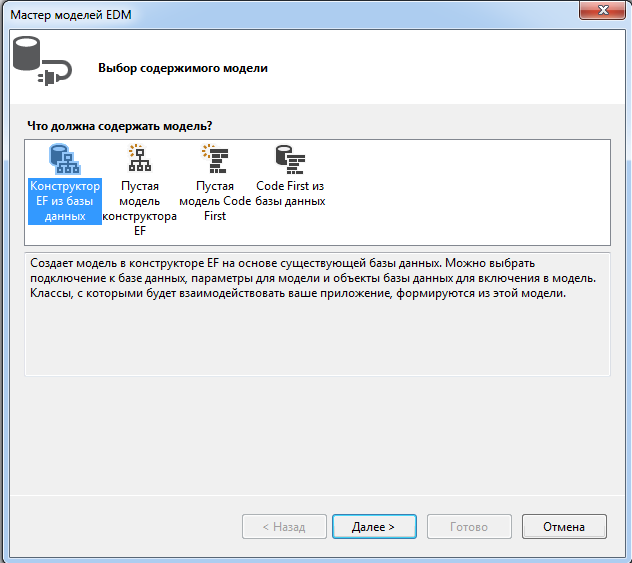


Рисунок 8 – Мастер создания модели, выбор содержимого модели

Выберите опцию «Конструктор EF из базы данных» (Generate from database) и нажмите на кнопку «Далее» (как было сказано ранее, при разработке будет использоваться поход Database First). Откроется окно следующего шага по созданию модели, на котором нужно установить подключение к базе данных (рисунок 9).

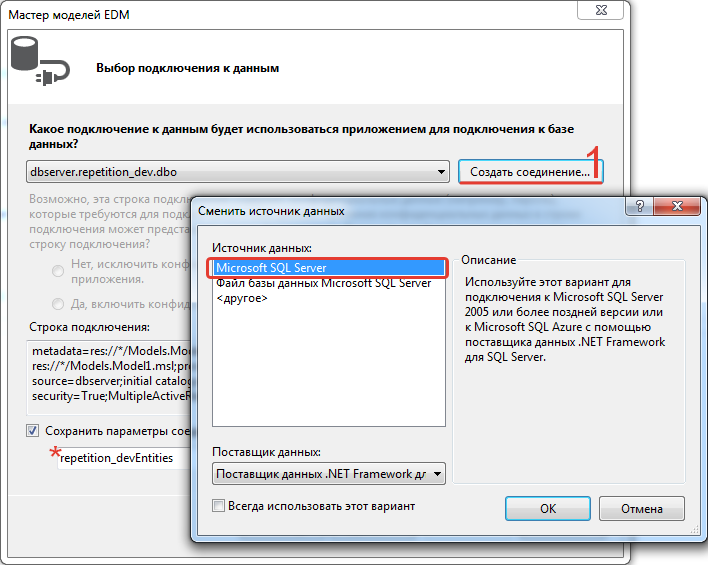


Рисунок 9 – Выбор подключения к данным

Если в проекте уже были подключения к БД, то они отобразятся в списке. В противном случае вам предстоит создать новое подключение, и для этого необходимо нажать на кнопку «Создать соединение». В открывшемся окошке в качестве источника данных выберите пункт «Microsoft SQL Server». Поставщик данных – .NET Framework. После нажатия на кнопку «Продолжить» перед вами появится окно со свойствами создаваемого подключения (рисунок 10). Здесь нужно указать имя сервера и базу, к которой создается подключение.

***ВНИМАНИЕ!*** *Если вы используете базу данных, размещенную на сервере ТПУ, то пропишите в поле «Имя сервера» полное имя сервера, выберите пункт «Проверка подлинности* *SQL Server», введите логин и пароль. Если же на вашем компьютере установлен SQL Server, то вы можете подключиться к своему локальному серверу – он должен отобразиться в списке имен серверов, если раскрыть соответствующий выпадающий список (либо в качестве имени написать localhost). В этом случае можно указывать либо вариант «Проверка подлинности Windows», либо вариант «Проверка подлинности SQL Server» с указанием логина и пароля пользователя, существующего в вашем экземпляре SQL Server.*

Пример строки подключения с указанием логина и пароля, а также описание других параметров подключения можно увидеть на примере по ссылке №2 в списке источников.

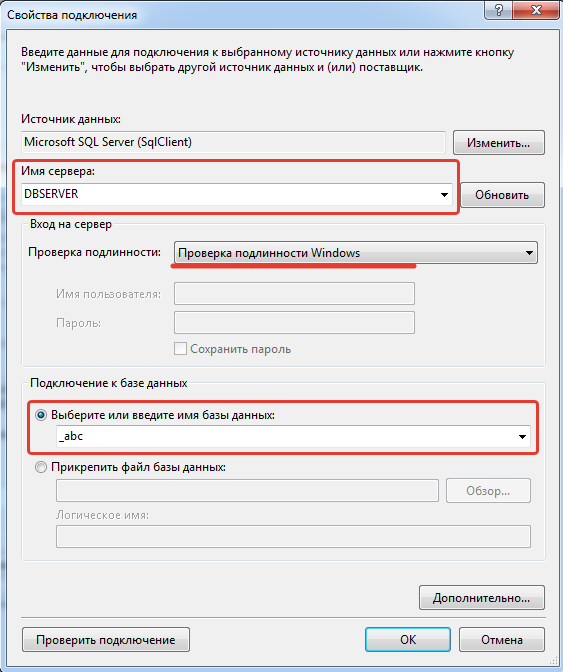


Рисунок 10 – Настройка подключения

Когда указано имя сервера, в пункте «Подключение к базе данных» вы можете или самостоятельно ввести имя БД или найти ее в заполнившемся списке баз данных указанного ранее сервера. Указав сервер и базу, нажмите «Проверить подключение». Если проверка прошла успешно, нажимайте кнопку «ОК». Следующий шаг – выбор объектов из БД, которые будут включены в модель (рисунок 11). Объектами могут быть таблицы, представления, хранимые процедуры и функции.

***ВНИМАНИЕ!*** *В случае если у вас имена таблиц заданы на английском языке и имеют форму множественного числа, то настоятельно рекомендуется поставить флажок «Формировать имена объектов во множественном или единственном числе». В этом случае значительно повысится читаемость кода, и будет очень хорошо видно, где идёт работа с одним объектом, а где – с коллекцией. Удобство данного способа Вы можете оценить в примерах кода ниже. Если же имена таблиц заданы на русском языке, то рекомендуется вручную переименовать сущности в форму единственного числа (дважды одиночный щелчок ЛКМ по заголовку сущности, либо ПКМ – переименовать). Читаемость кода в этом случае будет не настолько высокой, как в случае с английским языком, но тем не менее намного выше, чем использование переменных с именами в форме множественного числа.*

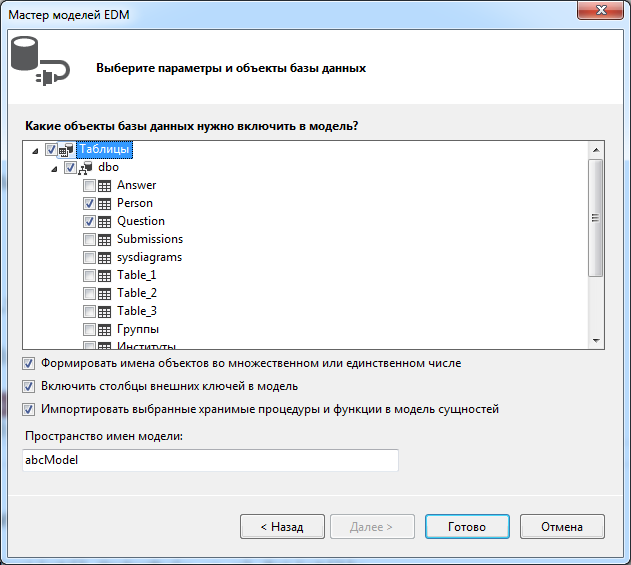


Рисунок 11 – Выбор подключаемых объектов из БД

На рисунке 11 видно, что для данного проекта были выбраны таблицы, и было указано имя модели. Когда выберете необходимые вам объекты, нажмите «Готово». Возможно, перед вами появится предупреждение системы безопасности (рисунок 12). Без лишней паники нажмите кнопку «ОК».

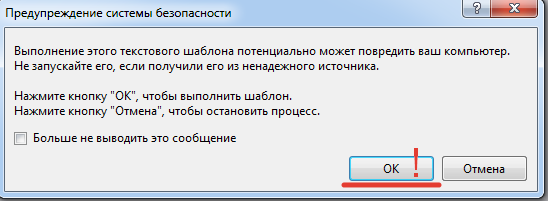


Рисунок 12 – Предупреждение системы безопасности

После создания модели перед вами откроется диаграмма загруженных объектов и браузер моделей, где в дереве указаны все подгруженные объекты. Как вы могли заметить, на рисунке 11 были выбраны таблицы Person, Question, но не Answer. Если вы забыли добавить какой-то объект, или новый объект появился позже в процессе разработки, вы всегда можете обновить модель. Для этого откройте диаграмму модели, нажмите правой кнопкой мыши в пустой области диаграммы и в появившемся меню выберите пункт «Обновить модель из базы данных» (рисунок 13).

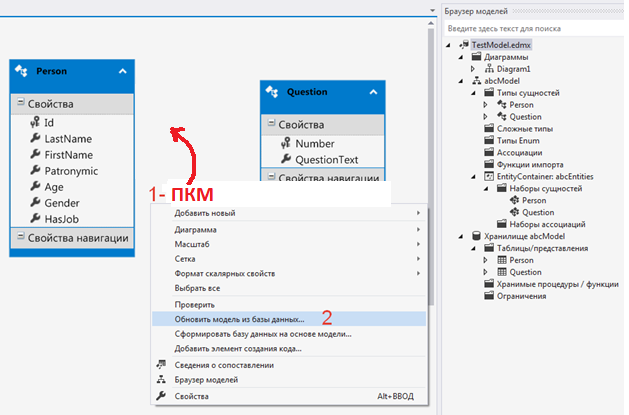


Рисунок 13 – Диаграмма модели и действия для обновления модели

После этого откроется уже знакомое окно (рисунок 14), где на вкладке «Добавить» вы сможете указать, какие объекты необходимо добавить. Если в какой-то уже ранее добавленной таблице были добавлены и/или удалены атрибуты, то их можно добавить, на вкладке «Обновить» выбрав группу объектов для обновления.

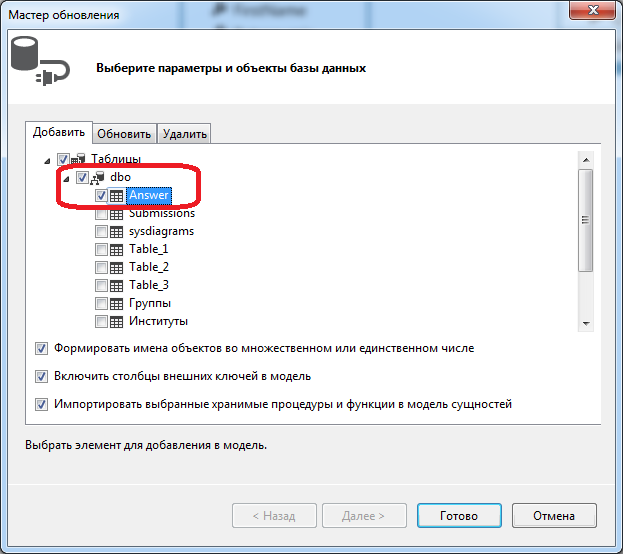


Рисунок 14 – Добавление новых объектов в модель

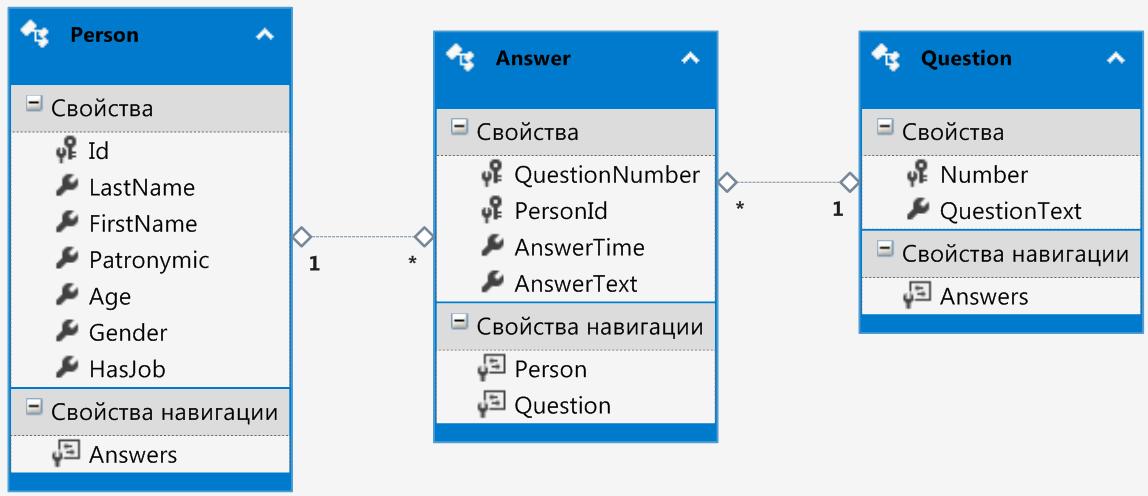


Рисунок 15 – Диаграмма обновленной модели

Когда модель создана, на основе таких объектов, как таблица и представление, создаются классы (рисунок 16). В классе создаются свойства, соответствующие атрибутам таблицы, а также свойства навигации, которые зависят от внешних связей, созданных для таблицы (рисунки 15 и 16). Однако можно заметить, что не все типы данных атрибутов переносятся в точности в том же виде, поскольку некоторые типы в языках T-SQL и C# различаются (см. ссылку №3 в списке источников). На рисунке 17 указано, каких типов коснулись изменения.

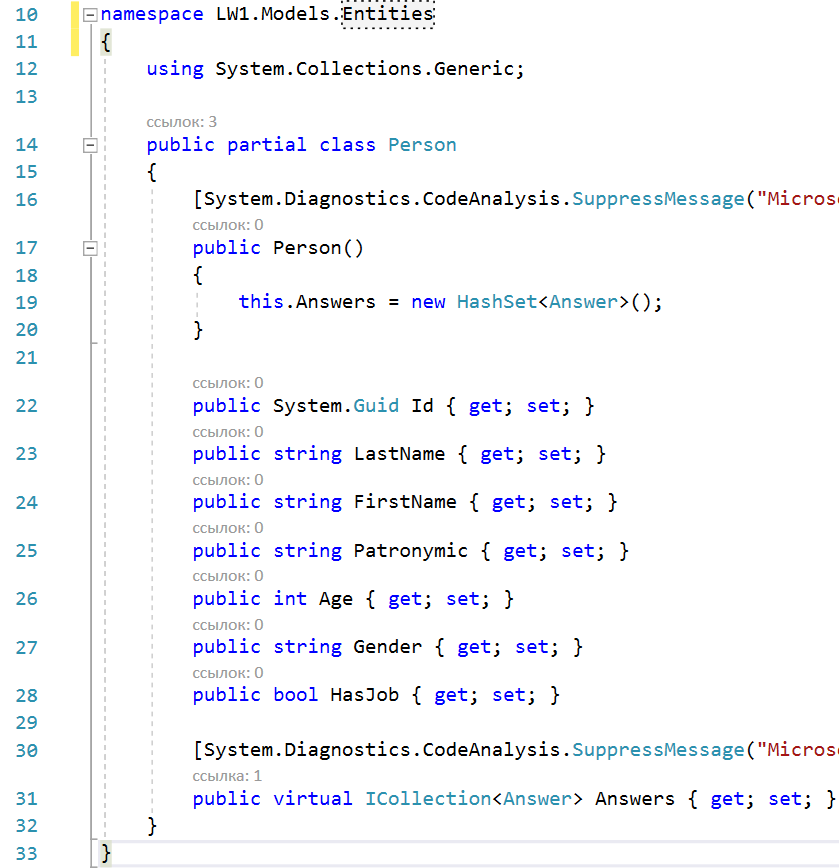


Рисунок 16 – Класс, созданный по таблице БД

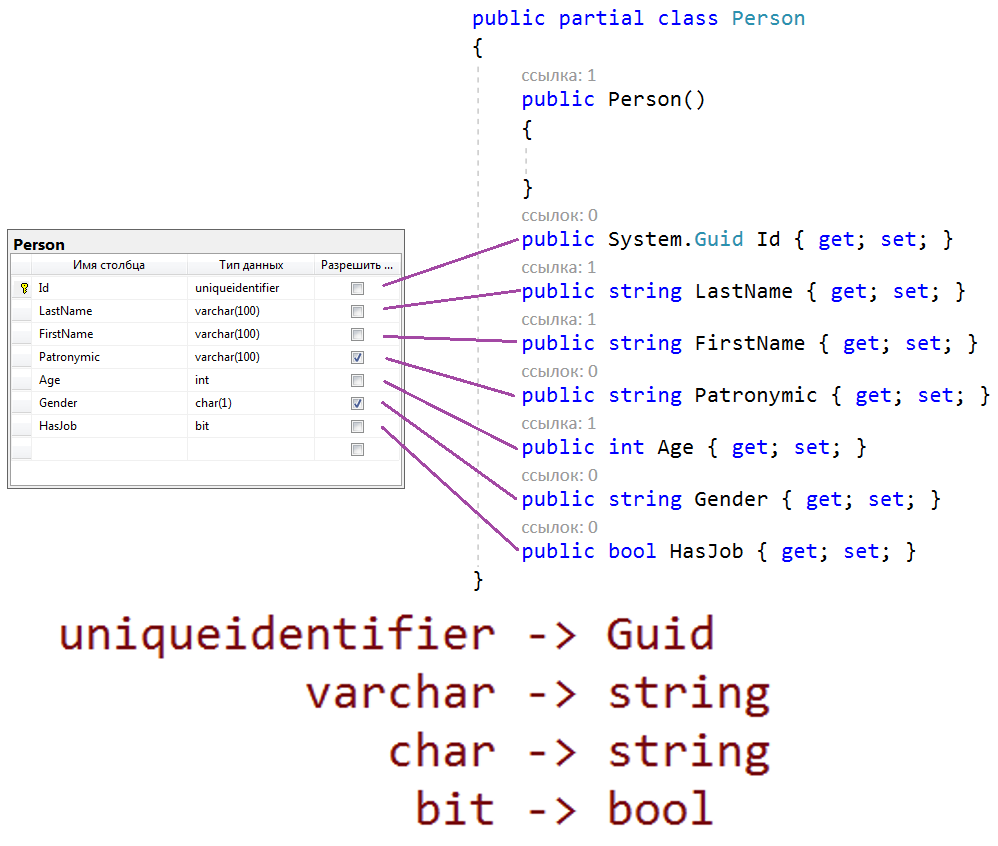


Рисунок 17 – Сопоставление типов атрибутов таблицы и свойств класса

* 1. **Создание метода для получения списка**

Теперь, когда добавлены объекты для работы, можно приступить к созданию методов их обработки. Начнем с уже знакомого действия – вывода списка объектов. Для удобства создайте новый контроллер, а также метод без параметров, который и будет возвращать ваш список.

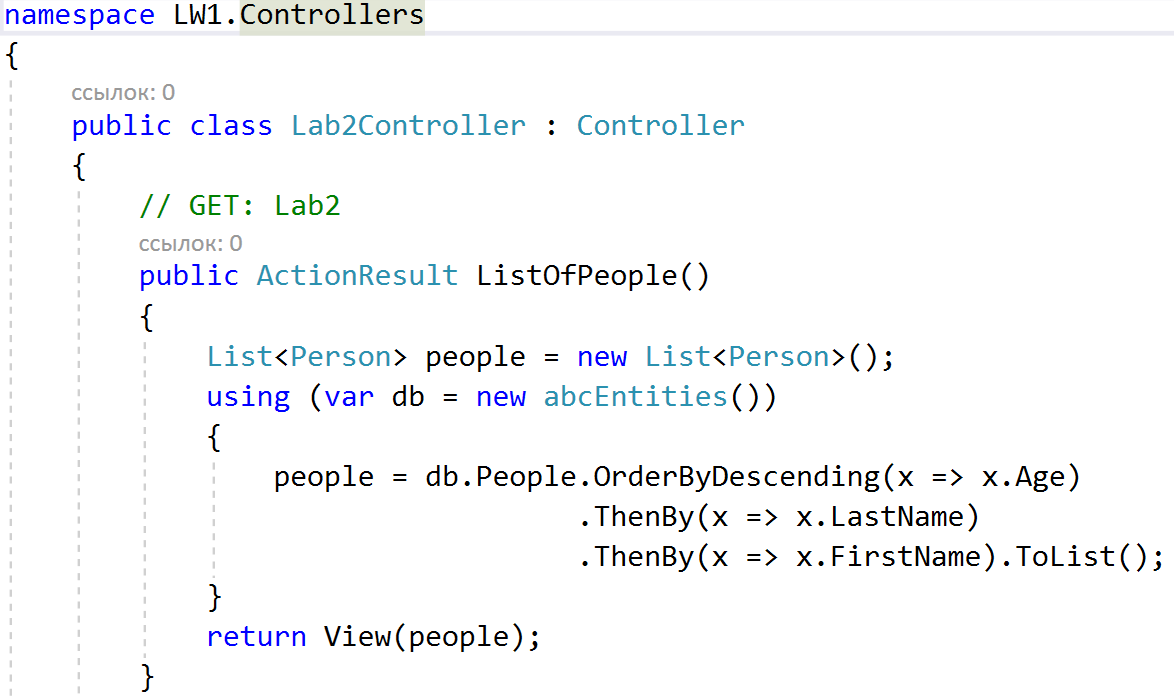


Рисунок 18 – Метод для получения списка

Как и в предыдущей лабораторной работе, здесь сначала инициализируется список, но на этот раз тип элемента списка – класс Person, а не string. Person – это класс, автоматически сгенерированный при импорте модели из БД; так как вы будете работать в своей предметной области в этом месте должен быть прописан *ваш класс*. Если в этот момент компилятор сообщает вам о том, что такой класс не описан, необходимо в разделе using прописать пространство имен вашей модели (название пространства можно подсмотреть в первой строке на рисунке 15), в нашем случае – **using LW1.Models.Entities;**

В следующей строчке открывается подключение к источнику данных. Таким образом реализуется использование объектов, унаследованных от IDisposable, а контекст данных именно таким и является (см. ссылку №4 в списке источников). В данном примере контекст данных называется abcEntities. В вашем случае название будет тем, которое вы укажете при создании подключения к данным (рисунок 9 – поле, отмеченное символом «\*»). При обращении к экземпляру контекста вы увидите добавленные наборы данных (в этом примере Answers, People, Questions). Поскольку метод должен вернуть список людей, обращаемся к набору People, а затем сортируем список сначала по возрасту в порядке убывания, а затем по фамилии и имени.

* 1. **Отображение списка записей**

Как и в прошлой лабораторной отобразим список в табличном виде (рисунок 18). Атрибуты для вывода можете выбрать сами, но отобразите в последнем столбце идентификатор записи в теге **<a>** (рисунок 19). Тег <a> предназначен для создания ссылок, чем мы и воспользуемся позднее.

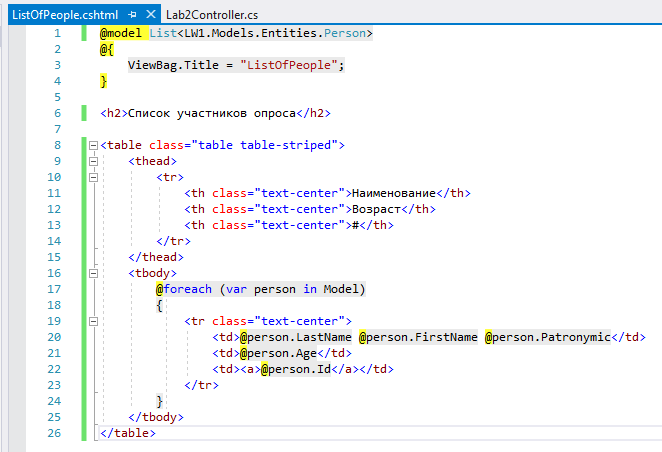


Рисунок 18 – Представление для отображения списка

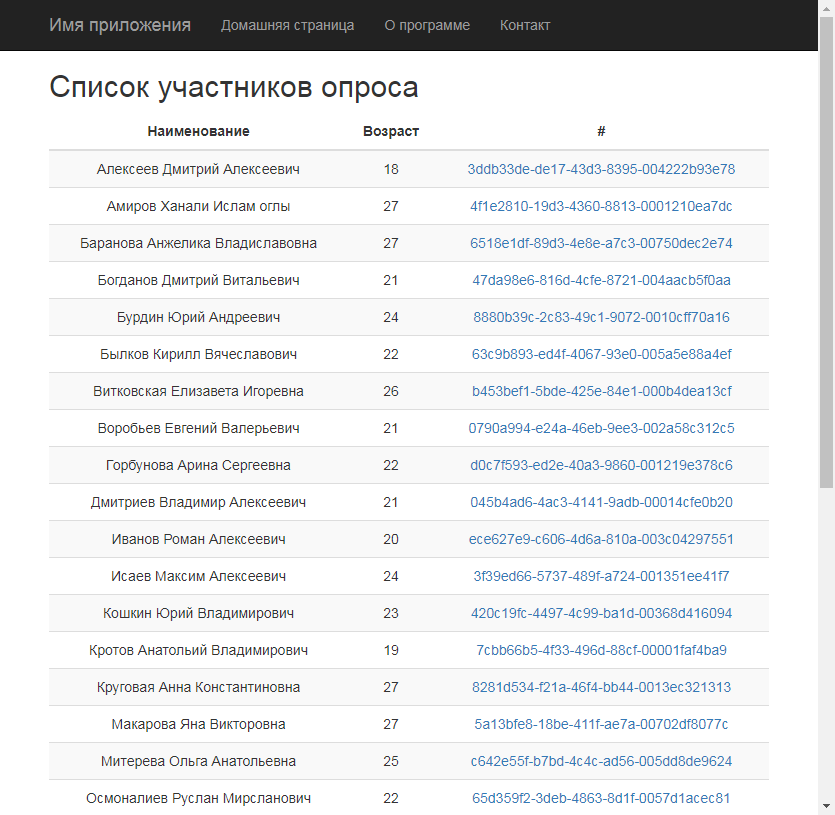


Рисунок 19 – Фрагмент страницы со списком участников опроса

На рисунке можно увидеть, что идентификатор, заключенный в тег <a> отображается характерным для ссылок цветом, но никак не реагирует на нажатие, поскольку для него не было указано никаких параметров – в том числе и адреса, по которому следует перейти.

Предположим, что по этой ссылке мы хотим перейти к карточке участника опроса, где будет указана вся хранящаяся о нем информация. Для этого понадобится создать метод, который будет возвращать одну единственную запись по запрошенному идентификатору, и представление для отображения этой информации.

* 1. **Добавление метода для получения одной записи**

Для создания метода вернемся в контроллер (и остановим отладку проекта!). Создаваемый метод должен возвращать страницу (ActionResult / ViewResult) по GET запросу, в котором передается идентификатор человека (параметр с типом данных Guid). Внутри метода должно происходить обращение к модели для получения данных с учетом переданного параметра. Полученный объект должен быть передан в представление. Таким образом, метод может выглядеть, как представлено на рисунке 20.

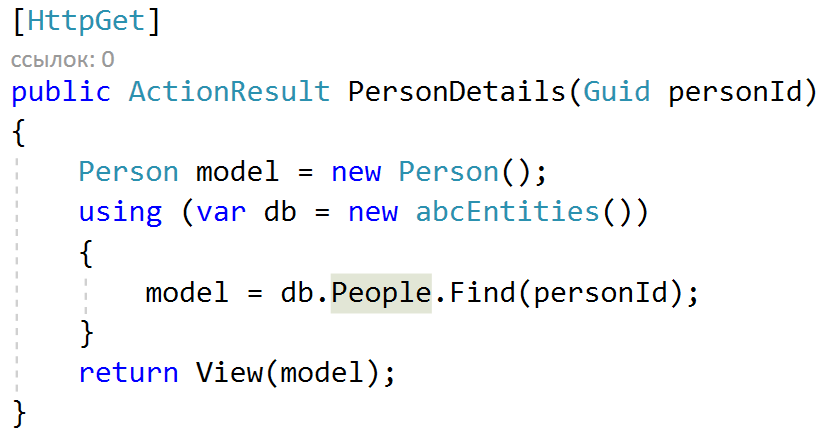


Рисунок 20 – Метод для получения объекта

Метод Find принимает в качестве параметра первичный ключ записи и возвращает объект, первичный ключ которого совпадает с переданным параметром (см. ссылку №6 в списке источников). Полученный объект передается в представление, название которого совпадает с методом.

Теперь, когда в контроллере создан метод, вернемся к представлению с отображением списка и укажем адрес для ссылки. Для этого воспользуемся конструкцией Html.ActionLink, которая создает гиперссылку на метод контроллера (рисунок 21). Все перезагрузки ActionLink содержат параметры для отображаемого текста и наименования метода, к которому предстоит обращаться, мы же воспользуемся вариантом, где также указывается наименование контроллера и передаваемый параметр. Для этой перезагрузки предполагается, что последним параметром будут указаны дополнительные HTML атрибуты. Мы ничего дополнительного пока дописывать не собираемся, поэтому в качестве последнего параметра пишем значение NULL.

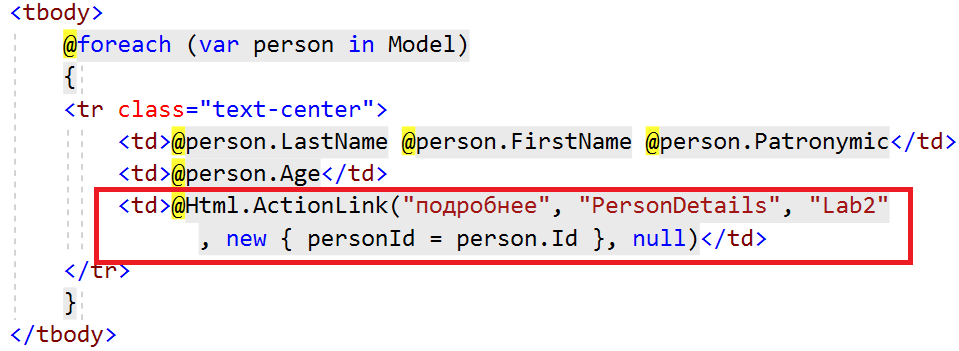


Рисунок 21 – Фрагмент кода представления с исправленной ссылкой на метод контроллера

* 1. **Отображение карточной формы**

При создании представления можно воспользоваться шаблоном, если указать модель и контекст данных, для которых оно создается (рисунок 22).

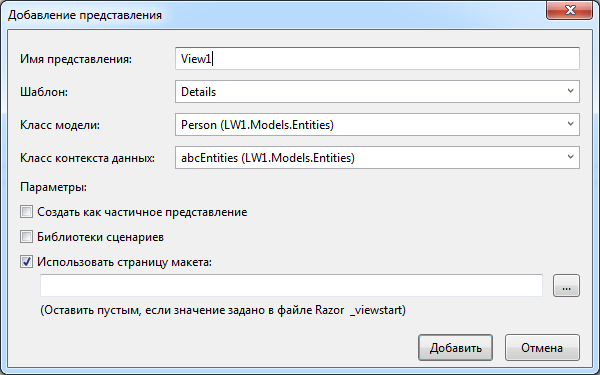


Рисунок 22 – Добавление нового представления

Представление, сгенерированное при помощи шаблона, представлено на рисунке 23.

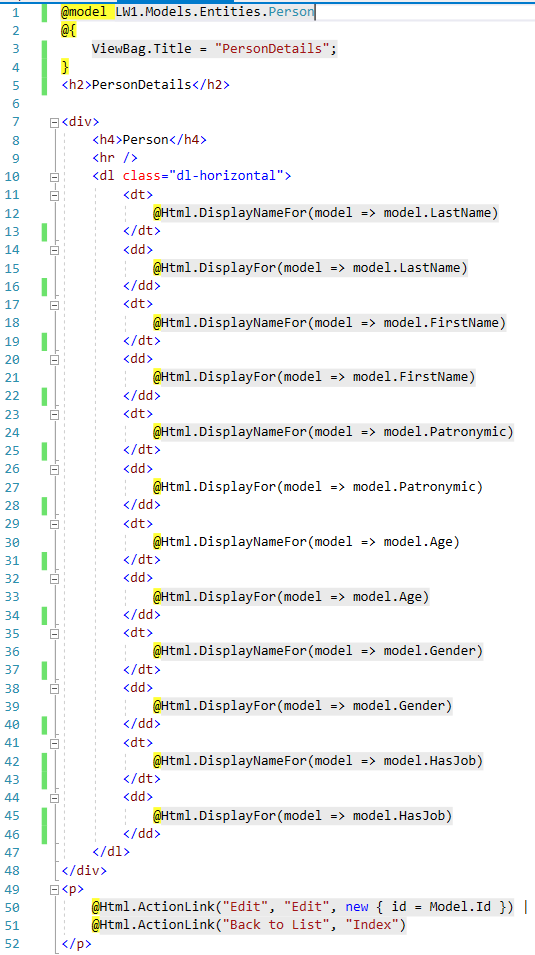


Рисунок 23 – Код html-разметки, сгенерированной по шаблону

Свойства класса отображаются в виде списка описаний. Тег <dl> (*description list*) обозначает сам список описаний, тег <dt> (*description term*) обозначает термин, тег <dd> (*description definition*) обозначает описание или определение. На странице это выглядит следующим образом:

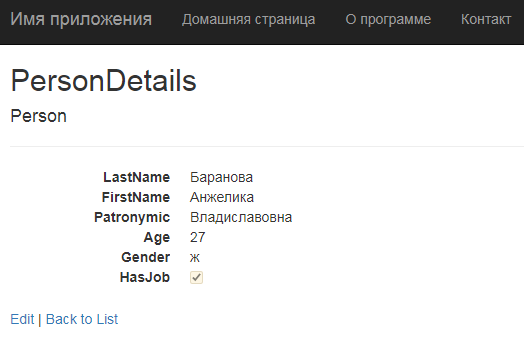


Рисунок 24 – Веб-страница представления, сгенерированного по шаблону

Далее представление «View1» следует переименовать в «PersonDetails» и изменить код разметки следующим образом:

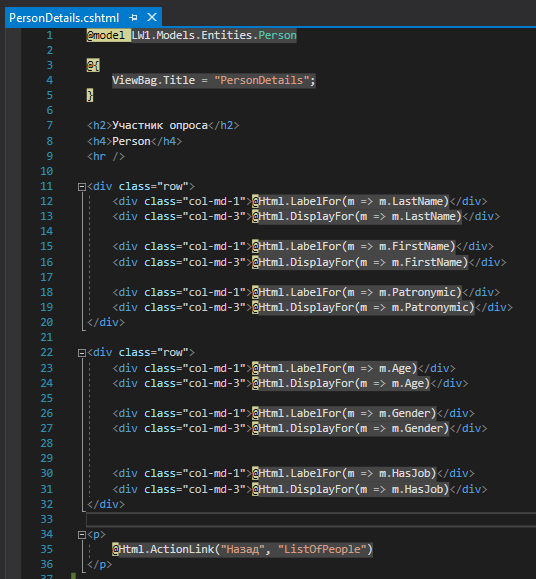


Рисунок 25 – Код представления после редактирования

Тогда расположение элементов на странице будет выглядеть следующим образом:

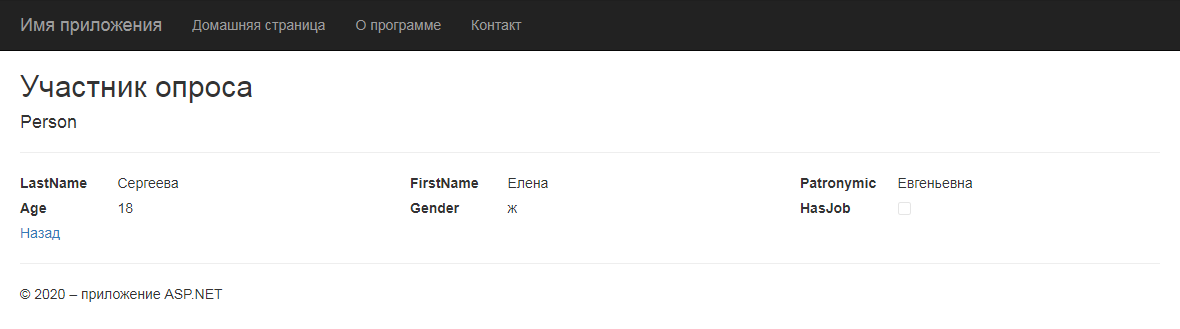


Рисунок 26 – Отображение представления после редактирования

Можно заметить, что внутри привычного тега <div> появился атрибут **class**, которому присвоено некоторое значение. Атрибут class отвечает за стилизацию содержимого тега, в который он заключен, в данном случае это тег <div>. Атрибут class можно использовать с любым HTML элементом. В измененном представлении элементы располагаются в две строки и три столбца.

Компоновка элементов, как на рисунке 26, обусловлена использованием классов, определенных в фреймворке Bootstrap. Данный фреймворк устанавливается по умолчанию при создании проекта по шаблону MVC (см. рисунок 27). Если по какой-то причине в вашем проекте его не оказалось, вы можете загрузить его через диспетчер пакетов NuGet.

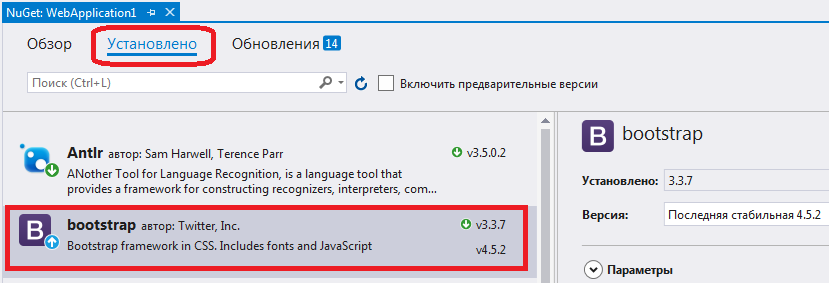


Рисунок 27 – Проверка установленных пакетов

Обратите внимание на версию фрейворка. Убедитесь, что у вас установлена версия НЕ позднее версии 3.4.1. В версиях Bootstrap 4 некоторые классы отличаются, и приведенные примеры могут отобразиться в этих версиях по-другому.

Фреймворк Bootstrap предоставляет классы для создания так называемого каркаса страницы и разметки элементов. Для этого в Bootstrap существует условная сетка (grid), которая и определяет разметку. Сетка состоит из 12 столбцов и при задании разметки можно указать, сколько столбцов должен занимать каждый элемент. Также сетка позволяет определять ряды, которые могут быть вложены друг в друга. Подсчет столбцов при этом ведется в рамках ряда, в котором находится элемент. Сетка позволяет создавать адаптивную разметку для отображения сайта на устройствах с разным разрешением экрана.

На рисунке 25 можно заметить, что атрибут class у тега div принимает значения row, col-md-1, col-md-3. Это значит, что на странице определены ряды и элементы, которые занимают 1 и 3 столбца в рамках ряда.

Пример размещения элементов относительно страницы и друг друга приведен на рисунках 28-30. Красной рамкой обозначается ряд (тег <div> с классом row). Синяя рамка – элемент внутри ряда, который занимает 9 из 12 столбцов ряда (определяется классом **col-\*-9**). Внутри этого элемента – два ряда в зеленой рамке. В первом зеленом ряду оранжевые элементы занимают по 5 столбцов каждый (определяется классом **col-\*-5**), и в правой части этого зеленого ряда остается свободное место.

Во втором зеленом ряду оранжевые элементы тоже занимают по 5 столбцов, но второй элемент сдвинут от первого на два столбца. Это определяется классом **col-\*-offset-2** в атрибутах этого элемента. В упомянутых классах под **\*** указывается тип устройства, для области просмотра которого предполагается разметка. На рисунке 28 представлена ситуация, когда область просмотра больше или равна предполагаемому расширению устройств, и явно отображаются все столбцы и пропуски между элементами. Когда область просмотра уменьшается и переходит границу для класса устройства (рисунки 29 и 30), то столбцы отображаются друг под другом и заполняют всю доступную ширину. Вложенность рядов и столбцов при этом сохранятся.

На рисунке 31 приведен код разметки первого элемента в красной рамке. Остальные элементы имеют ту же структуру и отличаются только указанием типа устройства в имени класса.

Подробнее про классы для разметки сетки и типы устройств можно прочитать по ссылке №7 в списке источников.

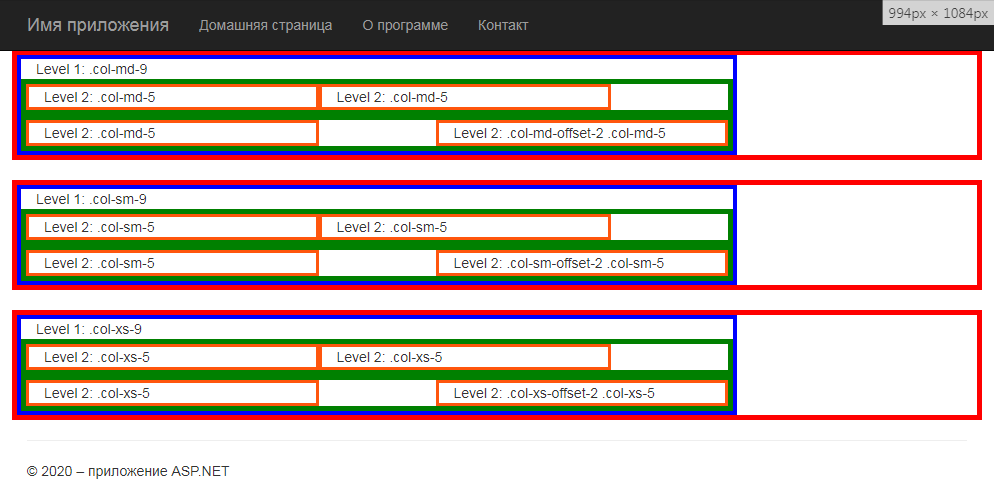


Рисунок 28 – Ширина области просмотра больше 992px

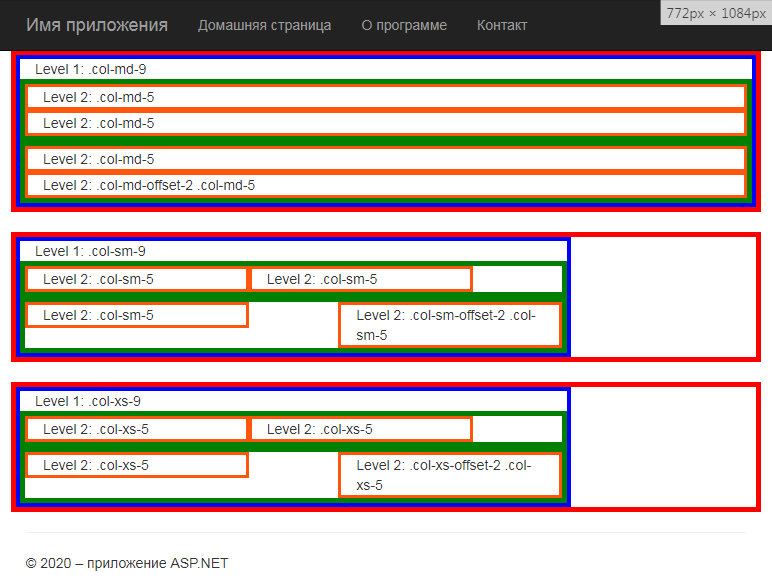


Рисунок 29 – Ширина области просмотра больше 768px

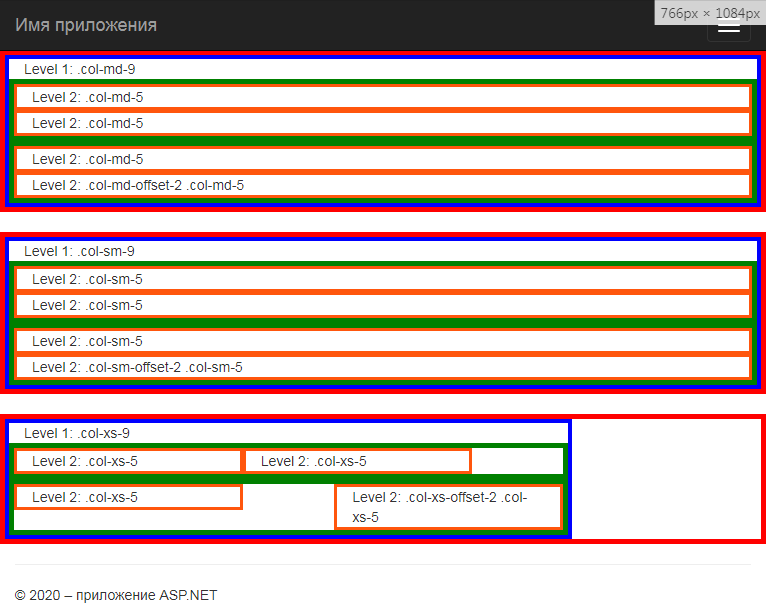


Рисунок 30 – Ширина области просмотра меньше 768px

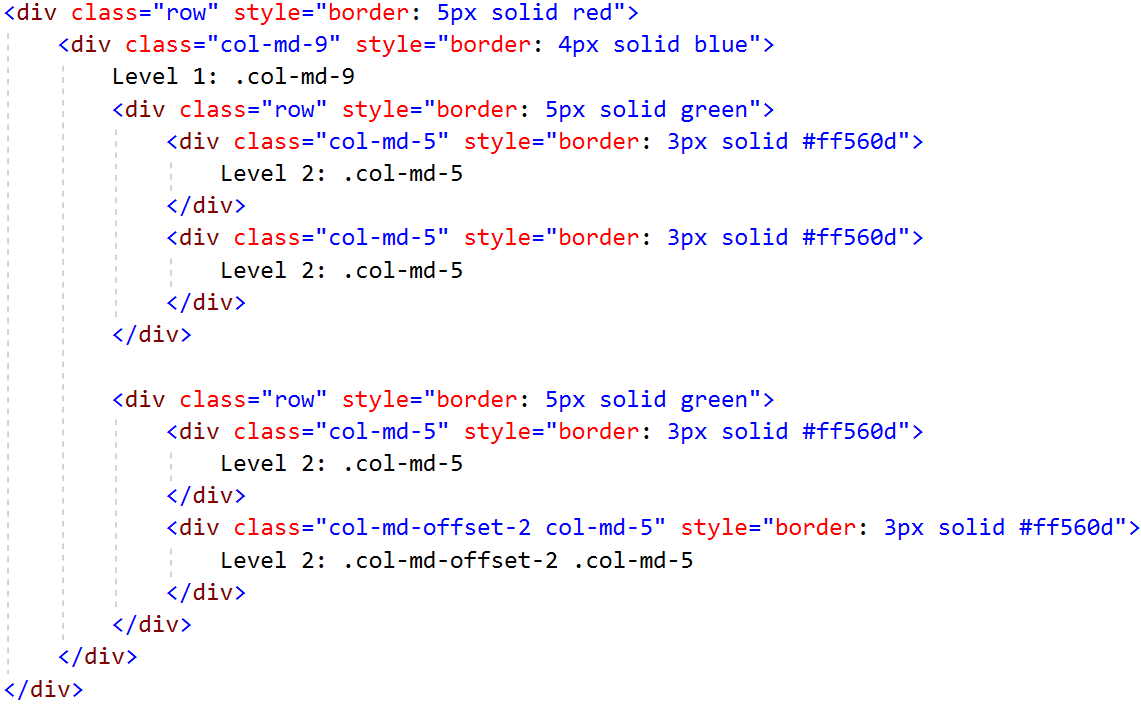


Рисунок 31 – Разметка первого ряда из рисунков 28-30

**Оформление отчетов**

Отчет о выполнении лабораторной работы оформляется в печатном виде, должен включать титульный лист и состоять из следующих структурных элементов и разделов:

– цель работы;

– постановка задачи;

– порядок выполняемых действий с описанием;

– выводы по работе.

**Использованные источники и материалы для самостоятельного изучения**

1. Сведения о проектах и решениях [Электронный ресурс] / Хранилище документации Майкрософт – Режим доступа: https://docs.microsoft.com/ru-ru/visualstudio/get-started/tutorial-projects-solutions?view=vs-2019
2. Работа с Entity Framework 6 [Электронный ресурс] / Professor Web – Режим доступа: https://professorweb.ru/my/entity-framework/6/level1/
3. Строка подключения [Электронный ресурс] / METANIT.COM – Режим доступа: https://metanit.com/sharp/adonet/2.2.php
4. Соглашения по наименованию в Code First [Электронный ресурс] / METANIT.COM – Режим доступа: https://metanit.com/sharp/entityframework/2.6.php
5. Использование объекта, реализующего IDisposable [Электронный ресурс] / Хранилище документации Майкрософт – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/api/system.idisposable?view=netcore-3.1#using-an-object-that-implements-idisposable
6. Различия методов Find(), FirstOrDefault() при использовании с Entity Framework [Электронный ресурс] / Сообщество Stack Overflow – Режим доступа: https://ru.stackoverflow.com/questions/693150/Различия-методов-find-firstordefault-при-использовании-с-entity-framework
7. Bootstrap 3 - Сетка [Электронный ресурс] / ИТ Шеф – всё о веб-разработке и программировании – Режим доступа: https://itchief.ru/bootstrap/grid-v3