ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

УДК 681.518.3
DOI 10.33514/ВК-1694-7711-2019-2-72-78

Абдымомунов Азим Абдымомунович, Жапаров Марат Турдалиевич
Н. Исанов атындагы КМАКТАУнун маалыматтык системалар жана технологиясы кафедрасынын магистранты,
Ф-м.и.к., доцент, Н. Исанов атындагы КМАКТАУнун маалыматтык системалар жана технологиясы кафедрасынын башчысы

Абдымомунов Азим Абдымомунович, Жапаров Марат Турдалиевич
магистрант кафедры информационные системы и технологии КГУСТА им. Н.Исанова,
к.ф.-м.н., доцент заведующий кафедры информационные системы и технологии КГУСТА им. Н.Исанова

Abdymomunov Azim Abdymomunovich, Japarov Marat Turdalievich
master's student of the Department of information systems and technologies of ksust named after N. Isanov,
Ph. D., associate Professor, head of the Department of information systems and technologies of the KSTU named after N. Isanov

БЕРИЛИШТЕР БАЗАСЫН WEB-ТИРКЕМЕ МЕНЕН ИНТЕГРАЦИЯЛООНУН ТЕХНОЛОГИЯСЫ

ТЕХНОЛГИЯ ИНТЕГРАЦИИ БАЗЫ ДАННЫХ С WEB-ПРИЛОЖЕНИЕМ

DATABASE INTEGRATION TECHNOLOGY WITH WEB APPLICATION

Аннотация: Бул макалада фреймворк ASP. Net MVC 5 алкагын колдонуу менен C # тилинде жазылган маалымат базасын долбоорлоодон жана толтуруудан кийин веб-тиркеме менен интеграциялоо технологиясы жөнүндө сөз болот.

Аннотация: В данной статье рассматривается технология интеграции после проектирования и наполнения Базы данных написанным на языке C# используя фреймворк ASP. Net MVC 5 с web-приложением.

Annotation: This article discusses the integration technology after designing and filling the database written in C # using the ASP. Net MVC 5 framework with a web application.

Негизги сөздөр: ADO. NET, маалыматтар базасы, ASP. Net, файл, объект, интеграция

Ключевые слова: ADO. NET, база данных, ASP. Net, файл, сущность, интеграция.

Keywords: ADO. NET, database, ASP. Net, file, entity, integration.

После проектирования и наполнения базы данных всеми необходимыми данными необходимо интегрировать БД с web-приложением, написанным на языке C# используя фреймворк ASP. Net MVC 5.

На данный момент подключить базу данных к сайту можно многочисленным количеством способов, но разобрав большую часть из них было решено остановиться на двух из них: технология ADO.NET и фреймворк Entity Framevork.

ADO. NET предоставляет собой технологию работы с данными, которая основана на платформе .NET Framework. Эта технология представляет нам набор классов, через которые мы можем отправлять запросы к базам данных, устанавливать подключения, получать ответ от базы данных и производить ряд других операций. Функционал ADO. NET построен таким образом, чтобы предоставить разработчикам унифицированный интерфейс для работы с самыми различными СУБД.

При работе с ADO. NET мы будем иметь прямое подключение к базе данных, а также нам будет необходимо писать все запросы на языке баз данных T-Sql. Преимуществами такого подхода является простота и скорость разработки и внедрения, но к сожалению, приходится расплачиваться безопасностью и скоростью обработки данных, а в нашем случае это критично.

В архитектуре ADO. NET существует привязка к физической структуре данных, поэтому при написании кода для обращения к базе данных необходимо помнить схемы таблиц, отношений. Для упрощения написания кода и его автоматического поддержания компанией Microsoft разработан Entity Framework, которая выводит абстракцию на новый уровень объектной модели. Это позволило провести отображение структуры базы данных на бизнес-объекты приложения, в результате чего позволило работать с данными как с обычными объектами языка.

Сущности (entities) — это концептуальная модель физической базы данных, которая отображается на предметную область. Формально говоря, эта модель называется моделью сущностных данных (Entity Data Model — EDM).

Модель EDM представляет собой набор классов клиентской стороны, которые отображаются на физическую базу данных. Тем не менее, нужно понимать, что сущности вовсе не обязаны напрямую отображаться на схему базы данных, как может показаться, исходя из названия. Сущностные классы можно реструктурировать для соответствия существующим потребностям, и исполняющая среда Entity Framework отобразит эти уникальные имена на корректную схему базы данных.

Центральной концепцией Entity Framework является понятие сущности или entity. Сущность представляет набор данных, ассоциированных с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их наборами.

Любая сущность, как и любой объект из реального мира, обладает рядом свойств. Например, если сущность описывает человека, то мы можем выделить такие свойства, как имя, фамилия, рост, возраст, вес. Свойства необязательно представляют простые данные типа int, но и могут представлять более комплексные структуры данных. И у каждой сущности может быть одно или несколько свойств, которые будут отличать эту сущность от других и будут уникально определять эту сущность. Подобные свойства называют ключами.

При этом сущности могут быть связаны ассоциативной связью один-ко-многим, один-ко-одному и многие-ко-многим, подобно тому, как в реальной базе данных происходит связь через внешние ключи.

Отличительной чертой Entity Framework является использование запросов LINQ для выборки данных из БД. С помощью LINQ мы можем не только извлекать определенные строки, хранящие объекты, из базы данных, но и получать объекты, связанные различными ассоциативными связями.

Другим ключевым понятием является Entity Data Model. Эта модель сопоставляет классы сущностей с реальными таблицами в БД. Entity Data Model состоит из трех уровней: концептуального, уровень хранилища и уровень сопоставления (маппинга).

На концептуальном уровне происходит определение классов сущностей, используемых в приложении. Уровень хранилища определяет таблицы, столбцы, отношения между таблицами и типы данных, с которыми сопоставляется используемая база данных. Уровень сопоставления (маппинга) служит посредником между предыдущими двумя, определяя сопоставление между свойствами класса сущности и столбцами таблиц. Таким образом, мы можем через классы, определенные в приложении, взаимодействовать с таблицами из базы данных.

Entity Framework предполагает три возможных способа взаимодействия с базой данных:

- Database first: Entity Framework создает набор классов, которые отражают модель конкретной базы данных.
- Model first: сначала разработчик создает модель базы данных, по которой затем Entity Framework создает реальную базу данных на сервере.
- Code first: разработчик создает класс модели данных, которые будут храниться в БД, а затем Entity Framework по этой модели генерирует базу данных и ее таблицы.

Исходя из этого всего было решено вести интеграцию базы данных и web-приложения посредством использования фреймворка Entity Framework.

Разработав саму модель базы данных необходимо было написать ее управление для сайта, как для удобства, так и для безопасности. Используя фреймворк Entity Framework, мы облегчим разработку и дадим разработчику frontend части web-приложения удобный функционал по управлению базой данных.

Так как web-приложение разрабатывалось на языке C# используя фреймворк ASP.net то средой разработки была Visual Studio 2017.

В первую очередь мы создадим проект. Для этого в среде разработки Visual Studio 2017 выполним следующие действия: Файл – Создать – Проект – Веб-приложение ASP. NET (.NET Framework). Далее в открывшемся окне введем имя проекта и выберем платформу .NET Framework 4.5 так как на данный момент она является самой распространенной и стабильной. На рис. 1. изображено окно создания нового приложения.

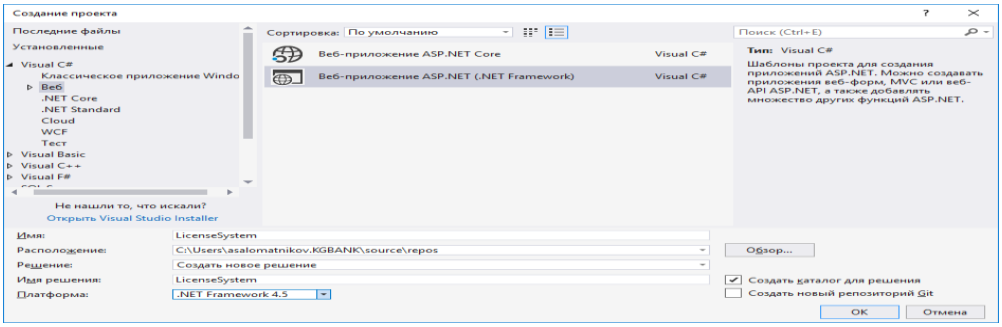

Рис. 1. Окно создания нового проекта Visual Studio

На нас лежит разработка БД поэтому разработку веб части мы оставим другим. Разработка базы данных начинается с создания модели данных для каждой таблицы. Visual Studio автоматически создала папки для разработки, поэтому в папке «Models» мы создадим модели для всех таблиц описанных ранее.

Чтобы создать модель необходимо в обозревателе решений в папке «Models» нажать правую кнопку мыши и выбрать действие – Добавить – Класс и в появившемся окне задать имя класса. После создания файла модели автоматически откроется окно ее редактирования. В окне редактирования мы уже пишем код создания. На рис. 2. изображен пример кода создания модели для хранения файлов.

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

namespace IdentityTest.Models
{
    public class Files
    {
        [Key]
        public Guid FileId { get; set; }
        public DateTime? DateCreate { get; set; }
        public Guid StatementId { get; set; }
        public byte[] FileData { get; set; }

        public int ContentLength { get; set; }
        public string ContentType { get; set; }
        public string FileName { get; set; }
        public string RejectErrorText { get; set; }

        public string FileStatementType { get; set; }

    }
}
```

Рис. 2. Пример кода модели File Model

Таким образом мы создадим остальные модели для всех спроектированных таблиц. Количество моделей должно быть равной или больше количества спроектированных таблиц.

Создав все модели необходимо создать контроллер, но перед этим необходимо подключить сам фреймворк Entity Framework. Благодаря нему и подходу "Code first" достаточно создать просто модели и таблицы в БД будут созданы автоматически.

Для того чтобы подключить фреймворк Entity Framework необходимо в обозревателе решений Visual Studio в нашем проекте нажать правой кнопкой по ссылкам и выбрать пункт контекстного меню под названием «Управление пакетами Nu Get». В открывшемся окне выбрать вкладку «Обзор» и в строке поиска ввести «Entity Framewok». Выбрав из списка найденных результатов нужный нажать на него и установить. На рис. 3 изображено окно обозревателя пакетов NuGet с уже установленным Entity Framewok.

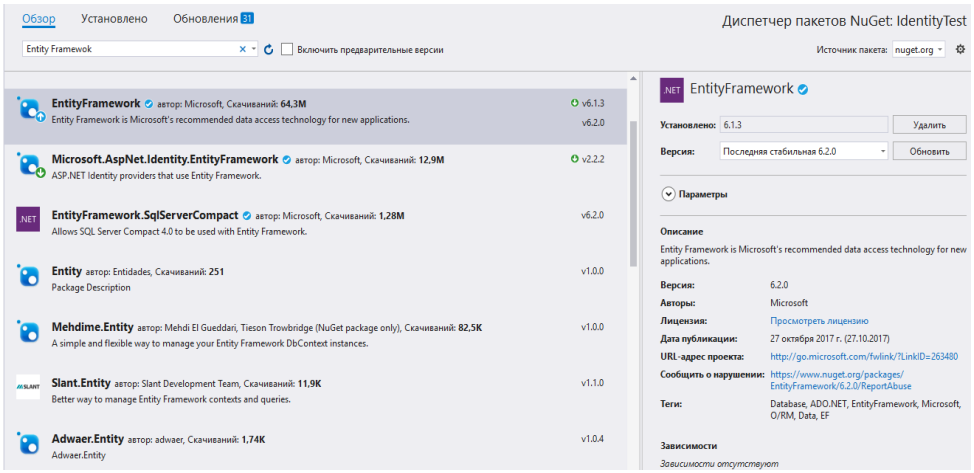

Рис. 3 Окно обозревателя пакетов NuGet

Далее по процессу разработки нам необходимо создать файл контекста. Данный файл будет выступать в роли посредника для подключения к базе данных через Entity Framework. Контекст данных представляет собой класс, производный от класса Db Context. Контекст данных содержит одно или несколько свойств типа Db Set <T>, где T представляет тип объекта, хранящегося в базе данных. Создадим контекст данных для работы с вышеприведенными моделями.

Также в файле контекста можно наполнить первоначальными данными наши таблицы. Для проверки корректности заполним таблицу со статусами первоначальными данными. На рис. 4 изображен участок кода контекста данных с классом, позволяющим наполнить таблицу со статусами первоначальными данными.

```
public class ApplicationContext : IdentityDbContext<ApplicationUser>
{
    public ApplicationContext() : base("IdentityDb")
    {
    }

    public static ApplicationContext Create()
    {
        return new ApplicationContext();
    }

    public DbSet<Statement> Statement { get; set; }
    public DbSet<StatementStatus> StatementStatus { get; set; }
    public DbSet<KindOfActivity> KindOfActivity { get; set; }
    public DbSet<StreetType> StreetType { get; set; }
    public DbSet<Streets> Streets { get; set; }
    public DbSet<DateDetail> DateDetail { get; set; }
    public DbSet<EmployeesModel> EmployeesModel { get; set; }
    public DbSet<Files> Files { get; set; }
    public DbSet<ApplicationRole> ApplicationRole { get; set; }
    public DbSet<ObjectType> ObjectType { get; set; }
    public DbSet<Specialty> Specialty { get; set; }
    public DbSet<Region> Region { get; set; }
    public DbSet<Area> Area { get; set; }
    public DbSet<City> City { get; set; }
}

public class ApplicationDbInitializer : CreateDatabaseIfNotExists<ApplicationContext>
{
    protected override void Seed(ApplicationContext context)
    {
        var s1 = new StatementStatus {Id = 1, Name = "Формирование"};
        var s2 = new StatementStatus {Id = 2, Name = "На корректировке"};
        var s3 = new StatementStatus {Id = 3, Name = "На рассмотрении"};
        var s4 = new StatementStatus {Id = 4, Name = "Утверждено"};
        var s5 = new StatementStatus {Id = 5, Name = "Отменено"};

        context.StatementStatus.Add(s1);
        context.StatementStatus.Add(s2);
        context.StatementStatus.Add(s3);
        context.StatementStatus.Add(s4);
        context.StatementStatus.Add(s5);
    }
}
```

Рис. 4 Код файла контекста данных

Класс по наполнению первоначальными данными Application Db Initializer наследуется от класса Create Database If Not Exists, и это означает что база данных будет создаваться если она еще не создана в остальных случаях данный метод запускаться не будет.

Последним шагом нужно будет прописать настройку, которая бы подключала базу данных к сайту. Для этого в обозревателе решений приложения нужно открыть файл web.config и прописать в подразделе <configuration> код в котором будет указано путь к базе данных, ее тип, данные для авторизации и само название подключения которое мы используем в файле контекста.

Код подключения:

<connectionStrings>

<add name = "Identity Db" connection String = "Server = 127.0.0.1; Database=License System; User ID=salex; Password=12341234qwer; Trusted_Connection=False;" provider Name="System. Data. Sql Client"/>
</connection Strings>

После проведенных манипуляция выше мы наконец подключили наше web-приложение к БД, что позволит нам проводить манипуляции с данными такими как удаление данных, добавление данных и изменение данных в таблицах, а также выстраивать сложные запросы.

Список использованной литературы:

1. Стивен Сандерсон - ASP .NET MVC Framework с примерами на C#
2. Александр Бондарь - Microsoft SQL Server 2014.
3. http://www.med.gov.kg/ - Сайт Министерства здравоохранения КР от куда брались нормативы и документы для нашей работы.

References:

1. Stephen Sanderson-ASP .NET MVC Framework with examples in C#
2. Alexander Bondar-Microsoft SQL Server 2014.
3. http://www.med.gov.kg/ - Website of the Ministry of health of the Kyrgyz REPUBLIC from where the standards and documents for our work were taken.

Рецензент: к.т.н., доцент Абдулаев А.А.

УДК 681.518.3
DOI 10.33514/ВК-1694-7711-2019-2-78-83

Абдымомунов Азим Абдымомунович, Жапаров Марат Турдалиевич
Н. Исанов атындагы КМАКТАУнун маалыматтык системалар жана технологиясы кафедрасынын магистранты,
Ф-м.и.к., доцент, Н. Исанов атындагы КМАКТАУнун маалыматтык системалар жана технологиясы кафедрасынын башчысы

Абдымомунов Азим Абдымомунович, Жапаров Марат Турдалиевич
магистрант кафедры информационные системы и технологии КГУСТА им. Н.Исанова,
к.ф.-м.н., доцент заведующий кафедры информационные системы и технологии КГУСТА им. Н.Исанова

Abdymomunov Azim Abdymomunovich, Japarov Marat Turdalievich
master's student of the Department of information systems and technologies of ksust named after N. Isanov,