ПWS

ПОИТ-4,

лабораторная 6

4 часа

**Разработка и исследование WCF Data Service**

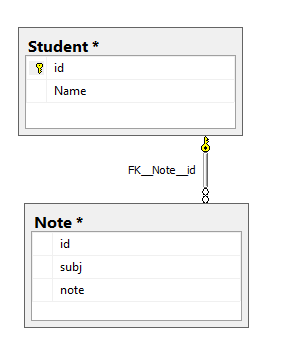
1. Используйте следующее подключение к серверу баз данных Microsoft SQL Server 2012.

**Server: W2012SQL2012B (172.16.193.89/172.16.0.89)**

**Login: WS**

**Password: ws**

1. Создайте базу данных с именем **WSXXX**, где XXX – инициалы студента. Можно создать 1 базу данных на троих студентов.
2. Создайте таблицы соответствующие следующей диаграмме.



|  |  |
| --- | --- |
| **Student** | **Список студентов** |
| id | Идентификатор студента |
| name | Фамилия, имя студента |
| **Note** | **Экзаменационные оценки студентов** |
| subj | дисциплина, изучаемая студентом |
| note | оценка полученная на экзамене |

1. Заполните таблицы базы данных (3 студента по 3 дисциплины с оценками).
2. Создайте приложение ASP.NET MVC. Создайте EDM-модель над базой данных **WSXXX.**
3. Создайте службу **WCF Data Service** с применением EDM-модели.
4. Продемонстрируйте работоспособность сервиса с помощью браузера (выбор списка, выбор по primary key, orderby, select, filter).
5. Разработайте программу клиента (в любой форме), демонстрирующего работу с сервисом.

**Для самостоятельного изучения**

1. Разработайте службу **WCF Data Service**  помощью которой можно добавлять/изменять данные в таблицу базы данных. Продемонстрируйте ее работоспособность.
2. <https://msdn.microsoft.com/ru-ru/library/cc668792(v=vs.110).aspx>

**1. ASP.NET MVC**

**ASP.NET MVC (Model-View-Controller)** — это архитектурный паттерн для создания веб-приложений.

**Ключевые понятия:**

* **Model:** Представляет данные приложения и логику их обработки.
* **View:** Отвечает за отображение данных (интерфейс пользователя).
* **Controller:** Обрабатывает пользовательские запросы, вызывает методы модели, передает данные в представление.

**Этапы создания приложения ASP.NET MVC:**

1. **Создание проекта ASP.NET MVC:**
   * В Visual Studio выберите шаблон "ASP.NET Web Application" с типом "MVC".
2. **Добавление модели данных:**
   * Используйте подход "Code-First" или "Database-First" для генерации моделей.
3. **Создание контроллера:**
   * Контроллеры обрабатывают запросы и возвращают представления или данные (например, JSON).
4. **Настройка маршрутов:**
   * В файле RouteConfig.cs определите маршруты (например, /Controller/Action/{id}).

**2. EDM-модель (Entity Data Model)**

**EDM (Entity Data Model)** — модель данных, описывающая структуру базы данных с использованием объектов (Entity Framework).

**Ключевые понятия:**

* **Entity Framework:** ORM (Object-Relational Mapping), позволяющая работать с базой данных через объекты .NET.
* **Database-First:** Метод генерации EDM из существующей базы данных.
* **EDM-модель:** Диаграмма объектов (Entities), их атрибутов и отношений.

**Шаги для создания EDM-модели:**

1. **Добавьте EDM в проект:**
   * Щелкните правой кнопкой на проекте > Add > New Item > Data > "ADO.NET Entity Data Model".
2. **Выберите подход:**
   * Для существующей базы данных выберите "Database-First".
3. **Подключение к базе данных:**
   * Укажите строку подключения и выберите таблицы для генерации моделей.
4. **Проверьте сгенерированную модель:**
   * В файле .edmx появится визуальная диаграмма таблиц, их связей и атрибутов.

**3. WCF Data Service**

**WCF Data Service** — это служба для предоставления данных через веб-протоколы, такие как OData.

**Ключевые понятия:**

* **WCF (Windows Communication Foundation):** Фреймворк для построения сервисов, которые работают через HTTP, SOAP, TCP и другие протоколы.
* **OData (Open Data Protocol):** Протокол для работы с данными через HTTP-запросы.

**Шаги для создания WCF Data Service:**

1. **Добавьте WCF Data Service в проект:**
   * Щелкните правой кнопкой на проекте > Add > New Item > "WCF Data Service".
2. **Настройте службу:**
   * Используйте EDM-модель как источник данных для сервиса.
   * Пример:

csharp

Копировать код

public class MyDataService : DataService<MyEntities>

{

public static void InitializeService(DataServiceConfiguration config)

{

config.SetEntitySetAccessRule("\*", EntitySetRights.All);

config.DataServiceBehavior.MaxProtocolVersion = DataServiceProtocolVersion.V3;

}

}

1. **Настройте Web.config:**
   * Проверьте строку подключения к базе данных.
2. **Проверьте службу:**
   * Перейдите по URL службы (например, http://localhost:port/MyDataService.svc).

**4. Тестирование WCF Data Service**

После настройки службы, убедитесь, что она поддерживает следующие операции OData:

**Примеры запросов:**

1. **Выбор списка:**
   * Запрос: http://localhost:port/MyDataService.svc/Students.
   * Описание: Возвращает всех студентов.
2. **Выбор по первичному ключу (Primary Key):**
   * Запрос: http://localhost:port/MyDataService.svc/Students(1).
   * Описание: Возвращает студента с ID = 1.
3. **Сортировка (OrderBy):**
   * Запрос: http://localhost:port/MyDataService.svc/Students?$orderby=name.
   * Описание: Возвращает студентов, отсортированных по имени.
4. **Выбор конкретных полей (Select):**
   * Запрос: http://localhost:port/MyDataService.svc/Students?$select=name.
   * Описание: Возвращает только имена студентов.
5. **Фильтрация (Filter):**
   * Запрос: http://localhost:port/MyDataService.svc/Students?$filter=id gt 2.
   * Описание: Возвращает студентов с ID больше 2.

**5. Клиентская часть**

Для тестирования сервиса необходимо разработать клиента, который будет отправлять HTTP-запросы и обрабатывать ответы.

**Ключевые варианты реализации клиента:**

1. **Консольное приложение (C#):**
   * Используйте библиотеку HttpClient для выполнения HTTP-запросов.
   * Пример:

csharp

Копировать код

using System;

using System.Net.Http;

using System.Threading.Tasks;

class Program

{

static async Task Main(string[] args)

{

var client = new HttpClient();

var response = await client.GetAsync("http://localhost:port/MyDataService.svc/Students");

var data = await response.Content.ReadAsStringAsync();

Console.WriteLine(data);

}

}

1. **Веб-приложение (JavaScript):**
   * Используйте fetch или библиотеку axios для выполнения запросов.
   * Пример:

javascript

Копировать код

async function fetchData() {

const response = await fetch('http://localhost:port/MyDataService.svc/Students');

const data = await response.json();

console.log(data);

}

fetchData();

1. **Пользовательский интерфейс (React, WinForms, или WPF):**
   * В интерфейсе отобразите данные, полученные из службы, в таблице или другом элементе управления.

**6. Дополнительные темы**

* **Разрешения в WCF:**
  + Убедитесь, что конфигурация SetEntitySetAccessRule разрешает необходимые операции (Read, Write, All).
* **Форматы данных:**
  + WCF Data Services поддерживает форматы JSON и XML.
* **Отладка:**
  + Проверьте ошибки при помощи инструмента Postman или консоли браузера.
* **Ограничения и оптимизация:**
  + Ограничьте размер возвращаемых данных с помощью параметров $top и $skip.

**Итог:** При сдаче лабораторной работы вы должны:

1. Продемонстрировать работоспособность приложения ASP.NET MVC.
2. Настроить и проверить WCF Data Service.
3. Показать правильность выполнения запросов OData.
4. Обеспечить взаимодействие клиента с сервисом.

В вашем проекте модели базы данных были **созданы вручную**. Это подтверждается наличием кода модели Note (и предположительно Student) в виде классов C# с атрибутами. Этот подход называется **Code-First** из Entity Framework. В отличие от EDM-модели (.edmx), где модели генерируются автоматически на основе существующей базы данных, здесь вы определяете модели вручную, и EF сопоставляет их с таблицами базы данных.

### ****Анализ структуры проекта****

#### **1. Подключение к базе данных**

В файле appsettings.json вы указали строку подключения:

json

Копировать код

"DefaultConnectionString": "Server=LEGION\\MSQLSERVER;Database=PWS\_Lab6;TrustServerCertificate=True;MultipleActiveResultSets=true;Integrated Security=true;"

Эта строка подключения используется в Program.cs через DI (Dependency Injection) для настройки StudentsDbContext:

csharp

Копировать код

builder.Services.AddDbContext<StudentsDbContext>(options =>

options.UseSqlServer(

builder.Configuration.GetConnectionString("DefaultConnectionString"),

sqlServerOptions => sqlServerOptions.EnableRetryOnFailure()

)

);

* **Класс StudentsDbContext:** Это ваш контекст данных, наследующий от DbContext. Он отвечает за взаимодействие с базой данных.

#### **2. Модели**

Ваши модели (Note и Student) определены вручную в виде классов C#.  
Пример модели Note:

csharp

Копировать код

public partial class Note

{

[DataMember]

[Key]

public int id { get; set; }

[Required]

[DataMember]

[StringLength(100)]

public string subj { get; set; }

[Column("note")]

[DataMember]

public int? note1 { get; set; }

[DataMember]

public int? studentId { get; set; }

[DataMember]

public virtual Student Student { get; set; }

}

##### **Особенности:**

1. **Атрибуты данных ([DataAnnotation]):**
   * [Key]: Определяет первичный ключ.
   * [Required]: Делает поле обязательным.
   * [StringLength(100)]: Ограничивает длину строки.
   * [Column("note")]: Указывает имя столбца в базе данных, если оно отличается от имени свойства.
   * [DataMember]: Используется для сериализации в WCF и OData.
2. **Свойства навигации:**
   * public virtual Student Student { get; set; }: Навигационное свойство для связи с таблицей Student.

##### **Как сопоставляются модели с таблицами базы данных?**

Entity Framework сопоставляет модели с таблицами автоматически, если имена классов (и их свойств) совпадают с именами таблиц (и их столбцов) в базе данных. Если имена различаются, вы можете уточнить их через атрибуты ([Table], [Column]) или Fluent API в OnModelCreating метода DbContext.

#### **3. OData и WCF**

Вы используете OData через Microsoft.AspNetCore.OData. Это означает, что вместо классического WCF Data Service ваш проект основан на OData-протоколе, встроенном в ASP.NET Core.

* OData модель определяется с помощью ODataConventionModelBuilder в Program.cs:

csharp

Копировать код

var s\_modelBuilder = new ODataConventionModelBuilder();

s\_modelBuilder.EntitySet<Student>("Students");

s\_modelBuilder.EntitySet<Note>("Notes");

* Запросы на основе OData позволяют:
  + **Выбор списка:** /odata/Students или /odata/Notes.
  + **Фильтрация:** /odata/Students?$filter=age gt 20.
  + **Сортировка:** /odata/Students?$orderby=name.

### ****Что делать, если база данных уже существовала?****

Если база данных существовала до начала работы, модели можно было сгенерировать автоматически с использованием подхода **Database-First**:

1. Установить Entity Framework Tools.
2. Выполнить команду:

bash

Копировать код

Scaffold-DbContext "строка\_подключения" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models

1. Это создаст C#-классы на основе структуры вашей базы данных.

### ****Резюме****

Ваши модели созданы вручную. Это стандартный **Code-First подход** из Entity Framework Core. Чтобы понять, соответствуют ли они структуре базы данных:

1. Проверьте таблицы Notes и Students в базе данных PWS\_Lab6.
2. Убедитесь, что имена столбцов и их типы совпадают с определениями в ваших моделях.