Лабораторная работа №2. Проектирование и создание Web-сервисов

Цель: научиться определять функциональные границы Web-сервисов, реализовывать, развёртывать и использовать их.

Теоретические сведения.

В настоящее время SOA и Web-сервисы получили достаточно широкое распространение, и применяются в самых разных качествах: от простого предоставления справочных данных в Сети, например, данных о прилете самолетов, курсов валют и драгоценных металлов, до работы с кредитными карточками и переводов текста online. Еще больше существует корпоративных Web-сервисов, которые используются при решении самых разных производственных задач.

Широкому распространению Web-сервисов немало способствовала платформа Microsoft .NET. Дело в том, что создать Web-сервис в той же Microsoft Visual Studio .NET очень просто. Достаточно написать свой класс, унаследованный от класса System.Web.Services.WebService и объявить его методы как Web-методы. Компилируйте, переносите на рабочий сервер и ваш новый Web-сервис готов к работе.

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Web;
using System.Web.Services;

namespace WebServicesExample
{
	public class nw : System.Web.Services.WebService
	{
		public nw()
		{
			InitializeComponent();
		}
		#region Component Designer generated code

		//Required by the Web Services Designer
		private IContainer components = null;

		private void InitializeComponent()
		{
		}
		protected override void Dispose( bool disposing )
```

```
        {
            if(disposing && components != null)
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #endregion

        // WEB SERVICE EXAMPLE
        [WebMethod]
        public string HelloWorld()
        {
            return "Hello World";
        }
    }
}
```

Так же просто написать и клиент для любого Web-сервиса. В состав средств разработки любых платформ входит утилита wsdl.exe. Достаточно ей указать адрес WSDL документа, желаемый язык разработки (C#, VB.NET) и она сгенерирует вам код класса, который является прокси-классом для указанного Web-сервиса, то есть его клиентом. Вы получаете готовый к использованию класс с тем же набором методов, что и Web-сервис. Любое обращение к этим методам автоматически транслируется Web-сервису и ответ Web-сервиса возвращается как результат. Другой способ — добавление в проект ссылки на сервис. В этом случае среда разработки сама выполнит генерацию прокси-класса.

Аналоги wsdl.exe присутствуют, как и было сказано, и в составе Java SE/EE SDK. Однако, для данной платформы приходится использовать две утилилы, которые покрывают функциональнось wsdl.exe: это утилиты wsimport и wsgen. Для создания клиента как раз применяется wsimport.

Web-сервисы .NET могут отвечать не только на запросы SOAP, но и на обычные запросы POST и GET. То же самое касается WSDL сервисов, написанных для других платформ: параметр запроса wsdl, переданный развёрнутой службе позволит получить WSDL-документ, который её описывает.

Написание любого Web-сервиса начинается с изучения WSDL документа – описания самого Web-сервиса, его методов, параметров методов и типов данных.

Создание службы может производиться по одному из сценариев:

- code first;
- contract first.

В первом случае создается код сервиса, а метаданные генерируются автоматически. Во втором — сначала описывается контракт службы в виде WSDL-документа, затем по нему генерируется код без реализации.

WSDL(Web Services Description Language) - это XML-ориентированный язык, предназначенный для определения web-сервисов и доступа к ним.

Документ WSDL является XML-документом, описывающим web-сервис. Он определяет расположение сервиса и операции (или методы), предоставляемые им.

Структура документа WSDL

Документ WSDL - это всего лишь простой документ XML, который содержит набор выражений, определяющих web-сервис.

В документе WSDL определяется web-сервис с помощью следующих основных элементов:

Элемент	**Определяет**
<portType>	Методы, предоставляемые web-сервисом
<message>	Сообщения, используемые web-сервисом
<types>	Типы данных, используемые web-сервисом
<binding>	Протоколы связи, используемые web-сервисом

Основная структура документа WSDL выглядит следующим образом:

```
<definitions>
<types>
  определение типов.....
</types>

<message>
  определение сообщения....
</message>

<portType>
  определение порта.....
</portType>

<binding>
  определение связей.....
</binding>

</definitions>
```

Документ WSDL может также содержать другие элементы, например, элементы расширения и элемент service, который позволяет объединить вместе в одном отдельном документе WSDL определения нескольких web-сервисов.

Порты WSDL

Элемент **<portType>** является наиболее важным элементом в WSDL.

Он определяет сам web-сервис, предоставляемые им операции и используемые сообщения. Порт определяет точку монтирования к web-сервису. Его можно сравнить с библиотекой функций (модулем, классом) в традиционном языке программирования, а каждую операцию можно сравнить с функцией в традиционном языке программирования.

Элемент <portType> можно сравнить с библиотекой функций (модулем, классом) в традиционном языке программирования.

Сообщения WSDL

Элемент **<message>** определяет элементы данных операции.

Каждое сообщение может содержать одну или несколько частей. Эти части можно сравнить с параметрами вызываемых функций в традиционном языке программирования.

Типы WSDL

Элемент **<types>** определяет тип данных, используемых web-сервисом.

Для максимальной платформо-независимости WSDL использует синтаксис XML Schema для определения типов данных.

Связи WSDL

Элемент **<binding>** определяет формат сообщения и детали протокола для каждого порта.

Пример WSDL

Это простейшая фрагмент WSDL-документа:

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>
<portType name="glossaryTerms">
 <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
 </operation>
</portType>
```

В этом примере элемент **portType** определяет "glossaryTerms" как имя **порта**, и "getTerm" как имя **операции**.

Операция "getTerm" имеет **входящее сообщение**, называемое "getTermRequest",

и **исходящее сообщение** --- "getTermResponse".
Элементы **message** определяют **части** каждого сообщения и ассоциированные типы данных.
Если сравнивать с традиционным программированием, то glossaryTerms --- библиотека функций, а "getTerm" --- функция с параметром "getTermRequest", которая возвращает getTermResponse после выполнения.
Порты WSDL описывают интерфейс (список допустимых операций) работы с web-сервисом.

Типы Операций

Запрос-ответ (request-response) --- самый распространенный тип операций. В WSDL определены такие типы операций:

Тип	**Описание**
Однонаправленный (One-way)	Операция может принимать сообщение, но не будет возвращать ответ
Запрос-ответ (Request-response)	Операция может принимать запрос и возвратит ответ
Вопрос-ответ (Solicit-response)	Операция может послать запрос и будет ждать ответ

Однонаправленная (One-Way) Операция

Пример однонаправленной операции:

```
<message name="newTermValues">
  <part name="term" type="xs:string"/>
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="setTerm">
    <input name="newTerm" message="newTermValues"/>
  </operation>
</portType >
```

В этом примере порт "glossaryTerms" определяет однонаправленную операцию под названием "setTerm".
Операция "SetTerm" позволяет ввод новых словарных термов с помощью сообщения "newTermValues" со входными параметрами "term" и "value". Однако, данная операция не предусматривает какого-либо выходного сообщения.

Операция типа "Запрос-Ответ"

Пример операции типа "запрос-ответ":

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>
```

```
<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>
<portType name="glossaryTerms">
 <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
 </operation>
</portType>
```

В этом примере порт "glossaryTerms" определяет операцию типа "запрос-ответ" с именем "getTerm".
Операция "getTerm" принимает на вход сообщение с именем "getTermRequest" и параметром "term" и возвращает сообщение с именем "getTermResponse" и параметром "value".
Связи WSDL определяют формат сообщения и подробности протокола для каждого порта.

Связь с SOAP

Пример операции запроса-ответа:

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>

<binding type="glossaryTerms" name="b1">
<soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http" />
 <operation>
   <soap:operation
    soapAction="http://example.com/getTerm"/>
   <input>
     <soap:body use="literal"/>
   </input>
   <output>
     <soap:body use="literal"/>
   </output>
 </operation>
</binding>
```

Элемент **binding** имеет два атрибута --- атрибут name и атрибут type.
Атрибут name определяет название (можно использовать любое) связи, а атрибут type --- точку монтирования для связи, в данном случае это порт "golossaryTerms".
Элемент **soap:binding** имеет два атрибута --- атрибут style и атрибут transport.
Элемент style может быть либо "rpc", либо "document". В данном случае используется значение document. Атрибут transport определяет используемый SOAP протокол. В данном случае это HTTP.
Элемент **operation** описывает все операции, с которыми порт может работать.
Для каждой операции должно быть определено в соответствие действие SOAP. Также необходимо указать кодировку ввода и вывода. В этом случае используется "literal".

Развертывание сервиса в MSVS.

Microsoft Visual Studio предоставляет возможность работы со службой при отладке без её развёртывания. Для этого запускается легковесный Web-сервер, в котором автоматически развернута только что созданная служба. После окончания разработки службу всё же будет необходимо развернуть в Web-контейнере. Таким контейнером может выступать платформа Microsoft IIS (Internet Information Services). Для развертывания службы можно использовать сервер по умолчанию, но лучше для служб иметь отдельный виртуальный сервер или хотя бы каталог.

Таким образом, для развертывания службы в Web-контейнере требуется выполнить следующие шаги:

1. Создать виртуальный сервер или хотя бы виртуальный каталог в IIS
2. Создать в каталоге папку bin
3. Поместить службу в виртуальный каталог, а соответствующую ей сборку .NET - в каталог bin.

Развёртывание сервиса в Java

Аналогично, для развёртывания SOAP-сервиса в Java нам понадобится либо контейнер сервлетов, либо хост-приложение. Во втором случае развёртывание происходит за счёт запуска хост-приложения, а в первом понадобятся следующие шаги:

1. Создать дескриптор службы, файл sun-jaxws.xml, и описать в нём конечные точки вашей службы.
2. Добавить сервлет и слушатель в конфигурацию Web-приложения (файл web.xml)
3. Развернуть WAR-файл с приложением в контейнере сервлетов.

Порядок выполнения работы

В соответствии с индивидуальным заданием

1. Определить функциональные границы веб-сервиса.

2. Создать WSDL-описание сервиса и задокументировать его.
   - При возникновении трудностей допускается использование подхода code first.
3. Реализовать сервис на выбранной платформе (допускается использование .NET, Java или Node.js) и выполнить его развертывание в контейнере.
4. Реализовать клиентское приложение и с помощью вызовов созданного сервиса из его кода продемонстрировать работу методов (вызовите объявленные методы, продемонстрируйте получение результатов).

Индивидуальное задание

1. В соответствии с темой заданиея (согласовывается с преподавателем) выполнить описание бизнес-процесса в виде одной Activity-диаграммы или их набора.
2. Определите функциональные границы двух-трех сервисов, которые могли бы быть использованы для представления первых шагов бизнес-процесса.
3. Выделите функциональность для организации первого из сервисов таким образом, чтобы можно было создать операции: однонаправленные и запрос-ответ.
4. Реализуйте первую службу (первый шаг процесса).
5. Создайте клиентское приложение, с помощью которого продемонстрируйте работу запросов. Основные требования к клиентскому приложению:
   - приложение должно использовать разработанную веб-службу или сервис
   - приложение должно быть оснащено графическим пользовательским интерфейсом, который должен обеспечивать демонстрацию работы всех функций, а также ввод пользователем всех основных параметров и отображение результатов