Лабораторная работа № 8

Разработка HTTP-сервисов

1. Теоретические сведения

Веб-служба, веб-сервис (англ. web service) — идентифицируемая веб-адресом программная система со стандартизированными интерфейсами. Веб-службы могут взаимодействовать друг с другом и со сторонними приложениями посредством сообщений, основанных на определённых протоколах (XML, JSON и т. д.). Веб-служба является единицей модульности при использовании сервис-ориентированной архитектуры приложения. Windows Communication Foundation (WCF) позволяет создать службу, предоставляющую сетевую конечную точку. Сетевые конечные точки отправляют данные в виде XML-кода или JSON, без конверта SOAP. В этом разделе показано, как предоставить такую конечную точку.

Единственный способ защитить сетевую конечную точку заключается в том, чтобы предоставить ее через протокол HTTPS, используя механизм безопасности транспорта. Поскольку при использовании безопасности на основе сообщений сведения безопасности обычно помещаются в заголовки SOAP и сообщения, отправляемые другим конечным точкам (не SOAP), не содержат конвертов SOAP, негде разместить данные по безопасности и приходится полагаться на механизм безопасности транспорта.

REST - метод взаимодействия компонентов распределённого приложения в сети Интернет, при котором вызов удаленной процедуры представляет собой обычный HTTP-запрос (обычно GET или POST; такой запрос называют*REST-запрос*), а необходимые данные передаются в качестве параметров запроса. Этот способ является альтернативой более сложным методам, таким как SOAP,CORBA и RPC.

В широком смысле REST поддерживает концепцию построения распределённого приложения, при которой компоненты взаимодействуют наподобие взаимодействия клиентов и серверов во Всемирной паутине.

В качестве необходимых условий для построения распределённых REST-приложений Филдинг перечислил следующие:[2]

- Клиент-серверная архитектура.
- Сервер не обязан сохранять информацию о состоянии клиента.

- В каждом запросе клиента должно явно содержаться указание о возможности кэширования ответа и получения ответа из существующего кэша.
- Клиент может взаимодействовать не напрямую с сервером, а с произвольным количеством промежуточных узлов. При этом клиент может не знать о существовании промежуточных узлов, за исключением случаев передачи конфиденциальной информации.
- Унифицированный программный интерфейс сервера. Филдинг приводил URI в качестве примера формата запросов к серверу, а в качестве примера ответа сервера форматы HTML, XML и JSON, различаемые с использованием идентификаторов MIME.

Пример работы WCF (.NET)

Создание сетевой конечной точки

```
[ServiceContract]
public interface IService
{
    [OperationContract]
    [WebGet]
    string EchoWithGet(string s);

    [OperationContract]
    [WebInvoke]
    string EchoWithPost(string s);
}
```

По умолчанию атрибут WebInvokeAttribute сопоставляет с операцией вызовы POST. Впрочем, можно указать метод HTTP (например, HEAD, PUT или DELETE) для сопоставления с операцией, задав параметр "method=". У атрибута WebGetAttributeотсутствует параметр "method=", и он сопоставляет с операцией службы только вызовы GET.

Создайте объект WebServiceHost.

```
WebServiceHost host = new WebServiceHost(typeof(Service), new
Uri("http://localhost:8000/"));
```

Добавьте конечную точку ServiceEndpoint с поведением WebHttpBehavior.

```
erviceEndpoint ep = host.AddServiceEndpoint(typeof(IService), new WebHttpBinding(),
"");
```

Это можно предотвратить любым из следующих способов.

- Всегда указывать непустой код URI для конечной точки, не являющейся конечной точкой SOAP.
- Отключить страницу справки. Это можно сделать с помощью следующего кода.

```
ServiceDebugBehavior sdb = host.Description.Behaviors.Find<ServiceDebugBehavior>();
sdb.HttpHelpPageEnabled = false;

host.Open();
Console.WriteLine("Service is running");
Console.WriteLine("Press enter to quit...");
Console.ReadLine();
host.Close();
```

Вызов операций службы, сопоставленных с операцией GET, в Internet Explorer

Откройте браузер Internet Explorer, введите "**http://localhost:8000/EchoWithGet?s=Hello, world!**" и нажмите клавишу ВВОД. Этот URL-адрес содержит базовый адрес службы (http://localhost:8000/), относительный адрес конечной точки (""), вызываемую операцию службы (EchoWithGet), вопросительный знак и следующий за ним список именованных параметров, в качестве разделителя между которыми используется амперсанд (&).

Полный код примера

```
// Service.cs
using System;
using System.Collections.Generic;
using System.ServiceModel;
using System.ServiceModel.Description;
using System.ServiceModel.Web;
using System.Text;

namespace Microsoft.ServiceModel.Samples.BasicWebProgramming
{
    [ServiceContract]
    public interface IService
    {
        [OperationContract]
        [WebGet]
        string EchoWithGet(string s);

        [OperationContract]
        [WebInvoke]
        string EchoWithPost(string s);
    }
    public class Service : IService
    {
        public string EchoWithGet(string s)
        {
            return "You said " + s;
        }

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

```
    {
        static void Main(string[] args)
        {
            WebServiceHost host = new WebServiceHost(typeof(Service), new
Uri("http://localhost:8000/"));
            try
            {
                ServiceEndpoint ep = host.AddServiceEndpoint(typeof(IService), new
WebHttpBinding(), "");
                host.Open();
                using (ChannelFactory<IService> cf = new
ChannelFactory<IService>(new WebHttpBinding(), "http://localhost:8000"))
                {
                    cf.Endpoint.Behaviors.Add(new WebHttpBehavior());

                    IService channel = cf.CreateChannel();

                    string s;

                    Console.WriteLine("Calling EchoWithGet via HTTP GET: ");
                    s = channel.EchoWithGet("Hello, world");
                    Console.WriteLine("   Output: {0}", s);

                    Console.WriteLine("");
                    Console.WriteLine("This can also be accomplished by navigating
to");
                    Console.WriteLine("http://localhost:8000/EchoWithGet?s=Hello,
world!");
                    Console.WriteLine("in a web browser while this sample is
running.");

                    Console.WriteLine("");

                    Console.WriteLine("Calling EchoWithPost via HTTP POST: ");
                    s = channel.EchoWithPost("Hello, world");
                    Console.WriteLine("   Output: {0}", s);
                    Console.WriteLine("");
                }

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

                host.Close();
            }
            catch (CommunicationException cex)
            {
                Console.WriteLine("An exception occurred: {0}", cex.Message);
                host.Abort();
            }
        }
    }
}
```

Пример на JavaEE

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import javax.servlet.http.HttpServletResponse;
```

```
import javax.ws.rs.Consumes;
import javax.ws.rs.FormParam;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.Context;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Request;
import javax.ws.rs.core.UriInfo;

// Will map the resource to the URL todos
@Path("/todos")
public class TodosResource {

  // Allows to insert contextual objects into the class,
  // e.g. ServletContext, Request, Response, UriInfo
  @Context
  UriInfo uriInfo;
  @Context
  Request request;

  // Return the list of todos to the user in the browser
  @GET
  @Produces(MediaType.TEXT_XML)
  public List<Todo> getTodosBrowser() {
    List<Todo> todos = new ArrayList<Todo>();
    todos.addAll(TodoDao.instance.getModel().values());
    return todos;
  }

  // Return the list of todos for applications
  @GET
  @Produces({ MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON })
  public List<Todo> getTodos() {
    List<Todo> todos = new ArrayList<Todo>();
    todos.addAll(TodoDao.instance.getModel().values());
    return todos;
  }

  // retuns the number of todos
  // Use http://localhost:8080/com.vogella.jersey.todo/rest/todos/count
  // to get the total number of records
  @GET
  @Path("count")
  @Produces(MediaType.TEXT_PLAIN)
  public String getCount() {
    int count = TodoDao.instance.getModel().size();
    return String.valueOf(count);
  }

  @POST
  @Produces(MediaType.TEXT_HTML)
  @Consumes(MediaType.APPLICATION_FORM_URLENCODED)
  public void newTodo(@FormParam("id") String id,
```

```
      @FormParam("summary") String summary,
      @FormParam("description") String description,
      @Context HttpServletResponse servletResponse) throws IOException {
    Todo todo = new Todo(id, summary);
    if (description != null) {
      todo.setDescription(description);
    }
    TodoDao.instance.getModel().put(id, todo);

    servletResponse.sendRedirect("../create_todo.html");
  }

  // Defines that the next path parameter after todos is
  // treated as a parameter and passed to the TodoResources
  // Allows to type
http://localhost:8080/com.vogella.jersey.todo/rest/todos/1
  // 1 will be treaded as parameter todo and passed to TodoResource
  @Path("{todo}")
  public TodoResource getTodo(@PathParam("todo") String id) {
    return new TodoResource(uriInfo, request, id);
  }

}
```

Пример клиентского кода AngularJS (организация CRUD для базы fireBase)

```
1.  angular.module('project', ['ngRoute', 'firebase'])
2.
3.  .value('fbURL', 'https://ng-projects-list.firebaseio.com/')
4.  .service('fbRef', function(fbURL) {
5.    return new Firebase(fbURL)
6.  })
7.  .service('fbAuth', function($q, $firebase, $firebaseAuth, fbRef) {
8.    var auth;
9.    return function () {
10.       if (auth) return $q.when(auth);
11.       var authObj = $firebaseAuth(fbRef);
12.       if (authObj.$getAuth()) {
13.         return $q.when(auth = authObj.$getAuth());
14.       }
15.       var deferred = $q.defer();
16.       authObj.$authAnonymously().then(function(authData) {
17.           auth = authData;
18.           deferred.resolve(authData);
19.       });
20.       return deferred.promise;
21.   }
```

```
})

.service('Projects', function($q, $firebase, fbRef, fbAuth) {
  var self = this;
  this.fetch = function () {
    if (this.projects) return $q.when(this.projects);
    return fbAuth().then(function(auth) {
      var deferred = $q.defer();
      var ref = fbRef.child('projects-fresh/' + auth.auth.uid);
      var $projects = $firebase(ref);
      ref.on('value', function(snapshot) {
        if (snapshot.val() === null) {
          $projects.$set(window.projectsArray);
        }
        self.projects = $projects.$asArray();
        deferred.resolve(self.projects);
      });

      //Remove projects list when no longer needed.
      ref.onDisconnect().remove();
      return deferred.promise;
    });
  };
})

.config(function($routeProvider) {
  var resolveProjects = {
    projects: function (Projects) {
      return Projects.fetch();
    }
  };

  $routeProvider
    .when('/', {
      controller:'ProjectListController as projectList',
      templateUrl:'list.html',
      resolve: resolveProjects
    })
    .when('/edit/:projectId', {
      controller:'EditProjectController as editProject',
      templateUrl:'detail.html',
      resolve: resolveProjects
    })
    .when('/new', {
      controller:'NewProjectController as editProject',
```

```
67.       templateUrl:'detail.html',
68.       resolve: resolveProjects
69.     })
70.     .otherwise({
71.       redirectTo:'/'
72.     });
73. })
74.
75. .controller('ProjectListController', function(projects) {
76.   var projectList = this;
77.   projectList.projects = projects;
78. })
79.
80. .controller('NewProjectController', function($location, projects) {
81.   var editProject = this;
82.   editProject.save = function() {
83.       projects.$add(editProject.project).then(function(data) {
84.           $location.path('/');
85.       });
86.   };
87. })
88.
89. .controller('EditProjectController',
90.   function($location, $routeParams, projects) {
91.     var editProject = this;
92.     var projectId = $routeParams.projectId,
93.         projectIndex;
94.
95.     editProject.projects = projects;
96.     projectIndex = editProject.projects.$indexFor(projectId);
97.     editProject.project = editProject.projects[projectIndex];
98.
99.     editProject.destroy = function() {
100.
    editProject.projects.$remove(editProject.project).then(function(data) {
101.               $location.path('/');
102.           });
103.       };
104.
105.       editProject.save = function() {
106.
    editProject.projects.$save(editProject.project).then(function(data) {
107.               $location.path('/');
108.           });
109.       };
```

```
110.    });
```

2. Задание на лабораторную работу.

Разработать HTTP-сервис по выбранной теме - реализовать CRUD (Create Read Update Delete) операции.
Обязательно ооолжны использоваться запросы в стиле REST:

GET - для получения данных
POST - для создания ресурса (сохранения данных)
PUT - для обновления ресурса (обновление данных)
DELETE - для удаления ресурса (удаления строк)

(рекомендуется использовать .Net, Java или Python, можно использовать другие языки программирования по согласованию с преподавателем). Приложение должно поставлять и получать данные в одном из следующих форматов: XML, JSON, YAML.

На сервере использовать любую БД для хранения данных, рекомендуется использовать MySQL, PostgreSQL, MS SQL Server. Приветствуется использование не реляционных СУБД (например, MongoDB или Casandra).

Клиентское приложение: это html-страница, где запросы (вызовы сервиса) осуществляются с использованием AJAX (рекомендуется использовать jQuery). Для студентов претендующих на оценку 9-10 следует использовать любой из перечисленных JavaScript framework: AngularJS, Backbone, KnockoutJS, AmberJS или разработать мобильный клиент для Android, iOS или Windows Phone.

Обязательные компоненты отчета:

- Краткое описание алгоритма
- Листинги серверного кода, отправки сообщений клиенту
- Листинги клиентского кода с обработкой событий с сервера
- Верификация

Отчѐт должен содержать:

- Краткое описание алгоритма
- Листинги серверного кода, отправки сообщений клиенту
- Листинги клиентского кода с обработкой событий с сервера
- Верификация

Варианты заданий

№	Название класса	Поля
1	Служащий	имя, возраст, стаж, место работы
2	Библиотека	автор, название, стоимость, год издания, отрасль знаний
3	Экзамен	ФИО студента, дата, оценка, группа
4	Адрес	название улицы, номер дома, характеристика дома (кирпичный, панельный, блочный, блочнокирпичный дом и т.д.)
5	Цех	название цеха, ФИО начальника, количество работающих, название- предприятия
6	Квитанция	номер, название операции (н-р, оплата коммульных услуг, за обучение, за телефон), дата, сумма
7	Товар	название товара, количество, стоимость, производитель
8	Кадры	ФИО рабочего, номер цеха, разряд, специальность
9	Автомобиль	марка, мощность, стоимость, цвет
10	Автобусный рейс	название пункта назначения; номер маршрута; время отправления, время прибытия
11	Страна	название страны; форма правления; площадь, континент
12	Животное	название, класс, средний вес, среда обитания (воздушное пространство, водное пространство, суша)
13	Корабль	название корабля, тип,водоизмещение, страна-производитель
14	Спортсмен	ФИО, возраст, количество побед, вид спорта
15	Зарплата	Ф.И.О., тарифная категория работника, объем выполненной работы, стоимость единицы

		продукции
16	Записная книжка	ФИО контактного лица, мероприятие, дата и время мероприятия
17	Организация	название организации, тип (госучреждение, ОАО, частное и т.д.), адрес, телефон, количество сотрудников
18	Топливная база	вид топлива, название емкости для хранения топлива и поле для хранения количества прихода/расхода топлива
19	Оборудование кафедры	кафедра, материально ответственное лицо, название оборудование, количество, стоимость единицы оборудования
20	Частный детектив	фамилия субъекта; вид правонарушения, дата правонарушения, сумма вознаграждения
21	Дневник метеонаблюдений	населенный пункт, дата, температура, давление, облачность (ясно, слабая, сильная, дождь)
22	Справочник по транзисторам	тип транзистора, напряжение питания, допустимый ток, стоимость, страна-производитель
23	Изделие	название изделия, шифр, количество, производитель
24	Справочник автоинспектора	номер автомобиля, марка, мощность, год выпуска, пробег в км на дату техосмотра, дата прохождения техосмотра
25	Персона	имя, пол, возраст, статус (служащий, рабочий, студент, безработный)
26	Кафедра	название кафедры, ФИО заведующего кафедры, число сотрудников, название факультета
27	Библиотечный каталог	УДК (универсальный десятичный код), отрасль знаний, Ф.И.О. автора, тип издания, стоимость, количество

28	Поезда	номер поезда, тип (пассажирский, скорый, эспресс) пункт назначения, пункт отправления, время прихода, время отправления
29	Справочник автоинспектора	номер автомобиля, марка, мощность, год выпуска, пробег в км на дату техосмотра, дата прохождения техосмотра, цвет
30	Памятка дачнику-овощеводу	Вид овоща (картофель, помидор, огурец, перец, редис, салат), сорт, номинальная урожайность в кг, рекомендуемая дата посадки, рекомендуемая дата уборки
31	Преподаватель	название дисциплины, объем лекций в часах, объем лабораторных занятий в часах, вид контроля (зачет, экзамен, зачет-экзамен, тест), ФИО преподавателя
32	Справочник по оборудованию	наименование оборудования, страна-изготовитель, стоимость, год изготовления
33	Меломан	название группы, Ф.И.О. руководителя, название альбома, тираж дисков, страна-группы
34	Великие даты	дата, страна, вид события (война, революция и т. д.), примерное число жертв
35	Альпинист	название вершины, страна расположения, высота, категория сложности
36	Живая планета	наименование животного, вид (млекопитающие, птицы, рыбы и т. д.), вес (средний) в кг, размер популяции
37	Медицинская карта	Ф.И.О. пациента, год рождения, рост, вес, диагноз, дата осмотра
38	Вкладчики банка	ФИО вкладчика, вид валюты, дата денежной операции (внесение на счет, снятие со счета), количество денежных средств внесения на счет или снятия
39	Журнал инспектора энергонадзора	Ф.И.О. инспектора, дата посещения предприятия, название предприятия, сумма штрафных санкций

40	Журнал ремонта оборудования	наименование оборудования, дата ремонта, Ф.И.О. слесаря-ремонтника, сложность ремонта, стоимость материалов, применяемых в ремонте
41	Журнал учета аварий на предприятии	дата аварии, место аварии (цех, участок, оборудование), сумма ущерба для предприятия
42	Журнал учета грузов на проходной	дата прохождения партии грузов, наименование груза, общая стоимость груза, область отправки (гомельская область, минская и т.д.)
43	Магазин	дата получения товара, название товара, общая стоимость товара, единица измерения товара
44	Агроном	название культуры, название удобрения, стоимость единицы удобрения, единица измерения удобрения
46	Кинологический центр	порода собаки, минимальный рост по экстерьеру, максимальный рост по экстерьеру, кличка собаки, фактический рост
47	Студент	ФИО, название группы, размер стипендии, категория (сельский, городской, из ближнего зарубежья, из дальнего зарубежья)
50	Музыкальный альбом	Название исполнителя, Жанр, Количество песен в альбоме, дата выхода.
52	Платеж	ИНН плательщика, Название организации, Адрес, Сумма платежа, Дата платеже
53	Поставщики	юридическое название поставщика, регистрационный номер, номер телефона, юридический адрес поставщика
54	Заказы на доставку	Желаемое врем доставки, вес груза, габариты груза, адрес доставки, bool значение указывающее хрупкий груз или нет
55	Учет деталей автомобилей	Название детали, Марка автомобиля, Описание детали, Дата производства, Производитель
56	Мед. препарат	Наименование, Производитель, Нужен ли рецепт, рекомендуемая цена, форма поставки (таблетки, капсулы, уколы)
57	Продукты	Торговое название, фирма производитель, вес (емкость), дата изготовления, срок годности

58	Служащие офиса	ФИО, год рождения, телефон, email, образование, специальность, должность, дата приема на работу
59	Справочник компьютерных игр	название, жанр, компания разработчик, год выпуска, системные требования.
60	Справочник сериалов	название, жанр, режиссер, год выпуска, количество серий, поисковый тэг