

Лабораторная работа №2

Основной для выполнения лабораторной работы № 2 послужило веб приложение на MVC 4.

По умолчанию в этом приложении уже есть логика по работе с пользователем. Авторизация, создание нового пользователя уже доступна. Так же доступны пару пунктов меню из которых в последствии можно сделать вызовы скрипта для прогона всех методов API.

В шаблоне в первую очередь заменили заголовки About, Contact на другие см. рисунок 1.

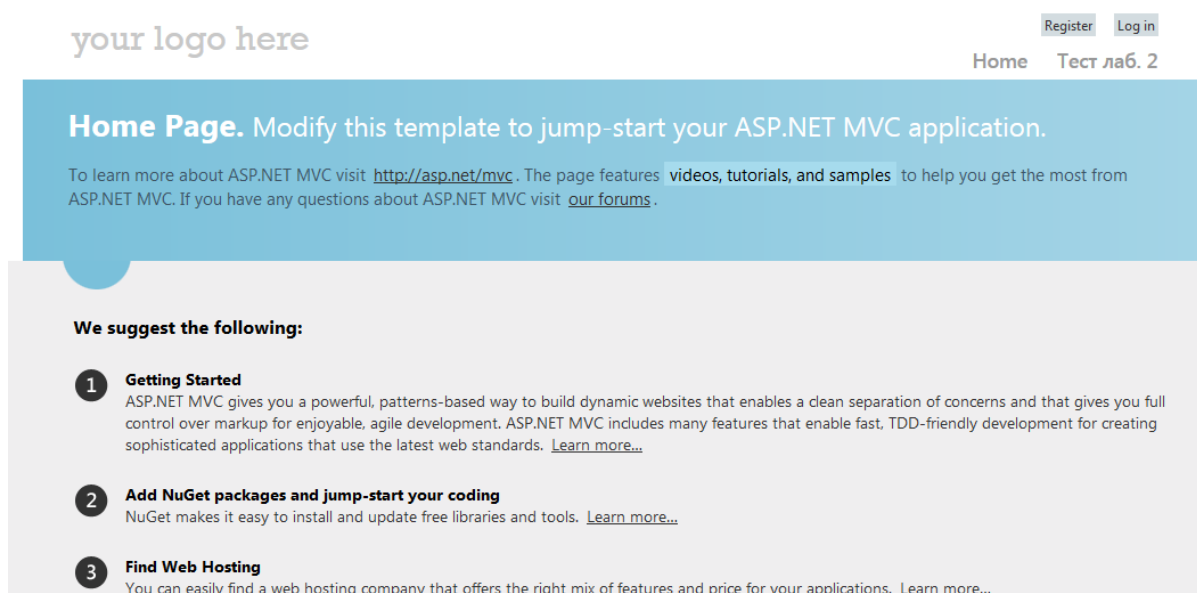


Рисунок 1 – Главная страница веб приложения.

Для реализации API используется технология WebAPI. В папке Controllers создан класс MainCotrollers.cs который содержит все методы API.

Диаграмма классов для этого модуля выглядит так (см. рисунок 2)



Рисунок 2 – Диаграмма классов модуля MainControlles.cs

Как видно из рисунка 2 на диаграмме представлены 4 основные сущности.

Customers (Покупатели),

Deliveries (Доставки),

Orders (Заказы),

Goods (Товары).

Остальные таблицы представлены на рисунке 3.

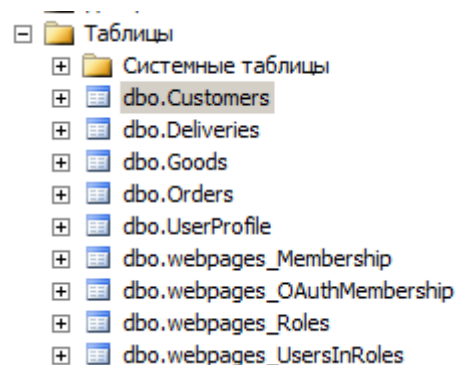


Рисунок 3 – Список таблиц базы данных

Таблицы UserProfile, webpages_Membership, webpages_OAuthMembership, webpages_Roles, webpages_UserInRoles. Это таблицы, которые были

созданы шаблоном Web Application в среде Microsoft Visual Studio 2010. Таблицы для основных сущностей создаём в Microsoft Sql Management Studio.


	Имя столбца	Тип данных	Разрешит...
	ID	int	<input type="checkbox"/>
	Title	nchar(255)	<input type="checkbox"/>
	Phone	nchar(10)	<input type="checkbox"/>
			<input type="checkbox"/>

Рисунок 4 – Создание таблицы Customers


	Имя столбца	Тип данных	Разрешит...
	ID	int	<input type="checkbox"/>
	Title	nchar(255)	<input type="checkbox"/>
	Cost	numeric(18, 2)	<input type="checkbox"/>
			<input type="checkbox"/>

Рисунок 5 – Создание таблицы Deliveries


	Имя столбца	Тип данных	Разрешит...
	ID	int	<input type="checkbox"/>
	Cost	numeric(18, 2)	<input type="checkbox"/>
	Title	nchar(255)	<input type="checkbox"/>
			<input type="checkbox"/>

Рисунок 6 – Создание таблицы Goods

	Имя столбца	Тип данных	Разрешит...
	ID	int	<input type="checkbox"/>
	CustomerId	int	<input checked="" type="checkbox"/>
	GoodId	int	<input checked="" type="checkbox"/>
	DeliveryId	int	<input checked="" type="checkbox"/>
	Count	int	<input type="checkbox"/>
			<input type="checkbox"/>

Рисунок 7 – Создание таблицы Orders

Также необходимо изменить стандартную таблицу UserProfile для поддержки авторизации OAuth 2.


	Имя столбца	Тип данных	Разрешит...
	UserId	int	<input type="checkbox"/>
	UserName	nvarchar(MAX)	<input checked="" type="checkbox"/>
	Token	nchar(36)	<input checked="" type="checkbox"/>
	ExpirationDate	datetime	<input checked="" type="checkbox"/>
	RefreshToken	nchar(36)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Рисунок 8 – Правка таблицы UserPofile

В UserProfile добавляются поля Token, ExpirationDate, RefreshToken.

Для доступа к данным используется Entity Framework. Пакет для 6 версии данного фреймворка можно установить из Nget. Для этого создадим проект ModelLib. И в класс MainModel.cs поместим следующий набор прокси классов для таблиц.

```
namespace ModelLib
{
    [Table("UserProfile")]
    public class UserProfile
    {
        [Key]
        [DatabaseGeneratedAttribute(DatabaseGeneratedOption.Identity)]
        public int UserId { get; set; }
        public string UserName { get; set; }
        public string Token { get; set; }
        public string RefreshToken { get; set; }
        public DateTime? ExpirationDate { get; set; }
    }
    [Table("Goods")]
    public class Good
    {
        [Key]
        [DatabaseGeneratedAttribute(DatabaseGeneratedOption.Identity)]
        public int ID { get; set; }
        public decimal Cost { get; set; }
        public string Title { get; set; }
    }
    [Table("Deliveries")]
    public class Delivery
    {
        [Key]
        [DatabaseGeneratedAttribute(DatabaseGeneratedOption.Identity)]
        public int ID { get; set; }
        public decimal Cost { get; set; }
        public string Title { get; set; }
    }
    [Table("Customers")]
    public class Customer
    {
        [Key]
        [DatabaseGeneratedAttribute(DatabaseGeneratedOption.Identity)]
        public int ID { get; set; }
    }
}
```

```

        public string Title { get; set; }
        public string Phone { get; set; }
    }
    [Table("Orders")]
    public class Order
    {
        [Key]
        // [DatabaseGeneratedAttribute(DatabaseGeneratedOption.Identity)]
        public int ID { get; set; }
        public int? GoodId { get; set; }

        public int? CustomerId { get; set; }
        public int? DeliveryId { get; set; }
        public int Count { get; set; }
    }
}

```

В HomeController.cs в обработчик About внесём код, который будет проверять все вызовы Api и результат выведет на страницу. Особенностью представления About будет то, что она будет работать отображать результаты как для авторизованного api так и для не авторизованного АПИ. На странице существует кнопка перехода

```

<a href =
"/Account/Authorize?response_type=code&client_id=464119&redirect_uri=htt
p://localhost:60116/Home/About" > Авторизовать приложение по OAuth
2.0</a>

```

Данная кнопка инициализирует процесс авторизации по OAuth 2.0.

Как видно из URL view Authorize.cs html принимает запрос. Чтобы это view отобразилось понадобится авторизация пользователя. После успешной авторизации у пользователя запрашивается подтверждение на доступ по OAuth 2.0

```

using (Html.BeginForm("AccessSuccess", "Account", new { redirect_uri =
 ViewBag.redirect_uri, client_id = ViewBag.client_id }))
{
    <fieldset>
        <input type="submit" value="Подтвердить" />
    </fieldset>
}

```

Принимает ответ контроллер AccessSuccess (в модуле AccountController.cs). Контролер перенаправляет на redirect_uri запрос

добавляя в параметры code, который равен имени пользователя. Когда контроллер About получает не пустое значение code инициализируется процесс получения токена авторизации.

По url [имя_сервера]/api/oauth/token

делается post запрос с

Grant_type = authorization_code, client_id =464119, client_secret="deadbeef", code=[код из параметров].

За выдачу токена отвечает контроллер OAuthController.cs

```
public class OAuthController : ApiController
{
    [Route("~/api/oauth/token")]
    public AuthParams Post(TokenParams tokenParams)
    {
        AuthParams par = new AuthParams();
        // создать access_token
        using (UsersContext context = new UsersContext())
        {
            UserProfile profile = null;
            if (tokenParams.grant_type == "authorization_code")
            {
                profile = context.UserProfiles.Where(x => x.UserName ==
tokenParams.code).FirstOrDefault();
            }
            else
            {
                profile = context.UserProfiles.Where(x => x.RefreshToken ==
tokenParams.refresh_token).FirstOrDefault();
            }
            if (profile != null)
            {
                par.access_token = Guid.NewGuid().ToString();
                par.token_type = "bearer";
                par.refresh_token = Guid.NewGuid().ToString();
                par.expires_in = 60;

                profile.Token = par.access_token;
                profile.RefreshToken = par.refresh_token;
                profile.ExpirationDate = DateTime.Now.AddSeconds(par.expires_in);
                // сохраняем параметры доступа по токenu
                context.SaveChanges();
            }
        }
        return par;
    }
}
```

В случае успешности операции будет возвращён набор для дальнейшей работы с методами API.

```
public class AuthParams
{
    public string access_token { get; set; }
    public string token_type { get; set; }
    public string refresh_token { get; set; }
    public int expires_in { get; set; }
}
```

Эти параметры сохраняются в сессии для дальнейшего использования (в том числе для обновления ключа авторизации когда его срок будет истекать).

Вызов всех методов API осуществляется посредством WebRequest

В HomeController.cs за это отвечает метод

```
GetResult(string serviceUrl, string method = "GET", Dictionary<string, string> param = null)
```

Он возвращает строку – результат выполнения операции.

Проверяет доступность вызова API класс MyOAuth и его метод CheckAccessToken , который выбрасывает исключение в случае неуспешной проверки.

Результат проверки работы АПИ без авторизации изображен на рисунке 9

Тест лабораторная работа 2

Запрос: `http://localhost:60116/api/goods?page=1&size=5`

Ответ: [{"ID":1,"Cost":100.00,"Title":"Ластик "}, {"ID":2,"Cost":200.00,"Title":"Карандаш "}]

Запрос: `http://localhost:60116/api/goods/1`

Ответ: Недостаточно прав для выполнения операции

Запрос: `http://localhost:60116/api/deliveries?page=1&size=5`

Ответ: [{"ID":1,"Cost":500.00,"Title":"Курьер "}, {"ID":2,"Cost":1000.00,"Title":"Служба доставки "}]

Запрос: `http://localhost:60116/api/customers?page=1&size=5`

Ответ: Недостаточно прав для выполнения операции

Запрос: `http://localhost:60116/api/orders?page=1&size=5`

Ответ: Недостаточно прав для выполнения операции

Запрос: `http://localhost:60116/api/orders`

Ответ: Недостаточно прав для выполнения операции

Запрос: `http://localhost:60116/api/orders?orderId=0&goodId=1&customerId=1&deliveryId=1`

Ответ: Недостаточно прав для выполнения операции

Запрос: `http://localhost:60116/api/orders/0`

Ответ: Недостаточно прав для выполнения операции

Запрос: `http://localhost:60116/api/orders?orderId=0`

Ответ: Недостаточно прав для выполнения операции

[Авторизовать приложение по OAuth 2](#)

Рисунок 9 – Результат вызова монолитной API

После успешной авторизации по OAuth 2.0

Тест лабораторная работа 2

Запрос: `http://localhost:60116/api/goods?page=1&size=5`

Ответ: [{"ID":1,"Cost":100.00,"Title":"Ластик "},{ "ID":2,"Cost":200.00,"Title":"Карандаш "}]

Запрос: `http://localhost:60116/api/goods/1`

Ответ: "Название: Ластик ; Цена: "

Запрос: `http://localhost:60116/api/deliveries?page=1&size=5`

Ответ: [{"ID":1,"Cost":500.00,"Title":"Курьер "},{ "ID":2,"Cost":1000.00,"Title":"Служба доставки "}]

Запрос: `http://localhost:60116/api/customers?page=1&size=5`

Ответ: [{"ID":1,"Title":"Иванов ","Phone":"2-25-23 "},{ "ID":2,"Title":"Петров ","Phone":"32-23-33 "}]

Запрос: `http://localhost:60116/api/orders?page=1&size=5`

Ответ: [{"ID":1,"GoodId":null,"CustomerId":null,"DeliveryId":null,"Count":0},
{"ID":2,"GoodId":null,"CustomerId":null,"DeliveryId":null,"Count":0},
{"ID":3,"GoodId":null,"CustomerId":null,"DeliveryId":null,"Count":0},
{"ID":4,"GoodId":null,"CustomerId":null,"DeliveryId":null,"Count":0},
{"ID":5,"GoodId":null,"CustomerId":null,"DeliveryId":null,"Count":0}]

Рисунок 10 – Результат проверки монолитной API после успешной авторизации.