5. Top 10 Chat, Audio & Video Calling API & SDK Providers for Enterprise Business// интернет-источник: https://habr.com/ru/post/453374/

Кимачук И.В.
студент, Карагандинский университет имени академика Е.А.Букетова
Попова Н.В.
ст.преподаватель, Карагандинский университет имени академика Е.А.Букетова

РАЗРАБОТКА WEB-ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ ASP.NET MVC

С каждым годом информационные технологии становятся все более важным в жизни людей и общества. Многочисленные компании, образовательные и государственные учреждения, имеют собственный Web-ресурс, несущий в себе всю информацию о деятельности данной организации.

В настоящий период создание веб-приложений считается одним из самых перспективных течений деятельности для большинства компаний, фирм, работающих в сфере сверхтехнологичных цифровых и компьютерных технологий. Отказ от традиционного программного обеспечения и перенос бизнес-инструментов на веб становиться трендом. Тот, кто владеет объективной информацией и грамотно представляет свои идеи, разрабатывая современные бизнес-модели, будет всегда добиваться успеха.

Создание веб-приложений стало актуальной темой для многих специализированных компаний, предприятий и одновременно легкодоступной для пользователей.

Для создания приложения BookStore с применением технологии ASP.NET MVC в среде разработки Visual Studio выбираем тип проекта ASP.NET MVC 4/5 Web Application и шаблон приложения Empty.

Для текущей задачи можно выделить две области данных: информация о книге и информация, связанная с оформлением покупки. Соответственно, создадим две модели. В приложении уже

определена директория Models. В ней будут находиться модели. Нажмем на эту директорию правой кнопкой мыши и в появившемся меню выберем Add->Class (Добавить->Класс) Назовем новый класс или модель Book и добавим в него код, описывающий модель книги и модель Purchase для оформления покупки (рис. 1). Дополнительно в папку Models добавим новый класс BookContext с кодом представленным на рисунке 2. Чтобы создать контекст, нужно унаследовать новый класс от класса DbContext. Свойства типа public DbSet<Book> Books {get; set;}помогают получать набор данных определенного типа (в данном случае типа Book).

```
namespace BookStore.Models
{
    public class Book
    {
        // ID книги
        public int Id { get; set; }
        // название книги
        public string Name { get; set; }
        // автор книги
        public string Author { get; set; }
        // цена
        public int Price { get; set; }
    }
public class Purchase
{
    // ID покупки
    public int PurchaseId { get; set; }
    // имя и фамилия покупателя
    public string Person { get; set; }
    // адрес покупателя
    public string Address { get; set; }
    // ID книги
    public int BookId { get; set; }
    // дата покупки
    public DateTime Date { get; set; }
}
}
```

Рисунок 1. Описание моделей приложения

```
9   public class BookContext:DbContext
10  {
        Ссылок: 14
11      public DbSet<Book> Books { get; set; }
        ссылка: 1
12      public DbSet<Purchase> Purchases { get; set; }
13  }
```

Рисунок 2. Листинг класса BookContext

В данном случае будет использоваться так называемый подход Code First, т.е. по существующим моделям EntityFramework будет создавать таблицы в базе данных. В секции configSections файла web.config нужно указать путь подключения к базе данных.

Для управления моделями в папку Controllers добавим контроллер HomeController. Контроллер по сути и является основным звеном приложения, которое связывает модель и интерфейс пользователя. Контроллер - это обычный класс, который наследуется от базового класса Controller. По умолчанию в контроллере содержится единственный метод Index, который возвращает некоторый исход метода View() - будущее представление. Изменим код данного метода как показано на рисунке 3 и добавим представление для данного метода. Visual Studio автоматически создаст и откроет представление Index.cshtml. Нужно изменить код по умолчанию в соответствии с требоаниями приложения.

```
using System.Web.Mvc;
using BookStore.Models;

namespace BookStore.Controllers
{
    public class HomeController : Controller
    {
        // создаем контекст данных
        BookContext db = new BookContext();

        public ActionResult Index()
        {
            // получаем из бд все объекты Book
            IEnumerable<Book> books = db.Books;
            // передаем все полученный объекты в динамическое свойство Books в ViewBag
            ViewBag.Books = books;
            // возвращаем представление
            return View();
        }
    }
}
```

Рисунок 3. Листинг контроллера

В папку Models добавим новый класс BookDbInitializer для инициалицации начальных данных в модели напишем код представленный на рисунке 4.

```
namespace BookStore.Models
{
    public class BookDbInitializer : DropCreateDatabaseAlways<BookContext>
    {
        protected override void Seed(BookContext db)
        {
            db.Books.Add(new Book { Name = "Война и мир", Author = "Л. Толстой", Price = 220 });
            db.Books.Add(new Book { Name = "Отцы и дети", Author = "И. Тургенев", Price = 180 });
            db.Books.Add(new Book { Name = "Чайка", Author = "А. Чехов", Price = 150 });

            base.Seed(db);
        }
    }
}
```

Рисунок 4. Заполнение таблиц базы данных

В файл Global.asax добавим метод Application_Start, который запустит создание класса, и заполнение базы данных исходными данными выполнив при старте приложения код:

```
Database.SetInitializer(new BookDbInitializer());
```

Импортируем в файл Global.asax пространства имен BookStore.Models и System.Data.Entity, или класс BookDbInitializer будет недоступен.

После запуска приложения, будет видно, что в таблице находятся ранее определенные данные (рис.5).

Название книги	Автор	Цена	
Война и мир	Л. Толстой	250	Купить
Отцы и дети	И. Тургенев	180	Купить
Чайка	А. Чехов	200	Купить
Страж	А. Пехов	230	Купить
Метро 2033	Д. Глуховский	170	Купить

Рисунок 5. Таблица данных

Метод Buy в контроллере HomeController отвечает за обработку выбора пользователя при оформлении покупки. Добавим следующие два метода отвечающие за получение и отображение данных о книгах на Web-странице (рис. 6), и соответствующее им представление (рис. 7) в папку View.

6

```
[HttpGet]
Ссылок: 0
public ActionResult Buy(int id)
{
    ViewBag.BookId = id;
    return View();
}
[HttpPost]
Ссылок: 0
public string Buy(Purchase purchase)
{
    purchase.Date = DateTime.Now;
    // добавляем информацию о покупке в базу данных
    db.Purchases.Add(purchase);
    // сохраняем в бд все изменения
    db.SaveChanges();
    return "Спасибо," + purchase.Person + ", за покупку!";
}
```

Рисунок 6. Метод, отвечающий за обработку выбора покупателя

```
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Покупка</title>
</head>
<body>
    <div>
        <h3>Форма оформления покупки</h3>
        <form method="post">
            <input type="hidden" value="@ViewBag.BookId" name="BookId" />
            <table>
                <tr><td><p>Введите свое имя </p></td>
                    <td><input type="text" name="Person" /> </td></tr>
                <tr><td><p>Введите адрес :</p></td><td>
                    <input type="text" name="Address" /> </td></tr>
                <tr><td><input type="submit" value="Отправить" /> </td><td></td></tr>
            </table>
        </form>
    </div>
</body>
</html>
```

Рисунок 7. Листинг представления для метода Buy

Для совершения покупки нужно перейти на страницу оформления заказа заполнить поля и нажать кнопку "Отправить". После этого заявка на покупку книги попадет в базу данных, а в браузере отобразится соответствующее уведомление. На рисунке 8 представлен пример главной страницы приложения

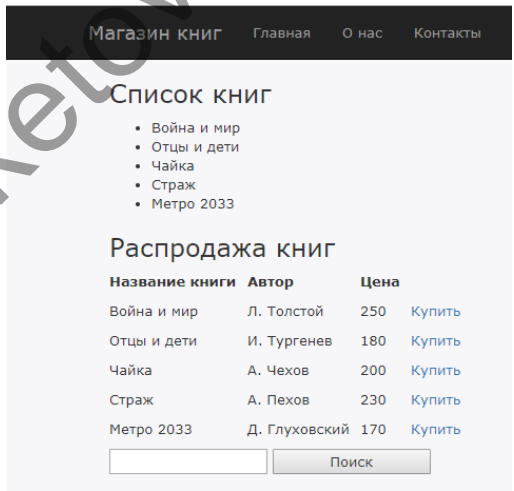

Рисунок 8. Главная страница приложения

В шапке приложения расположена панель с навигацией по сайту. Для поиска интересующей книги можно воспользоваться текстовым полем и кнопкой «Найти» в нижней части окна. Так же можно редактировать, удалять и получать информацию о каждой книге. Что бы добавить новую книгу в список и таблицу, нужно вписать данные о книге, такие как название, автор и цена затем нажать “Create”.

Данная таблица сделана с помощью шаблонов формирования, которые дают возможность по заданной модели и контексту данных сформировать всю нужную разметку для представлений можно управлять записями в базе данных, и значитель ускорить процесс разработки Web-приложения

Список используемой литературы

1 Фримен А., ASP.NET Core MVC 2 с примерами на С-шарп.- Киев:Диалектика. 2019.- 1008 с.

2 Фримен А., Entity Framework Core 2 для ASP.NET Core MVC. - Киев: Диалектика. 2019.- 624 с.

3 https://docs.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2019

4 https://habr.com/ru/sandbox/79099/

5 https://metanit.com/sharp/entityframework/1.1.php

6 https://www.entityframeworktutorial.net/what-is-entityframework.aspx

Нұрбекқызы А., Мархаматова Н.М.
студенты 1 курса, Карагандинский технический университет имени А.Сагинова
Шаихова Г.С.
к.т.н, и.о.доцента, Карагандинский технический университет имени А.Сагинова

АВТОМАТИЗАЦИЯ РАБОТЫ АВТОШКОЛЫ

В последние годы персональный компьютер стал обязательной частью каждой автошколы. Благодаря быстрому формирова-