МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра программной инженерии

Отчет   
по лабораторной работе №6  
на тему: «Создание и развёртывание web-сервисов»  
по дисциплине «Проектирование и архитектура программных систем»

Выполнили: Марочкин М.А. Шифр: 170584   
 Шорин В.Д. Шифр: 171406  
 Щекотихин С.Е. Шифр: 170590  
ИПАИТ  
Направление: 09.03.04 «Программная инженерия»  
Группа: 71-ПГ  
Проверил:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Отметка о зачете:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата «\_\_\_\_» \_\_\_\_\_\_ 2019 г.

Орел, 2019 г.

## Задание на лабораторную работу

Разработать web-сервис, предоставляющий информацию о рейсах и наличии билетов на эти рейсы по запросам пользователей. Должна быть возможность просмотреть все рейсы по заданному маршруту, просмотреть наличие билетов, заказать билет на рейс.

Реализовать клиентское приложение для работы с данным сервисом. (Клиентское приложение и web-сервис могут быть реализованы на разных языках программирования). Для реализации web-сервиса рекомендуется использовать библиотеку WCF платформы .NET.

## Контрольные вопросы.

*1. Какие возможности предоставляет платформа .NET для создания web-сервисов?*

Создание веб-сервисов на платформе .NET – довольно простой процесс. Для этого не требуется знания о протоколах HTTP и SOAP или языке описания веб-сервисов WSDL. Среда разработки, используя библиотеку по работе с веб-сервисами самостоятельно создает прокси-классы веб-сервисов, скрывая, таким образом процесс взаимодействия с ними от разработчика.

*2. Опишите основные принципы работы платформы WCF.*

Принцип 1: границы установлены явно. Этот принцип подчеркивает тот факт, что функциональность службы WCF выражается через четко определенные интерфейсы (т.е. описания каждого члена, его параметров и возвращаемых значений). Единственный способ, которым внешний клиент может связаться со службой WCF, — через интерфейс, при этом оставаясь в блаженном неведении о деталях ее внутренней реализации.

Принцип 2: службы автономны. Говоря о службах, как об автономных сущностях, имеется в виду тот факт, что каждая служба WCF является (насколько возможно) отдельным "островом". Автономная служба должна быть независимой от проблем с версиями, развертыванием и установкой. Чтобы помочь в продвижении этого принципа, мы опять возвращаемся к ключевому аспекту программирования на основе интерфейсов. Как только интерфейс внедрен, он никогда не должен изменяться (или вы рискуете разрушить существующие клиенты). Когда требуется расширить функциональность службы WCF, просто напишите новый интерфейс, который моделирует необходимую функциональность.

Принцип 3: службы взаимодействуют через контракт, а не реализацию.Третий принцип — еще один побочный продукт программирования на основе интерфейсов — состоит в том, что реализация деталей службы WCF (на каком языке она написана, как именно выполняет свою работу, и т.п.) не касается вызывающего ее внешнего клиента. Клиенты WCF взаимодействуют со службами исключительно через их открытые интерфейсы. Более того, если члены службы представляют сложные специальные типы, они должны быть полностью детализированы в виде контракта данных, гарантируя, что клиенты смогут отобразить содержимое на определенную структуру данных.

Принцип 4: совместимость служб базируется на политике.Поскольку интерфейсы CLR предоставляют строго типизированные контракты всем клиентам WCF (и также могут быть использованы для генерации соответствующего документа WSDL на основе выбранной привязки), важно понимать на то, что интерфейсы и WSDL сами по себе недостаточно выразительны, чтобы детализировать аспекты того, что способна делать служба. Учитывая это, SOA позволяет определять политики, которые еще более проясняют семантику службы (например, ожидаемые требования безопасности, применяемые для общения со службой). Используя эти политики, можно отделять низкоуровневые синтаксические описания службы (предоставляемые интерфейсы) от семантических деталей их работы и способов их вызова.

*3. Что такое контракт в терминологии WCF?*

WCF содержат три вида контрактов:

1) сервисов для описания функциональных операций, реализованных сервисом. Внутри контракта сервиса имеются контракты об операциях, как отдельные операции сервиса, которые реализуют функции;

2) данных, определяющих формат данных, которыми будут обмениваться сервисы. Это относится как к запросу на сервис, так и к ответу сервиса. Если используются примитивные типы - int, string и др., то контракт не нужен, потому что .Net понимается как сериализация и дисериализация типов. В случае применения комплексных типов — Customers, Order и др., необходимо указать принцип сериалиализации и дисериализаии этих объектов;

3) сообщений, как тип контракта, который используется для того, чтобы получить контроль над заглавном SOAP пакета.

*4. Опишите механизм построения клиентского приложения на основе описания web-сервиса.*

Стандарты веб-сервисов обычно используются совместно и согласованно. После обнаружения WSDL в UDDI или другом месте генерируется SOAP-сообщение для отправки на удаленный сайт.

При предоставлении документа по адресу веб-сервиса программа использует XML-схему определенного типа, позволяющую преобразовать данные из ее входного источника и на основе того же WSDL-файла создать экземпляр XML-документа в формате, согласованном с целевым веб-сервисом. WSDL-файл используется для определения как входного, так и выходного преобразования данных.

SOAP-процессор отправляющего компьютера преобразует данные из собственного ("родного") формата в тип данных, предопределенный в соответствии с содержащейся в WSDL-файле XML-схемой на основе таблиц преобразования для текстов, значений с плавающей точкой и других данных. Таблицы преобразования "связывают" собственные типы данных с соответствующими конкретной XML-схеме. SOAP-процессор получающего компьютера выполняет обратное преобразование данных из типов XML-схемы в собственные типы данных.

Файлы описаний веб-сервисов обычно регистрируются с помощью URL. URL, повсеместно используемый в Сети, указывает на IP-адрес, соответствующий веб-ресурсу. Схемы веб-сервисов являются одной из форм веб-ресурса, они содержатся в доступных через Интернет файлах и к ним применим тот же механизм, что используется при загрузке HTML-файлов. Главное отличие между загрузкой HTML-файла и обращением к ресурсу веб-сервиса заключается в том, что веб-сервис оперирует XML-документами, а не HTML-документами и опирается на соответствующие технологии, такие как использование схем, преобразование, проверка подлинности, что и обеспечивает поддержку удаленного соединения приложений. Но способ, согласно которому схемы веб-сервисов публикуются и загружаются, одинаков: HTTP-операция по указанному URL.

Для проверки достоверности сообщений веб-сервисы используют XML-схемы. После получения документа реализация веб-сервиса сначала должна проанализировать XML-сообщение и удостовериться в корректности данных, выполнить проверку качества услуг, такую как проверку политики безопасности или соглашений торговых партнеров, а затем произвести последовательность связанных с данным документом коммерческих операций.

Веб-сервис, доступный по данному интернет-адресу, идентифицируется с помощью публичного WSDL-файла, который может быть загружен на отправляющий компьютер и использоваться при генерации сообщения.

Технологии веб-сервисов сформировались из основных структур. Этих технологий достаточно для построения, развертывания и публикации базовых веб-сервисов. На самом деле, необходим лишь базовый протокол SOAP. С момента появления веб-сервисов к ним все время добавляются другие технологии. Хотя для деловой связи, а также для создания "моста" с несовместимыми технологиями вполне достаточно базовых принципов, данная форма веб-взаимодействия тем не менее была одобрена очень быстро.

*5. Опишите принципы развертывания wcf-службы.*

Развертывание службы Windows Communication Foundation (WCF), который размещается в Internet Information Services (IIS) состоит из следующих задач:

* Убедитесь, что службы IIS, ASP.NET, WCF и компонента активации WCF службы правильно установлен и зарегистрирован.
* Создание нового приложения IIS или повторно использовать существующее приложение ASP.NET.
* Создание SVC-файла для службы WCF.
* Развертывание реализации службы в приложение IIS.
* Настройка службы WCF.

**Листинг**

**Сервер**

using System;using System.Collections;using System.ServiceModel;

namespace Microsoft.ServiceModel.Samples.BasicWebProgramming {

public static class Variables {

public static ArrayList flights = new ArrayList();

public static int ticketId = 1;

public static int flightId = 1;

}

public class Ticket { public int id; public float price;

public Ticket(float price) { id = Variables.ticketId; this.price = price; Variables.ticketId++; }

public int GetId() { return id; }

}

public class Flight {

public int id; public string startPlace; public string endPlace; public DateTime time;

public ArrayList tickets = new ArrayList(); public float price;

public Flight(float price, string startPlace, string endPlace) {

id = Variables.flightId;

Variables.flightId++;

this.startPlace = startPlace;

this.endPlace = endPlace;

this.price = price;

time = DateTime.Now;

tickets.Add(new Ticket(price)); tickets.Add(new Ticket(price));

tickets.Add(new Ticket(price)); tickets.Add(new Ticket(price));

}

public string Str() {

return String.Format("Id: {0}, Start-place: {1}, End-place: {2}, Time: {3}, NumberOfTickets: {4}, TicketPrice: {5}",

id, startPlace, endPlace, time, tickets.Count, this.price);

}

public int GetId() { return id; }

}

[ServiceContract]

public interface IService {

[OperationContract]

string EchoWithGet(string s);

[OperationContract]

string EchoWithPost(string s);

[OperationContract]

string GetFlights();

[OperationContract]

int BuyTicket(int flightId);

}

public class Service : IService {

public string EchoWithGet(string s) { return "You said " + s; }

public string EchoWithPost(string s) { return "You said " + s; }

public string GetFlights() {

Console.WriteLine("Client call GetFlights");

string returnStr = "";

foreach (Flight flight in Variables.flights) {

returnStr += flight.Str() + "\n";

}

return returnStr;

}

public int BuyTicket(int flightId) {

Console.WriteLine("Client call BuyTicket");

Ticket ticket = null;

foreach (Flight flight in Variables.flights) {

if (flight.GetId() == flightId) {

if (flight.tickets.Count != 0) {

ticket = (Ticket)flight.tickets[0];

flight.tickets.Remove(ticket);

Console.WriteLine(String.Format("Client buy ticket: {0}", ticket.id));

}

else { return 0; }

}

}

if (ticket == null) return 0;

else return ticket.id;

}

}

public class Program {

static void Main(string[] args) {

Variables.flights.Add(new Flight(10, "Орел", "Москва"));

Variables.flights.Add(new Flight(11, "Орел", "Белгород"));

Variables.flights.Add(new Flight(12, "Орел", "Липецк"));

Variables.flights.Add(new Flight(13, "Орел", "Брянск"));

Variables.flights.Add(new Flight(14, "Орел", "Курск"));

ServiceHost host = new ServiceHost(typeof(Service), new Uri("http://127.0.0.1:7777/"));

host.AddServiceEndpoint(typeof(IService), new BasicHttpBinding(), "");

Console.WriteLine("Server started work at :7777");

host.Open();

Console.WriteLine("Press <Enter> to terminate");

Console.ReadLine();

host.Close();

}

}

}

**Клиент**

using System;using System.Collections.Generic;using System.Net.Mime;using System.ServiceModel;

using System.ServiceModel.Description;using System.ServiceModel.Web;using System.Text;

namespace WCFClient {

[ServiceContract]

public interface IService {

[OperationContract] string EchoWinGet(string s);

[OperationContract] string EchoWithPost(string s);

[OperationContract] string GetFlights();

[OperationContract] int BuyTicket(int flightId);

}

internal class Program {

public static void Main(string[] args) {

Uri serverUri = new Uri("http://localhost:7777/");

EndpointAddress address = new EndpointAddress(serverUri);

BasicHttpBinding binding = new BasicHttpBinding();

ChannelFactory<IService> factory = new ChannelFactory<IService>(binding, address);

IService myService = factory.CreateChannel();

while (true) {

Console.WriteLine("1 - List of flights"); Console.WriteLine("2 - Buy ticket");

Console.WriteLine("0 - Exit"); Console.Write("Choose command: ");

string command = Console.ReadLine();

switch (command) {

case "1": { Console.WriteLine(myService.GetFlights()); break; }

case "2": {

Console.Write("Write flight id to buy ticket: ");

int flightId = Convert.ToInt32(Console.ReadLine());

int ticketId = myService.BuyTicket(flightId);

Console.WriteLine(String.Format("Your ticket id: {0}", ticketId));

break;

}

case "0": { Environment.Exit(0); break; }

default: break;

}

}

return;

}

}

}