

МІНІСТЕРСТВО ОСВІТИ Й НАУКИ, МОЛОДІ ТА СОРТУ УКРАЇНИ
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторних робіт з дисципліни
“ Архітектури та технології WEB служб ”
для студентів напрямку 6.050103
“Програмна інженерія”
денної форми навчання

2016

–

Методичні вказівки до лабораторних робіт з дисципліни “Архітектури та технології WEB додатків” для студентів напрямку 6.050103 “Програмна інженерія” денної форми навчання /Уклад.: Степаненко О.О. – Запоріжжя: ЗНТУ, 2016. – 35 с.

Укладачі: О.О. Степаненко, доцент, к.т.н.,

Рецензент: С.К. Корнієнко, доцент, к.т.н.

Відповідальний
за випуск: С.О. Субботін, проф., д.т.н.

Затверджено
на засіданні кафедри
"Програмні засоби"

Протокол № 1 від 29.08.2016 р.

ЗМІСТ

Лабораторна робота 1. Тема: «Створення web – служб»	4
Основні теоретичні положення.....	4
Засоби розробки.....	10
Порядок виконання роботи	11
Приклад виконання	12
Складання звіту лабораторної роботи.....	20
Контрольні питання	20
Завдання на групу.....	20
Лабораторна робота 2. Тема: «Перевірка коментарів у блозі, інтегрована з базою даних, з використанням XML»	21
Основні теоретичні положення.....	21
Простий приклад використання JDBC 4.0 при роботі з XML	23
Необхідні програмні продукти й технології.....	33
Складання звіту лабораторної роботи.....	33
Контрольні питання	34
Список літератури	35

Лабораторна робота 1. Тема: «Створення web – служб»

1 Мета роботи

Мета роботи – створення web –служби для використання її додатками й закріпити навички створення розподілених додатків.

2 Основні теоретичні положення

Генезис Web

Споконвічно World Wide Web була мережею документів. WebbСервери спілкувалися із клієнтами по протоколу HTTP (Hypertext Transfer Protocol) і пересилали інформацію у формі гіпертекстових документів, створених засобами мови HTML (Hypertext Markup Language). Такі документи відображалися в браузері і містили посилання на інші документи. Цього було цілком достатньо для задоволення запитів творців Web — учених, яким був необхідний засіб для обміну різними документами: результатами досліджень, лабораторними протоколами, звітами й т.п. Таким чином, Web з'явився як документноорієнтована мережа.

Єдиновладдя вчених тривало недовго. Незабаром стали з'являтися комерційні Webсайти, що містять документи, що рекламували продукти й сервіси тих або інших компаній. Такі сайти дотепер становлять основне наповнення Web, але з недавнього часу все більша кількість комерційних сайтів підтримує транзакції із клієнтами, перетворюючись у такий спосіб у сайти електронної комерції. Багато комерційних сайтів будуються не на основі звичайних WebСерверів, а за допомогою багатоланкової архітектури, що припускає використання серверів додатків.

Основною відмінністю звичайних WebСерверів від серверів додатків є те, що останні не просто повертають документ, а можуть обробляти користувацькі запити й містять код, що реалізує бізнесов-логіку. Звичайно сервери додатків генерують документи динамічно, залежно від зазначених користувачем параметрів. Також слід зазначити, що застосування серверів додатків дозволяє створювати масштабовані рішення, здатні одночасно обслуговувати велика кількість транзакцій.

Поява різноманітних мобільних пристроїв привела до того, що замість традиційних браузерів багато комерційні WebДодатка тепер крім протоколу HTTP підтримують і протокол WAP (Wireless Access Protocol) і здатні повертати інформацію не тільки в стандарті HTML, але й у стандарті, що задовольняє користувачів, що звертаються до сервісів через мобільні пристрої, — WML (Wireless Markup Language).

Природно, електронна комерція не може обмежуватися простою обробкою транзакцій — наступним логічним кроком у розвитку Web стала інтеграція бізнесопроцесів різних компаній. У такий спосіб на світло з'являється сервіс-орієнтований Web. У його основі лежать дві відносно нові технології — SOAP (Simple Object Access Protocol) і XML (Extensible Markup

Language). Згідно із цим сценарієм Web складається з набору серверів додатків, що обмінюються інформацією у форматі XML по протоколу SOAP.

Основою сервіс-орієнтованого Web є Web-сервіс — набір логічно зв'язаних функцій, які можуть бути програмно викликані через Internet. Інформація про те, які функції надає даний Web-сервіс, утримується в документі WSDL (Web Service Description Language), а для пошуку існуючих Web-сервісів передбачається використання спеціальних реєстрів, сумісних зі специфікацією UDDI (Universal Description, Discovery and Integration).

Сервіс-орієнтований Web

Web-сервіси (служби) являють собою технологію інтеграції додатків, яка може використовуватися в Internet. Як приклад можливого використання Web-сервісів розглянемо планування подорожей. Звичайно в такій ситуації потрібні: замовлення квитків на літак, бронювання місць у готелі, оренда автомобіля й, можливо, використання послуг місцевих компаній, що організують екскурсії.

Традиційно використовуючи Internet, вам доведеться відвідати сервер авіакомпанії, сервер готелю або мережі готелів, сервер компанії по оренді автомобілів і сервер компанії, що спеціалізується на організації екскурсій в обраному вами місці. Усі ці дії можуть зайняти досить багато часу, перш ніж ви досягнете мети. І при цьому жодна із задіяних вами компаній не буде знати, які ваші плани, а отже, не зможе оптимізувати ваш час. Проблема полягає в тому, що компанії, що спеціалізуються на окремих областях — авіа, готельної, прокату автомобілів і т.п., у більшості випадків замкнені самі на себе й використовують власні засоби зберігання й вистави даних.

Більш зручно було б запустити додаток, яке б прийняв від вас необхідну інформацію й виконало всі рутинні дії — замовлення квитків, бронювання готелю й т.п. — автоматично, без вашої участі. Щоб це стало можливим, ви повинні використовувати Web-сервіси. Давайте розглянемо, що зміниться в цьому випадку.

Припустимо, авіакомпанія надає Web-сервіс, що дозволяє додаткам одержувати список рейсів між двома містами для заданої дати. У цьому випадку більше не потрібно звертатися до WebуВузлу авіакомпанії й указувати різні критерії пошуку — уся необхідна інформація доступна у вигляді єдиного Xml-документа. Тепер припустимо, що авіакомпанія, готель і агентство по прокату автомобілів надають Web-сервіси, що дозволяють програмно здобувати квитки, бронювати номери й орендувати автомобілі. У цьому випадку можна об'єднати виклики всіх цих сервісів у єдиний додаток, який зможе виконати всю рутинну роботу без участі користувача.

Втім, на цьому функціональність нового класу ВебДодатків не закінчується. Наш додаток може, наприклад, періодично звертатися до Web-сервісу авіакомпанії для визначення статусу рейсу й у випадку його затримки сповіщати сервіси готелю, служби прокату й т.п. для продовження бронювання.

Крім очевидного підвищення якості обслуговування клієнтів використання Web-сервісів має безліч інших переваг. Наприклад, якщо агентство прокату автомобілів знає, що ваш рейс затримується, воно може більш гнучко розпорядитися своїми автомобілями. У міру зростання числа

Web-сервісів ми зможемо побачити більш комплексні приклади їх використання. Однак слід зазначити, що впровадження концепції Web-сервісів вимагає не тільки перегляду багатьох бізнес-правил і схем взаємодії між галузями й секторами тієї або іншої галузі, але й підвищення безпеки обміну даними.

Розглянувши практичне застосування Web-сервісів, звернемося до стандартів, що лежать в основі цих сервісів.

Стандарти для Web-сервісів

Як ми вже знаємо, в основі WebСервісів лежать Internet-стандарти. Ці стандарти визначають протоколи, а не способи їх реалізації. Таке твердження є запорукою успіху Internet — жодна компанія не може впливати на InternetcСтандарти й задавати власні правила гри. Наприклад, стандарти Web-сервісів розробляються спільно такими компаніями, як IBM, Microsoft, Ariba і деякими іншими, і обговорюються комітетом World Wide Web Consortium (W3C).

Web-сервіси базуються на три основні WebpСтандартах:

- SOAP (Simple Object Access Protocol) — на протоколі для посилки повідомлень по протоколу HTTP і іншим InternetpПротоколам;
- WSDL (Web Services Description Language) — мовою для опису програмних інтерфейсів Web-сервісів;
- UDDI (Universal Description, Discovery and Integration) — на стандарті для індексації Web-сервісів.



Малюнок 2- 1

На мал. 2-1 показане, як ці три стандарти взаємодіють друг з іншим.

Сервери додатків є сховищами Web-сервісів і роблять їхніми доступними через протоколи HTTP GET, HTTP POST і HTTP SOAP.

Існуючі Web-сервіси описуються в WsdleДокументах, які розташовуються або на сервері додатків, або в спеціальних XmlлiСховищах.

WSDL-документ може посилатися на інші WSDL-документи й документи XSD (XML Schema), у яких описані типи даних, використовувані Web-сервісами. XML-сховища використовуються для керування WSDL-документами. У середині WSDL-документа перебуває адреса (URL) Web-сервісу. Web-сервіси описані й проіндексовані в бізнесреєстрі, що містить адреси (URL) WSDL-документів.

У наступних розділах ми розглянемо три основні Web-стандарту, на яких базуються Web-сервіси SOAP, WSDL і UDDI, більш докладно.

SOAP — Simple Object Access Protocol

SOAP — це стандарт для відсилання й одержання повідомлень по Internet. Споконвічно цей протокол був запропонований фірмою Microsoft у якості засобу для вилученого виклику процедур (RPC, Remote Procedure Call) по протоколу HTTP, а специфікація SOAP 1.0 (Userland, Microsoft, Development) була тісно пов'язана з Component Object Model. Фірма IBM і ряд інших компаній, у тому числі Lotus, внесли певний вклад у розвиток цього протоколу, і його стандарт був спрямований на розгляд комітетом W3C.

Специфікація SOAP визначає XML-«конверт» для передачі повідомлень, метод для кодування програмних структур даних у форматі XML, а також засобу зв'язку по протоколу HTTP.

SOAP-повідомлення бувають двох типів: запит (Request) і відповідь (Response). Запит викликає метод вилученого об'єкта, відповідь повертає результат виконання даного методу. На мал. 2 і 3 наведені приклади запиту й відповіді у форматі SOAP.



Малюнок 2- 2



Малюнок 2- 3

Специфікація SOAP визначає формат кодування, який, у свою чергу, задає спосіб вистави даних в XmlFormatі.

Однієї з перших реалізацій протоколу SOAP була JavaeВерсія, створена фірмою Developmentor. Компанія IBM випустила власну JavaeВерсію, відому за назвою IBM SOAP4J, доступну на WebyВузлі alphaworks (<http://www.ibm.com/alphaworks/>). Згодом ця версія була інтегрована в проект Apache XML (<http://www.xml.apache.org/>). У цей час вона зветься Apache SOAP 2.0 і працює під керуванням Apache Tomcat, IBM Websphere Application Server і інших серверів, що підтримують сервлети. Версія Microsoft, відома за назвою SOAP Toolkit, дозволяє використовувати Visual Basic. Більш повний список відомих реалізацій протоколу SOAP можна знайти на WebyВузлі SOAP Webservices Resource Center (<http://www.soap-wrc.com/webservices/>), а список доступних SOAP Web-сервісів — на WebyВузлі SOAP Web Services (<http://www.xmethods.net/>).

Wsdl-webservices орієнтована мова

Для того щоб додатка могли використовувати Web-сервіси, програмні інтерфейси останніх повинні бути детально описані — із цього погляду мова WSDL відіграє ту ж роль, що й мова Interface Definition Language (IDL) у розподілених обчисленнях. Опис може включати таку інформацію, як протокол, адреса сервера, номер використовуваного порту, список доступних операцій, формат запиту й відповіді й т.п.

Для опису цієї інформації було запропоновано кілька мов. Одним з них була мова Service Description Language (SDL), розроблений Microsoft, що й входив у першу версію Microsoft SOAP Toolkit. Компанія IBM переробила специфікацію й, використавши специфікацію Network Accessible Service Specification Language (NASSL), випустила NASSL Toolkit як частина SOAP4J. Ідеї, реалізовані в NASSL, вплинули на специфікацію мови SOAP Contract Language (SCL), запропоновану Microsoft. У цей час обидві специфікації (NASSL і SDL/SCL), а також пропозиції інших фірм враховані в специфікації

мови WSDL. Для опису бізнес-логіки IBM і Microsoft працюють над специфікацією мови Web Services Flow Language (WSFL).

На мал. 2-4 показаний кістяк опису сервісів мовою WSDL.



Малюнок 2- 4

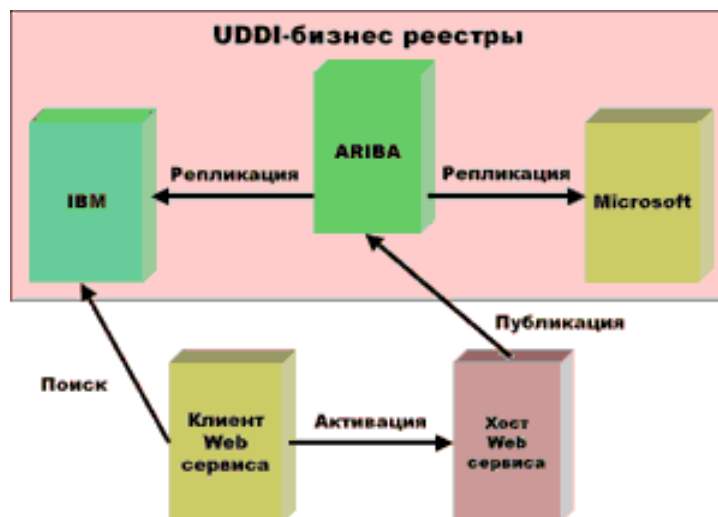
Як ми бачимо, опис сервісів являє собою XmlDocument, що полягає з декількох елементів, у тому числі з опису простору імен (namespace), опису типів і елементів, повідомлень, порту, а також можливих операцій — запитів і відповідей.

Файл, що містить опис сервісів, є досить комплексним документом, тому для його створення по можливості слід користуватися автоматичними генераторами, включеними до складу засобів розробки.

Опис мови WSDL можна знайти на WebaСайті компанії Microsoft за адресою <http://www.msdn.microsoft.com/xml/general/wsdl.asp/>.

UDDI — Universal Description, Discovery and Integration

Завдання UDDI — надати механізм для виявлення Web-сервісів. UDDI задає бізнес-реєстр, у якому провайдери Web-сервісів можуть реєструвати сервіси, а розроблювачі — шукати необхідні їм сервіси. Компанії IBM, Microsoft і Ariba створили власні UddieРеєстри (незабаром ці реєстри будуть об'єднані в WebeРеєстр), в одному з яких розроблювачі можуть зареєструвати свої Web-сервіси, після чого дані будуть автоматично репліювані в інші реєстри (Див. мал. 5).



Малюнок 2- 5

UDDI базується на елементах чотирьох типів: Business Entity, Business Service, Binding Template і Technology Model. Елемент Business Entity описує індустрію, що надає даний Web-сервіс. Цей елемент може включати опису категорій для даної індустрії, що полегшують більш детальний пошук сервісів.

Business Service — це клас сервісів у рамках певної галузі промисловості або послуг. Кожна галузь належить певному елементу Business Entity.

Разом Binding Template і Technology Model визначають Web-сервіс. Technology Model містить абстрактний опис, а Binding Template — конкретну специфікацію сервісу. Кожний елемент Binding Template належить певному елементу Business Service, але кілька елементів Binding Template можуть посилатися на один елемент Technology Model.

Бізнес-Реєстр UDDI сам є SOAP Web-сервісом. Він підтримує операції створення, модифікації, видалення й пошуку елементів усіх чотирьох розглянутих вище типів.

Повний опис UDDI можна знайти на WebaСайті за адресою <http://www.uddi.org/>.

Засоби розробки

Розглянувши три основні ВебаСтандарту, на яких базуються Web-сервіси, — SOAP, WSDL і UDDI, ми тепер знаємо, що вони є досить комплексними при створенні всіх необхідних для опису Web-сервісу Xml-документів. Цю роботу виконують спеціальні засоби розробки, надаючи розроблювачам можливість зосередитися на бізнес-логіці створюваного сервісу, а не на низькорівневих деталях його реалізації. Серед найбільш популярних у цей час засобів розробки Web-сервісів слід згадати Microsoft SOAP Toolkit і IBM XML and Web Services Development Environment (WSDE).

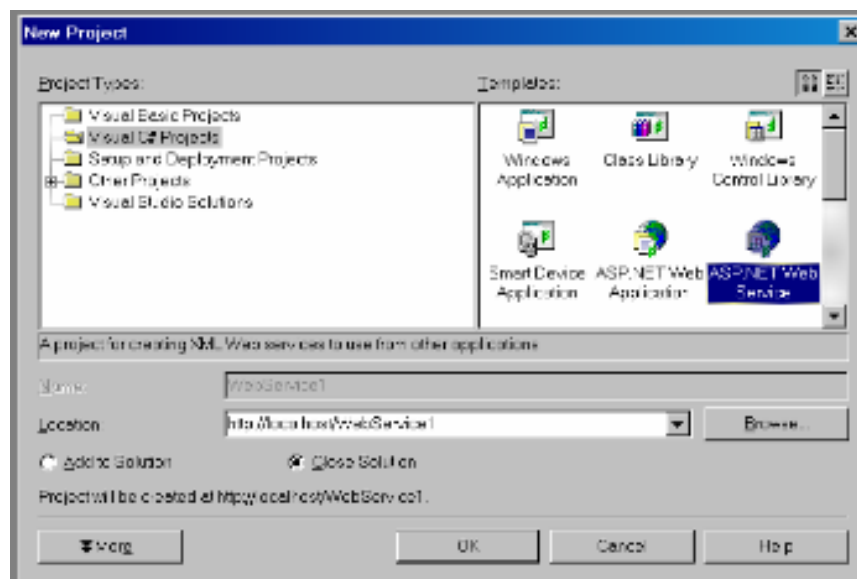
Нижче ми розглянемо ще один засіб для створення Web-сервісів — Microsoft Visual Studio.NET, яке незабаром стане основним інструментом для розроблювачів, що створюють рішення для платформи Microsoft .Net.

3 Порядок виконання роботи

3.1 Підготовка до роботи

Для створення WebСервісу засобами Microsoft Visual Studio.NET необхідно виконати наступні дії:

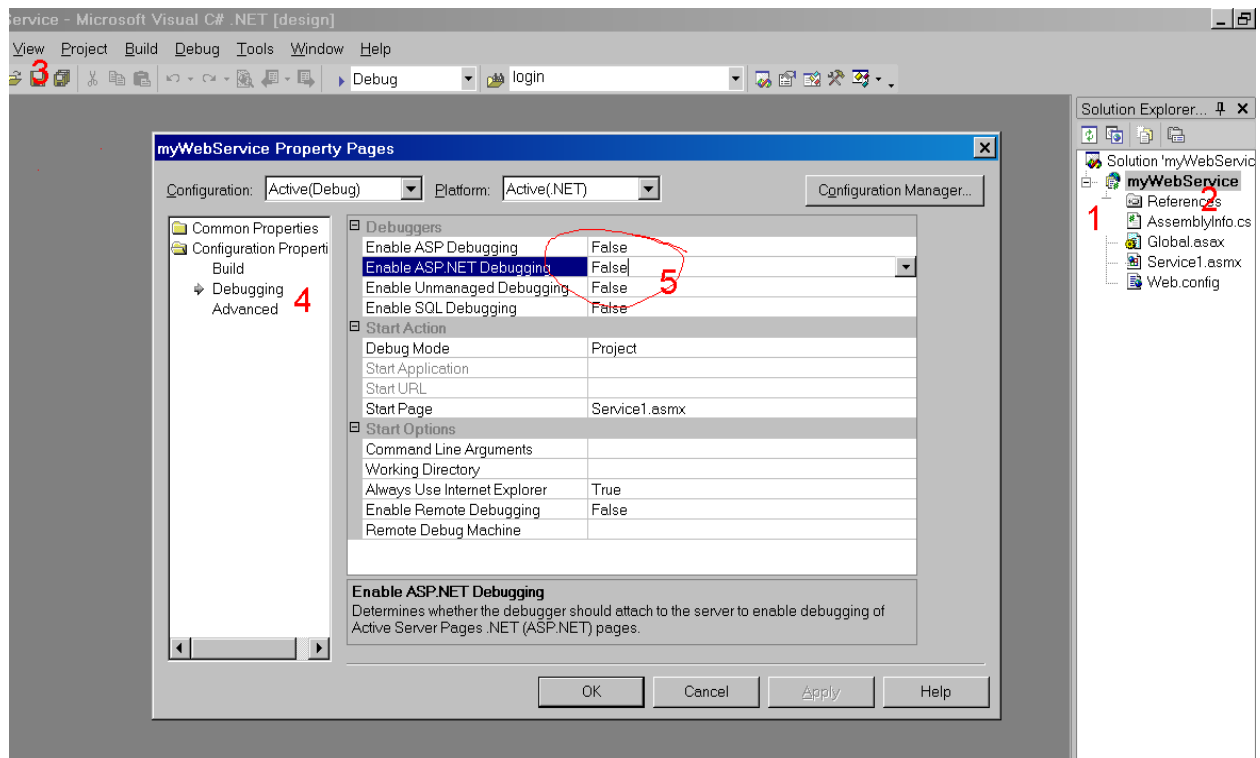
1. Запустити Visual Studio.NET 7.0.
2. Виконати команду File | New | Project (або вибрати команду Create New Project на стартовій сторінці VS Home Page).
3. У панелі New Projects вибрати Visual C# Projects, у панелі Templates вибрати Web Service (мал. 3-1).
4. У поле Location вибрати `http://localhost/mywebservice1`. Наш перший проект буде називатися mywebservice
5. Задати назва сервісу й нажати Ok.



Малюнок 3- 1

Установимо необхідні опції проекту (мал. 3-2)

- а) В **Solution explorer** вибрати проект **mywebservice**
- б) **View -> Property Pages**
- в) У галузі дерева **Configuration Properties -> Debugging** установлюємо **ASP.NET debugging** значення «False». Натискаємо «**OK**»



Малюнок 3- 2

На цьому підготовка до роботи над проектом закінчена. Тепер приступимося до створення самого проекту.

3.2 Приклад виконання

3.2.1 Створення web - служби

Для прикладу розглянемо web службу за допомогою якої будемо переводити курс із якої або валюти на рублі, при цьому курс уводить сам користувач. У реальних умовах процес перекладу можна виразити як «СУМА * КУРС», тобто добуток суми на курс дає нам шукане значення.

При створенні нової web служби в Visual Studio 7.0 (або .Net), програмістові надається шаблонний код web служби, для того, що б не переписувати однотипний службовий код по декілька раз. Це сильно спрощує роботу й дозволяє зосередити увагу на алгоритмі й структурі програми.

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

```
namespace mywebservice
{
    /// <summary>
    /// Summary description for Service1.
    /// </summary>
    [WebService(Namespace="http://localhost/mywebservices")]
```

```

public class Service1 : System.Web.Services.WebService
{
    public Service1()
    {
        //CODEGEN: This call is required by the ASP.NET Web Services Designer
        Initializecomponent();
    }

    #region Component Designer generated code

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

    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    private void Initializecomponent()
    {
    }

    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
    protected override void Dispose( bool disposing )
    {
        if(disposing && components != null)
        {
            components.Dispose();
        }
        base.Dispose(disposing);
    }

    #endregion

    // WEB SERVICE EXAMPLE
    // The Helloworld() example service returns the string Hello World
    // To build, uncomment the following lines then save and build the project
    // To test this web service, press F5

    //      [Webmethod]
    //      public string Helloworld()
    //      {
    //          return "Hello World";
    //      }

    }
}

```

Розроблювачі подбали про починаючих програмістів і відразу створили функцію, що повертає фразу «Hello world». Нам необхідно створити свою функцію яка буде виконувати дія яке нам необхідно, а саме переклад валют.

При створенні проекту вже створився файл service1.asmx.... у ньому нам має бути написати код

Додамо до існуючого коду функцію convertrib2Euro якої будемо передавати два параметри: перший
тип: число із плаваючою крапкою
призначення: Сума

курс: потрібний курс

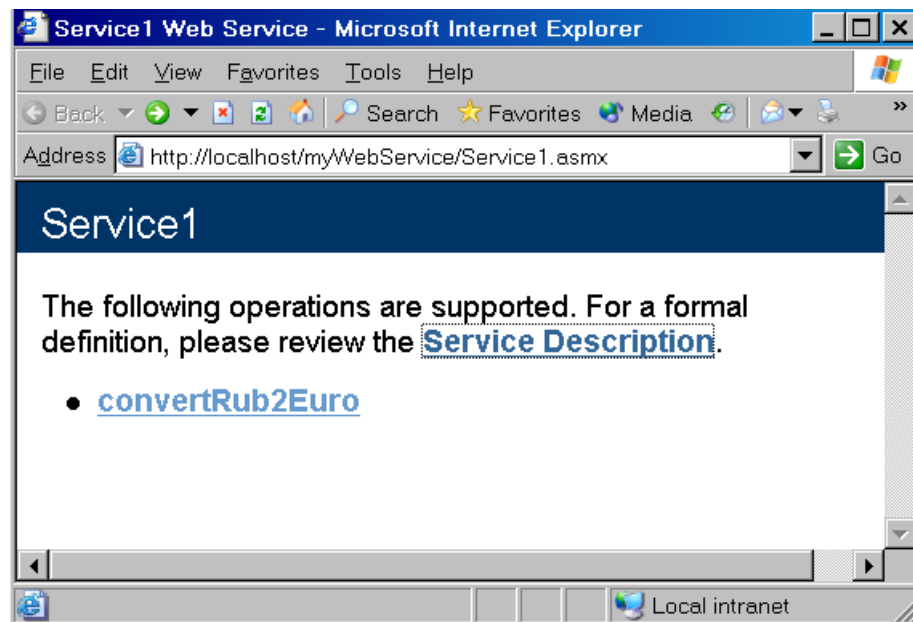
Функція повертає строкове значення.

На малюнку 3-3 наведений текст функції

```
[WebMethod]
public string convertRub2Euro(double num,double kurs)
{
    return Convert.ToString(num*kurs);
}
```

Малюнок 3- 3

Тепер запустите проект нажавши кнопку  на панелі інструментів. Запуститися браузер зі сторінкою на якій будуть відображені функції, доступні для використання. Директива “[Webmthod]” перед кодом функції вказує компіляторові на те що ця функція буде викликатися іншими додатками.



Малюнок 3- 4

Нажавши на **Service Description**, можна подивитися в XML форматі як і які методи підтримує наша web служба. Нижче наведений побачений результат:

```
<?xml version="1.0" encoding="utf-8" ?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tm="http://microsoft.com/wsdl/mime/textmatching/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
  xmlns:tns="http://localhost/mywebservices" xmlns:s="http://www.w3.org/2001/XMLSchema"
  xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  targetnamespace="http://localhost/mywebservices"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementformdefault="qualified" targetnamespace="http://localhost/mywebservices">
      <s:element name="convertRub2Euro">
        <s:complexType>
          <s:sequence>
```

```

<s:element minoccurs="1" maxoccurs="1" name="num" type="s:double" />
<s:element minoccurs="1" maxoccurs="1" name="kurs" type="s:double" />
  </s:sequence>
</s:complexType>
</s:element>
<: <s:element name="convertrub2Euroresponse">
<: <s:complexType>
<: <s:sequence>
  <s:element minoccurs="0" maxoccurs="1" name="convertrub2Euroresult" type="s:string" />
    </s:sequence>
  </s:complexType>
  </s:element>
  </s:schema>
  </wsdl:types>
<: <wsdl:message name="convertrub2Eurosoapin">
  <wsdl:part name="parameters" element="tns:convertrub2Euro" />
    </wsdl:message>
<: <wsdl:message name="convertrub2Eurosoapout">
  <wsdl:part name="parameters" element="tns:convertrub2Euroresponse" />
    </wsdl:message>
<: <wsdl:porttype name="Service1Soap">
<: <wsdl:operation name="convertrub2Euro">
  <wsdl:input message="tns:convertrub2Eurosoapin" />
  <wsdl:output message="tns:convertrub2Eurosoapout" />
    </wsdl:operation>
  </wsdl:porttype>
<: <wsdl:binding name="Service1Soap" type="tns:Service1Soap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
<: <wsdl:operation name="convertrub2Euro">
  <soap:operation soapaction="http://localhost/mywebservices/convertrub2Euro" style="document"
    />
<: <wsdl:input>
  <soap:body use="literal" />
    </wsdl:input>
<: <wsdl:output>
  <soap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
<: <wsdl:binding name="Service1Soap12" type="tns:Service1Soap">
  <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
<: <wsdl:operation name="convertrub2Euro">
  <soap12:operation soapaction="http://localhost/mywebservices/convertrub2Euro"
    style="document" />
<: <wsdl:input>
  <soap12:body use="literal" />
    </wsdl:input>
<: <wsdl:output>
  <soap12:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
<: <wsdl:service name="Service1">
<: <wsdl:port name="Service1Soap" binding="tns:Service1Soap">
  <soap:address location="http://localhost/mywebsevice/Service1.asmx" />
    </wsdl:port>
<: <wsdl:port name="Service1Soap12" binding="tns:Service1Soap12">
  <soap12:address location="http://localhost/mywebsevice/Service1.asmx" />
    </wsdl:port>

```

```
</wsdl:service>
</wsdl:definitions>
```

Примітивна web – служба готова до використання. Нажавши на назву функції, відкриється наступна сторінка в якій крім технічної інформації про службу присутні поля для введення вхідних параметрів створеної функції

Service1

Click [here](#) for a complete list of operations.

convertrub2Euro

Test

To test thereperation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
num:	12
kurs:	12

SOAP

Розглянемо приклад простого виклику та відповіді SOAP протоколу. The **placeholders** shown need to be replaced with actual values.

```
POST /mywebservice/Service1.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```
Soapaction: "http://localhost/mywebservices/convertrub2Euro"
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
```

```
  <soap:Body>
    <convertrub2Euro xmlns="http://localhost/mywebservices">
      <num>double</num>
      <kurs>double</kurs>
    </convertrub2Euro>
```

```
  </soap:Body>
</soap:Envelope>
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <convertrub2Euroresponse xmlns="http://localhost/mywebservices">
      <convertrub2Euroresult>string</convertrub2Euroresult>
```

```

    </convertRub2EuroResponse>
  </soap:Body>
</soap:Envelope>

```

HTTP POST

Ось приклад простого HTTP POST запиту та відповіді. The **placeholders** shown need to be replaced with actual values.

```

POST /mywebservice/Service1.asmx/convertRub2Euro HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: length

num=string&kurs=string
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<string xmlns="http://localhost/mywebservices">string</string>

```

Уведемо суму рівну 12 і курс рівний 12, після натискання на кнопку invoke буде виданий результат у вигляді xml

```

<?xml version="1.0" encoding="utf-8" ?>
<string xmlns="http://localhost/mywebservices">144</string>

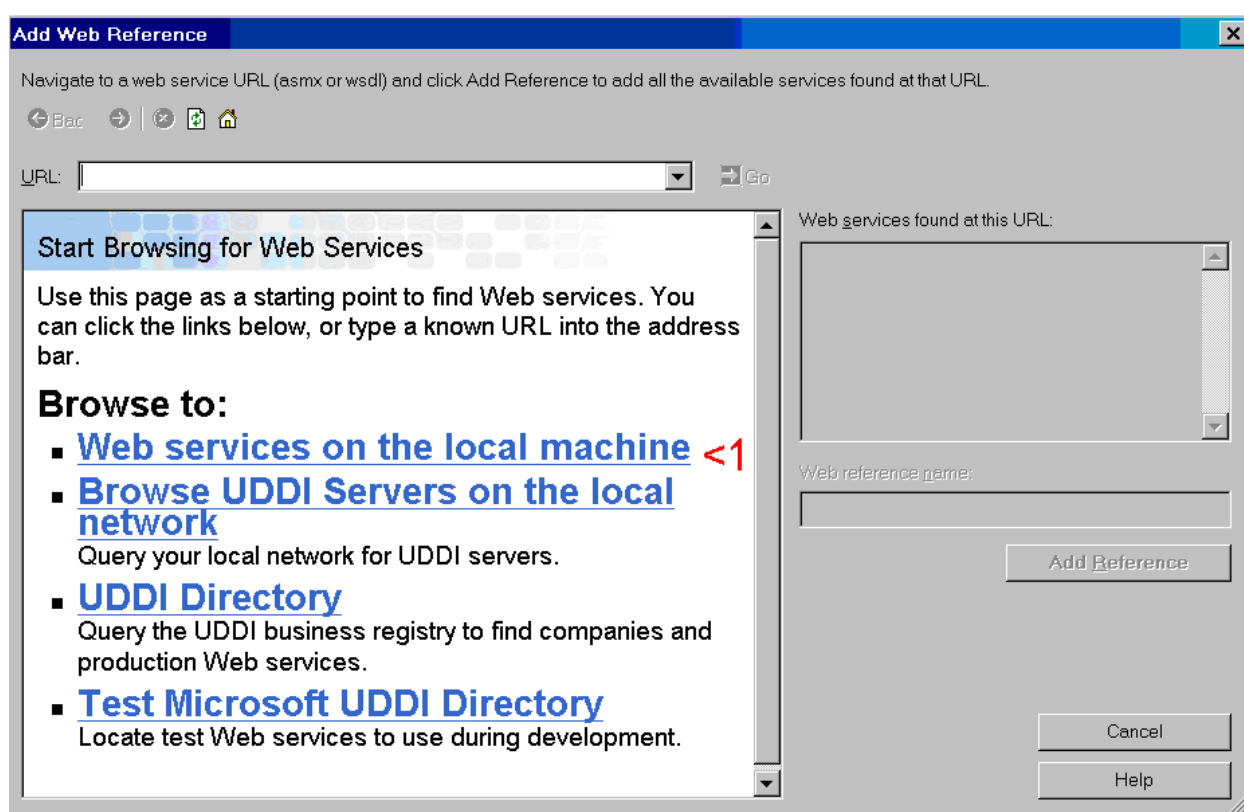
```

Тут знаходиться полічений результат і тип даних які необхідно використовувати при роботі з ними

3.2.2 Створення клієнта для доступу до web службі.

Web – служба може бути використана додатком працюючим на будь-якій платформі. Створимо найпростіше windows додаток конторі буде використовувати створену web – службу для перекладу курсу валют. Сам додаток може перебувати де завгодно поетом в етап створення windows додатка пропустимо й приступимося до підключення web – служби до нашого додатка.

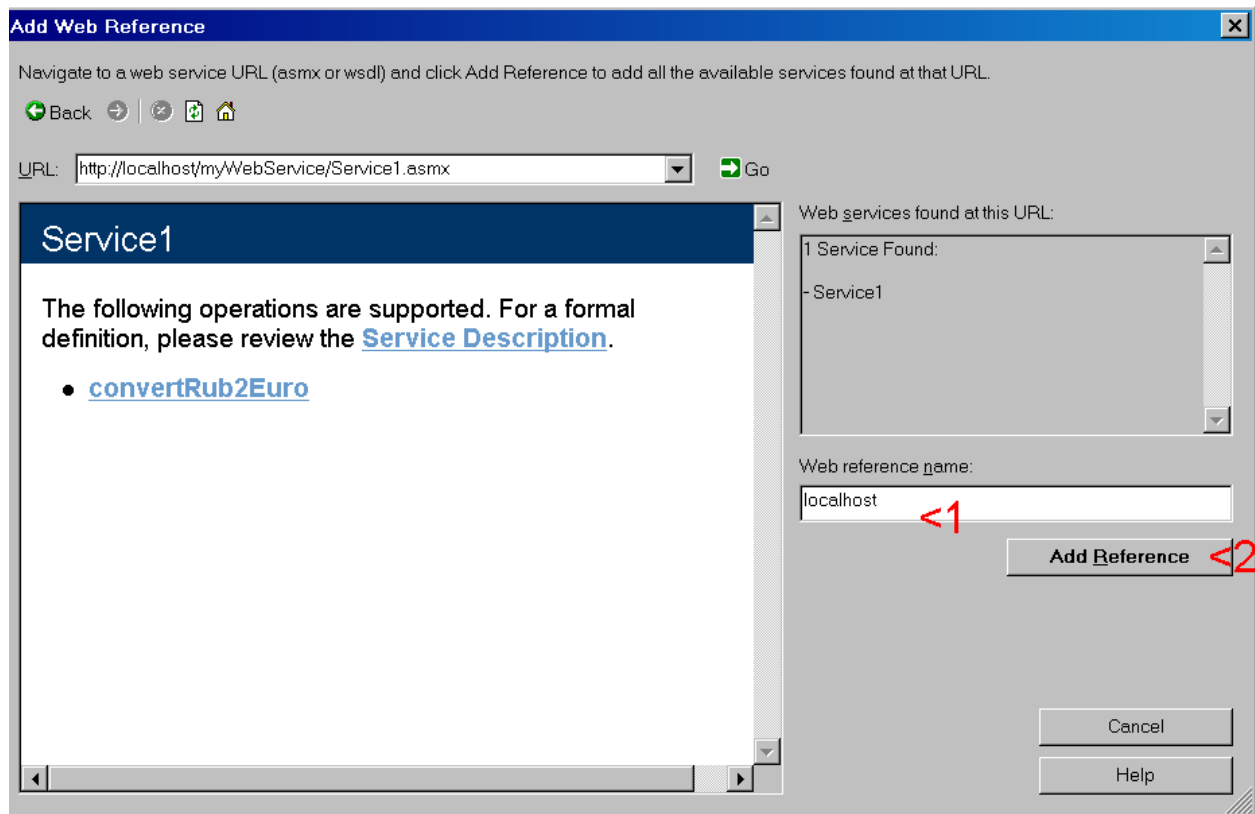
В **Solution Explorer** -> **Add Reference** натискаємо праву кнопку миші й вибираємо **Add Web Reference**. У вікні, що з'явилося (мал. 3-5) вибираємо **Web services on the local machine**, тому що наша створена web служба перебувати на локальному комп'ютері. Якщо є локальна мережа й служба доступна з неї то шукати її природно потрібно в **Browse UDDI Servers on the local network**



Малюнок 3- 5

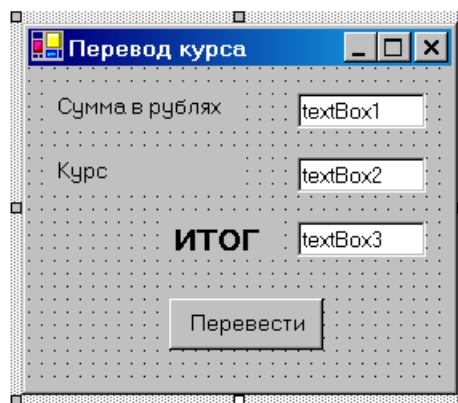
Далі в списку, що з'явився, вибираємо нашу web – службу. Після того, як система САМА знайде функції потрібно додати їх у проект кнопкою **Add Reference** (рис 3-6).

Поле **Web Reference Name** можна залишити недоторканим, але для того що б внести значеннєве навантаження в ім'я посилання на службу її можна додати значеннєве навантаження.



Малюнок 3- 6

У результаті проведених дій у нас є проект windows додатка, який зв'язано з розробленою раніше web службою. Тепер необхідно розробити інтерфейс самого додатка. У даному прикладі будемо використовувати наступний вид додатка.



Малюнок 3- 7

При натисканні на кнопку «Перевести» будемо виконувати наступні дії для виклику функції web – служби.

```
private void button1_Click(object sender, System.EventArgs e)
{
    localhost.Service1 serv= new webserviceclient.localhost.Service1();
    textbox3.Text =
serv.convertrub2Euro(Convert.ToDouble(textbox1.Text),Convert.ToDouble(textbox2.Text)).ToString
();
}
```

У результаті одержимо додаток яке буде використовувати web – службу, а точніше функцію web – служби для роботи. Тому що для прикладу був

обраний досить простий варіант завдання, це зовсім не виходить, що здатності web служб обмежуються подібними можливостями...

3.3 Складання звіту лабораторної роботи.

Для складання звіту лабораторної роботи необхідно надати наступні дані й відповісти на контрольні питання.

1. Мета
2. Умова завдання
3. Скриншот документа з робочими функціями web – служби
4. Функції web служби (Опис вхідних/вихідних параметрів+код)
4. Висновок

4 Контрольні питання

1. Що називається web – службою?
2. Перелічіть стандарти для Web-сервісів?
3. Що таке SOAP?
4. Що таке WSDL?
5. Що таке UDDI?
6. Якою директивою визначається використання функції іншими додатками?
7. Яке розширення мають файли web – служб?
8. Який формат даних, що вертаються, використовується в web - сервісах(службах)?
9. Як додати посилання на web службу локальної мережі в windows додаток?
10. Чи можливо використовувати доступ до БД из web ?
11. Яку роль відіграє ім'я посилання на службу в windows додатку?
12. Приведіть приклади, де можна використовувати web – служби.

5 Завдання на групу

1. Створити web – службу для виводу середнього геометричного для 4х чисел, притім для тих з них, які більше заданого М.
2. Створити web – службу для доступу до БД.
3. Створити web – службу для виводу із БД усіх покупців магазину.(БД створюється самостійно)
4. Створити web – службу для перевірки імені користувача й пароля
5. Створити web – службу для перевірки імені користувача й пароля з використанням бази даних MS Access
6. Створити web – службу для використання якого-небудь методу оптимізації.

Лабораторна робота 2. Тема: «Перевірка коментарів у блозі, інтегрована з базою даних, з використанням XML»

1. Мета роботи

Мета роботи – створення додатку, який використовує XML для перевірки коментарів у блозі.

2. Основні теоретичні положення

У цій роботі розглядається додаток, архітектура якого представлено на малюнку 1.



Малюнок 1. Схема перевірки коментарів у блозі, інтегрованому з базою даних

Це Web-додаток обробляє інформацію, надану Web-сервісами онлайн-щоденників (блогів). Інформація, що включає в себе дані, які відносяться до всіх блогів зазначеного власника, а також усі коментарі до записів у цих блогах, може бути отримана у вигляді Xml-Документів Atom (приклад такого документа наведено в [лістингі 1](#)). Додаток дозволяє авторам блогів швидко переглянути свіжі коментарі до своїх записів і видаляти коментарі з небажаним змістом. Воно здатне представляти дані у вигляді Web-сторінок з формами вводу у форматі XHTML. Враховуючи той факт, що в XML надаються як коментарі, так і вихідний текст сторінок (XHTML), логічно використовувати засоби, що дозволяють маніпулювати даними безпосередньо в цьому форматі.

Лістинг 1. Приклад коментарів у вигляді Xml-документа у форматі Atom

```

<?xml version="1.0" encoding="UTF-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <title type="text">Websphere Community Blog</title>
  ...
  <entry>
    <id>tag:blogger.com,1999:blog-1417695962027703953.post-6498982274841848264</id>
    <published>2009-10-17T13:06:00.000-05:00</published>
    <updated>2009-10-17T13:06:00.000-05:00</updated>
    <atom:title xmlns="" xmlns:atom="http://www.w3.org/2005/Atom" type="text">
      Questionable spamming comment title
    </atom:title>
  </entry>
</feed>
  
```

```

<atom:content      xmlns=""      xmlns:atom="http://www.w3.org/2005/Atom"
type="html">
    Questionable spamming comment content
</atom:content>
...
<atom:author xmlns="" xmlns:atom="http://www.w3.org/2005/Atom">
    <atom:name>Joe Smith</atom:name>
    <atom:uri>http://joe.uri.com</atom:uri>
    <atom:email>jsmith@email.com</atom:email>
</atom:author>
</entry>
...
</feed>

```

У якийсь момент автор блога може помітити, що небажані коментарі надходять від тих самих користувачів або від адрес одного домена (див. елемент `atom:author` у [лістингі 1](#)). У цьому випадку додаток дозволяє не тільки вилучити коментар, але й позначити користувача або весь домен прапором "спам". Цей прапор не втримується в документах, що надходять від Web-сервісів, тому його необхідно зберігати в деякій базі даних. Оскільки інформація вже представлена в XML, логічно використовувати в якості бази даних `XmlлСховище`. У майбутньому це дозволить реалізувати такі функції додатка, як, наприклад, побудова статистичних звітів про спамерів і автоматичні рекомендації про видалення коментарів.

Як найкраще реалізувати такий додаток?

До того часу, як СУБД почали надавати власну підтримку XML, існувало два основні способи роботи з даними в цьому форматі. Перший з них полягав у тому, що дані серілізувались у рядок, який потім зберігався в БД у вигляді великого текстового об'єкта (CLOB). Цей підхід негативно позначається на швидкодії й не дозволяє звертатися до збережених даних, як до документа XML.

Другий спосіб зводиться до відображення даних в XML на структуру реляційної БД, таблиці якої повинні приблизно відповідати схемам документів XML. Такий підхід має проблеми з точністю відображення, які обумовлені можливими розбіжностями в реляційній схемі й схемі XML. Через це доводиться підтримувати спеціальний код, що виконує перетворення даних, до яких, як і раніше, не можна звертатися за допомогою `XmlлЗапитів`. Нарешті, нові вимоги до додатка можуть впливати на схеми XML, у той час як зміни в реляційних схемах найчастіше виявляються надто трудомісткими.

У зв'язку з ростом популярності XML у корпоративних додатках деякі СУБД почали надавати можливість зберігання даних не тільки в табличному виді, але й у вигляді `XmlлДокументів`. У цьому випадку інформація втримується в колонках, що мають тип "XML". До таких СУБД ставляться Apache Derby, IBM DB2®, Oracle Database і Microsoft® Sqlserver.

Крім того, раніше виникали складності з передачею інформації із проміжного шару додатка в базу даних. До появи JDBC 4.0 єдиним варіантом було використання типів даних String або CLOB. Раніше вже згадувалося, що при цьому погіршується продуктивність через необхідність серілізації. До того ж, при такому підході потрібні нестандартні розширення SQL для розбору даних при збереженні в стовпцях типу "XML". На щастя, в JDBC 4.0 з'явилася стандартна підтримка типу даних SQLXML, завдяки якому документи можуть зчитуватися й записуватися у вигляді XML. JDBC 4.0 дозволяє звертатися до даних як до рядків за допомогою потоків читання й записи, а також як до об'єктів Source і Result з JAXP. Таким чином, дані у форматі XML можуть вільно передаватися між проміжним шаром і базою даних без необхідності додаткових витрат на відображення й втрат продуктивності.

Простий приклад використання JDBC 4.0 при роботі з XML

Далі розглянемо приклад використання засобів підтримки XML в JDBC 4.0. Читання даних у вигляді XML проводиться аналогічно вибірці інших типів даних, тобто шляхом виконання наступної послідовності дій:

1. підготовка запиту (prepared statement);
2. виконання підготовленого запиту й одержання вибірки;
3. одержання об'єкта типу SQLXML з вибірки;
4. читання вмісту об'єкта типу SQLXML за допомогою одного з надаваних getометодів;
5. звільнення об'єкта типу SQLXML.

Простий приклад наведено в лістингу 2.

Лістинг 2. Читання XmlнДаних за допомогою JDBC 4.0

```
Preparedstatement ps =
    dbconnection.preparestatement("SELECT      somexmlcolumn      FROM
somexmltable");

ResultSet result = ps.executeQuery();

result.next();

SQLXML xml = result.getsqlxml("somexmlcolumn");

Streamsource source = xml.getsource(Streamsource.class);

// Читання даних з потоку

xml.free();
```

У цьому прикладі читання даних проводиться за допомогою Jaxp-об'єкта source, що є екземпляром Streamsource. Таким чином, звертатися до XmlнДаним може будь-який API, здатний працювати із джерелами JAXP. Streamsource дозволяє використовувати будь-яка виставу даних у пам'яті. Як правило, потокова вистава є більш ефективною в порівнянні з такими типами джерел, як

DOM або SAX, робота з якими пов'язана з накладними витратами на конвертацію даних в об'єкти й виклики методів API.

Для запису документів XML у базу даних через JDBC 4.0 виконуються наступні дії:

1. підготовка запиту;
2. формування об'єкта SQLXML після установки з'єднання з базою даних;
3. одержання доступу до вмісту об'єкта SQLXML через один з доступних setометодів;
4. вказівка об'єкта SQLXML у якості параметра підготовленого запиту;
5. запис умісту об'єкта SQLXML з використанням раніше отриманого доступу;
6. виконання запиту;
7. звільнення об'єкта SQLXML.

Приклад виконання цих дій наведено в лістингу 3.

Лістинг 3. Запис документа XML у базу даних через JDBC 4.0

```
PreparedStatement ps = dbconnection.prepareStatement(
    "UPDATE somexmltable SET somexmlcolumn = ?");

SQLXML xml = dbconnection.createSqlXml();

StreamResult result = new StreamResult(xml.setBinaryStream());

ps.setSqlXml(1, xml);

// Запис у потік

ps.executeUpdate();

xml.free();
```

Проміжний шар програмного забезпечення

Як згадувалося вище, до переваг СУБД із підтримкою XML у порівнянні з CLOB або відображенням на реляційну структуру можна віднести кращу продуктивність, відповідність схемі XML і простоту програмування. Більш висока продуктивність пояснюється тим, що дані представляються в єдиному форматі без необхідності перетворення з однієї моделі в іншу (що в найкращому разі приводить до серіалізації й розбору). Відповідність схемі XML означає, що єдина версія документа XML передається й зберігається в базі даних, тобто не відбувається відновлення документа на основі вмісту реляційних таблиць, що служать для зберігання окремих частин XML. Нарешті, розробка додатків полегшує завдяки тому, що не потрібно писати код для відображення структури XML на реляційну структуру бази даних.

Далі розглянемо проміжний шар програмного забезпечення. Обробка даних безпосередньо в XML дозволяє одержати ті ж переваги в проміжному шарі, а

також, що більш суттєво, при передачі інформації із проміжного шару в базу даних і назад. Крім того, проміжний код додатка може звертатися до безлічі джерел інформації крім бази даних XML, наприклад до каналів Web-сервісів. При цьому всі переваги роботи з XML поширюються на ці джерела даних.

Загальна продуктивність роботи підвищується за рахунок того, що дані не доводиться копіювати при передачі в базу даних або інший XmlЧДжерело й назад. Крім того, поліпшується продуктивність кожного шару окремо, оскільки на кожному рівні можна використовувати найбільш ефективне XmlВиставу даних, при цьому надаючи стандартизовані W3C інтерфейси для доступу до інформації.

Крім того, підтримується висока точність вистави даних. У цьому простому прикладі точність проходження схеми XML не є серйозною проблемою. Однак в інших випадках документи XML можуть бути досить складними, наприклад у податковій звітності, або винятково важливими, наприклад у фінансовій звітності. У таких ситуаціях критично щоб дані зберігали своя вистава на всіх стадіях обробки.

Нарешті, XML дозволяє спростити розробку, оскільки використовується єдина модель даних, з якої можна працювати погодженим образом через інтерфейси XML. З погляду Java™ТРозроблювачів може здатися, що в ряді випадків у проміжному шарі простіше відображати об'єкти на структури даних DOM або JAXB. Однак ці випадки виявляються далеко не настільки прості, наприклад, при роботі зі згаданими вище податковими документами. Крім того, у цьому випадку доводиться вивчати не тільки JDBC і XML, але й інші моделі XmlПрограмування й виконання запитів. Якщо ж дані завжди представлені тільки в XML, то розроблювачеві досить знати тільки модель даних XML і стандарти W3C для навігації по документах, їх перетворенню й вибірці інформації. Більше того, більшість розроблювачів мають ці навички, оскільки такі мови, як Xpath і Xquery уже використовуються в базах даних XML для навігації й виконання запитів.

Подібне спрощення розробки легше всього досягається за допомогою інфраструктури, яка бере на себе більшу частину низькорівневої роботи з одержання інформації від джерела, її перетворенню й збереженню в базі даних, причому без створення проміжних копій. При створенні демонстраційного додатка ми будемо використовувати набір інструментів (Feature Pack) XML для IBM Websphere Application Server V7.0. Цей інструментарій не створює зайвих копій даних, уникаючи тим самим додаткових накладних витрат. При цьому на всіх етапах обробки інформація представлена в XML, що дозволяє легко здійснювати навігацію, конвертації й запити до даних.

Пакет доповнень XML для сервера додатків IBM Websphere Application Server V7.0

Пакет доповнень XML для IBM Websphere Application Server V7.0 надає кошти для навігації, конвертації й запитів до XML у проміжному шарі додатка на основі таких стандартів W3C, як Xpath 2.0, XSLT 2.0 і Xquery 1.0. Враховуючи популярність XML у шарі зберігання даних, ми застосуємо загальний підхід, пропонуваній JDBC 4.0, у додатку, що використовує цей інструментарій і базу даних XML, наприклад IBM DB2 з purexml®, Apache Derby або Oracle. При цьому взаємодія між базою даних і інструментарієм буде виглядати, як показано на малюнку 2.



Малюнок 2. Проста діаграма взаємодії пакета доповнень XML від IBM і бази даних XML

Далі ми продемонструємо роботу з XML у базі даних і проміжному шарі додатка за допомогою СУБД з підтримкою XML, JDBC 4.0 і типу даних SQLXML, а також пакета доповнень XML для Websphere.

Вихідний код цього додатка входить до складу пакета доповнень XML, тому ви можете побрати його за основу ваших власних експериментів. У розділі Ресурси наведені посилання на пакет доповнень XML, а також СУБД Apache Derby і DB2 Express.

Бази даних із власною підтримкою XML, JDBC 4.0, а також пакет додавань XML для Websphere дозволяють створювати додатки із простий і високопродуктивною архітектурою.

Реалізація демонстраційного додатка

У цьому розділі ми розглянемо реалізацію демонстраційного додатку за допомогою пакета доповнень XML, JDBC 4.0 і СУБД із підтримкою XML. Спочатку буде реалізована функція одержання підозрілих коментарів до записів у блозі з використанням Web-сервісу, що передає стрічки Atom (див. рисунок 1). У процесі обробки цих коментарів додаток буде звертатися до бази даних, перевіряючи, не чи були їх автори раніше занесені в список спамерів. На всіх етапах обробки дані будуть представлені у вигляді документів XML. Отже, приступимо до одержання даних про коментарі у вигляді стрічок Atom.

Набір доповнень XML дозволяє завантажувати дані стрічок Atom через Http3'єднання з Web-сервісом блогів і звертатися до них через процедури

Xquery. Ці процедури виконуються середовищем пакета доповнень XML, яка, у свою чергу, викликається через Java API того ж пакета.

Для вибірки підозрілих коментарів зі стрічки Atom використовується наступні вираження Xpath в Xquery:

Лістинг. Вираження Xpath

```
declare variable $comments := (
  /atom:feed/atom:entry[atom:author/atom:name = 'Anonymous'] |
  /atom:feed/atom:entry[matches(atom:content, $my:vulgarwords, 'i')]
  [atom:published > current-datetime() - $my:monthsago];
```

У даному прикладі Xpath використовується для пошуку коментарів у зазначеному користувачем тимчасовому інтервалі, які були відправлені анонімними користувачами й містили вульгарні слова. Обрані коментарі зберігаються в змінній comments.

Далі в Xquery можна перевірити, чи надходив раніше спам від тих же авторів. Для цього виконується запит до бази даних, у якому \$i - це показаний вище вкладений запит на вибірку коментарів (лістинг 5).

Лістинг 5. Вираження Xquery

```
let $spammedbefore := local:hasemailhasspammedbefore($i/atom:author/atom:email/text())
```

Звернувшись до визначення функції, показаної в лістингу 4, ви побачите, як інформація завантажується з бази даних у вигляді XML (лістинг 6).

Лістинг 6. Тіло функції в Xquery

```
declare function local:hasemailhasspammedbefore($emailaddress) as
xs:boolean {
  let $domainname := substring-after($emailaddress, '@')
  return
    if ($domainname = '') then
      false()
    else
      let $jdbcuri := concat('jdbc://getauthorswhohavespammedfromdomain?',
$domainname)
      let $domainspammers := collection($jdbcuri)
      return
        not(empty($domainspammers/spammers/spammer/email[.
$emailaddress]))
};
```

У цьому прикладі з адреси e-mail витягає ім'я домена, яке потім конкатенується з текстом запиту jdbc://getauthorswhohavespammedfromdomain. Після цього формується колекція Xpath 2.0 на основі URI jdbc://getauthorswhohavespammedfromdomain?ІМ'Я_ДОМЕНА.

Колекції в Xpath 2.0 являють собою зручний спосіб інтеграції XmlнДаних, які надходять не з основного документа, оброблюваного програмою на Xquery або XSLT. Реалізація функції collection не стандартизоване, тому кожне середовище виконання Xpath 2.0 може надавати конструктори колекцій за замовчуванням, а також засобу розширення, за допомогою яких користувачі можуть динамічно використовувати власні реалізації. У пакеті доповнень XML роль такого засобу розширення відіграє інтерфейс Xcollectionresolver.

Наша реалізація інтерфейсу Xcollectionresolver відповідає за формування колекцій, чий URI починається із префікса jdbc://. При цьому інша частина URI використовується для пошуку відповідного іменованого запиту. Конструктор колекцій одержує іменований запит, додає в нього параметри й виконує його в базі даних за допомогою JDBC.

У лістингу 7 показано створення екземпляра конструктора після визначення декількох базових запитів. Конструктор одержує на вхід JdbcнЗ'єднання з базою даних для виконання запитів.

Лістинг 7. Конфігурування конструктора колекцій

```
dbstatementssupportssqlxml = new HashMap<String, String>();

dbstatementssupportssqlxml.put("getauthorswhohavespammedfromdomain",
    "SELECT CONTACTS from SPAMMERS where DOMAINNAME = ?");

dbstatementssupportssqlxml.put("updateauthorswhohavespammedbydomain",
    "UPDATE SPAMMERS SET CONTACTS = ? WHERE DOMAINNAME = ?");

dbstatementssupportssqlxml.put("insertauthorswhohavespammedbydomain",
    "INSERT INTO SPAMMERS (CONTACTS, DOMAINNAME) VALUES (?, ?)");

Connection conn = getdatabaseconnection();

Jdbccollectionresolver inputresolver =
    new Jdbccollectionresolver(conn, dbstatementssupportssqlxml);
```

Фрагменти реалізації конструктора показано в лістинге 8. Повний варіант реалізації можна знайти в прикладах коду, що входить до складу пакета доповнень XML. Конструктор фактично виконує дії по вибірці XmlнДаних через JDBC, які були описані вище.

При цьому в конструктор минулого додані наступні можливості, що полегшують його використання в різних ситуаціях.

1. Іменовані запити передаються в конструктор ззовні, а не втримуються в коді в жорстко заданому виді.
2. Конструктор аналізує метадані рядків, що вертаються, і стовпців, тим самим гарантуючи, що тільки дані типу SQLXML перетворюються в XML.

3. Конструктор використовує два інших класу з API пакета доповнень XML, а саме Xsequencecursor і Xitemview, для формування послідовності XML на основі результатів запиту JDBC.

Лістинг 8. Реалізація конструктора

```
public Xsequencecursor getcollection(String uri, String base) {
    // Пошук запиту в заданому наборі (див. лістинг 7)
    String query = lookupnamedquery(uri);
    Preparedstatement p = dbconnection.preparestatement(query);
    Resultset rs = p.executequery();
    ...

    // Ітерування за результатами виконання запиту
    Resultsetmetadata metadata = rs.getmetadata();
    int coltype = metadata.getcolumnntype(jj+1);
    if (coltype = Types.SQLXML) {
        SQLXML sqlx = rs.getsqlxml(...);
        Streamsource source =
sqlx.getSource(Streamsource.class);
        Xitemview item = itemfactory.item(source);
        sqlx.free();
    }

    // Створення послідовності на основі отриманих Xml Даних
    // за допомогою API пакета доповнень XML
    ...
    Xitemview itemview[] = items.toArray(new Xitemview[0]);
    Xsequencecursor sequence = itemfact.sequence(itemview);

    return sequence;
}
```

Отже, тепер у нас є конструктор колекцій загального призначення, який може використовуватися в будь-якій програмі Xquery, що ухвалює на вхід довільне число аргументів колекції, що й повертає, XmlДаних, які у свою чергу можуть оброблятися за допомогою Xpath 2.0, XSLT 2.0 або Xquery 1.0. Нижче наведені ще два приклади. У першому з них створюється список спамерів на основі елемента name, а в другому - вертається число спамерів, від яких було отримано більш десяти коментарів.

Лістинг 9. Приклади Xquery

```
Java:
    dbstatementssupportssqlxml.put("getallspammers", "SELECT CONTACTS from
SPAMMERS");
Xquery:
    let $allspammers := collection('jdbc://getallspammers')
    return
        for $i in $allspammers
        let $first := $i/name/first
        order by $i/name/last
        return
            <name>
                <first>{ $first }</first>
                <last>{ $i/name/last }</last>
            </name>
```

Java:

```
dbstatementssupportssqlxml.put("getallspamauthorswherespamcountgreaterthan",
    "SELECT CONTACTS from SPAMMERS where COUNT > ?");
Xquery:
    let $mincount := 10
    let $allspammers := collection(concat(
        'jdbc:// getallspamauthorswherespamcountgreaterthan?',
        $mincount)
    )
    return
        count($allspammers)
}
```

Слід зазначити, що оскільки конструктор є частиною Xpath 2.0, його можна використовувати в XSLT 2.0 з таким же успіхом, що й в Xquery 1.0. У лістингу 10 наведений простий приклад виклику конструктора з XSLT 2.0.

Лістинг 10. Приклади XSLT

```
<xsl:variable name="allspammers"
select="collection('jdbc://getallspammers')"/>

<xsl:template match="/">
    <p>The current spammer database contains the following domains
and spammers.</p>
    <xsl:for-each select="$allspammers">

        <table>
        <tr>
            <th>Name</th><th>Email</th>
        </tr>
        <xsl:for-each select="$allspammers">
        <tr>
            <td><xsl:value-of select="name"/></td>
            <td><xsl:value-of select="email"/></td>
        </tr>
        </xsl:for-each>
        </table>
    </xsl:template>
```

Аналогічним образом можна реалізувати запис у базу даних XML. XSLT 2.0 дозволяє записувати результати перетворення в кілька документів із зазначеними URI за допомогою інструкції `xsl:result-document`. Як і раніше, дозвіл подібних URI залишається на розсуд середовища виконання. До складу пакета доповнень входить інтерфейс `Xresultsresolver`, завдяки якому можна вказувати, куди слід записувати результати. У нашому додатку такий розв'язний клас (`resolver`) реалізований на основі JDBC. Він використовує іменовані запити й нумеровані параметри, серед яких один має спеціальне ім'я —XML—.

Це дозволяє записувати дані в базу даних, як показано в лістингу 11.

Лістинг 11. Приклад використання інструкції result-document в XSLT

```

<!-- Перевірка: вставка або відновлення даних? -->

<xsl:variable
  select="count($spammersbydomain/spammers/spammer) eq 0"/>
  name="insert"

<!-- Підготовка іменованого запиту типу insert -->

<xsl:variable name="insertjdbcuri"
  select="concat('jdbc://insertauthorswhohavespammedbydomain?--
XML--&', $domain)"/>

<!-- Підготовка іменованого запиту типу update -->

<xsl:variable name="updatejdbcuri"
  select="concat('jdbc://updateauthorswhohavespammedbydomain?--
XML--&', $domain)"/>

<!--
У випадку insert слід додати вміст xsmldoc у БД.
А якщо ні, то оновити дані, передані в xmldoc.
-->

<xsl:template match="/">
  <xsl:when test="$insert">
    <xsl:result-document
      href="{ $insertjdbcuri}"
method="xml" indent="yes">
      <xsl:copy-of select="$xmldoc"/>
    </xsl:result-document>
  </xsl:when>
  <xsl:otherwise>
    <xsl:result-document
      href="{ $updatejdbcuri}"
method="xml" indent="yes">
      <xsl:copy-of select="$xmldoc"/>
    </xsl:result-document>
  </xsl:otherwise>
</xsl:template>

```

Якщо зазначений спамер раніше не зустрічався, то буде виконана перша частина інструкції `xsl:when` і інформація про нього буде додана в базу даних. У цьому випадку середовище XML одержить запит на запис умісту змінної `xmldoc` по наступному URI:

Лістинг 12. URI для запису результату

```
jdbc://insertauthorswhohavespammedbydomain?--XML--&domain.com
```

Цей URI буде перетворений (дозволений) в іменований запит, який у нашому випадку виглядає в такий спосіб:

Лістинг 13. Іменований запит

```
INSERT INTO SPAMMERS (CONTACTS, DOMAINNAME) VALUES (?, ?)
```

У цьому прикладі розв'язний клас поміщає вміст документа XML, який утримується в змінній `xmldoc`, на місце першого параметра. У якості другого параметра використовується `domain.com`.

Як і у випадку із чтением данных, запис `XmlнДокументів` по зазначеним URI може використовуватися в різних мовах роботи з XML. У попередньому прикладі застосовувався XSLT (див. листинг 11), але аналогічним образом можна написати програму на Xquery 1.0, яка буде записувати результати в базу даних XML.

Зауваження: дозвільні класи на основі JDBC і альтернативні методи

Підхід, описуваний у цій статті - створення дозвільних класів загального призначення, які реалізують інтерфейси з пакета доповнень XML і використовують можливості JDBC 4.0 - це лише один з можливих способів природньої й швидкої інтеграції доповнень XML і бази даних з підтримкою XML.

Ви також можете створити конструктор колекцій у вигляді функції розширення `Xpath 2.0`. Це більш спеціалізоване рішення, приклад якого наведено в листингу 14.

Лістинг 14. Функція спеціального призначення в Xquery

```
declare function local:hasemailhasspammedbefore($emailaddress) as
xs:boolean {

let $domainname := substring-after($emailaddress, '@')

return
  if ($domainname = '') then
    false()
  else
    let
      $domainspammers
    my:getauthorswhohavespammerfromdomain($domainname)
    return
      not(empty($domainspammers/spammers/spammer/email[.
$emailaddress]))
};
```

Перевагою такого підходу є стислість коду, відповідального за читання й запис даних. При цьому недоліком є необхідність створення окремих функцій розширення на кожний запит (замість одного конструктора колекцій на всі запити). Більше того, у загальному випадку це рішення працює тільки для читання даних, оскільки створення функцій розширення для запису не завжди виявляється можливим. Таким чином, конструктори колекцій і розв'язні класи для запису даних являють собою більш загальний, більш природній і погоджений підхід.

Слід зазначити, що конструктор колекцій і розв'язний клас для запису входять до складу прикладів до пакета доповнень XML. Інакше кажучи, цей

код - це не більш ніж частина демонстраційного додатка і його буде необхідно розширити для використання в реальній системі.

Висновок.

Обробляючи дані у вигляді документів XML (для цього служить пакет доповнень XML), ви можете одержати аналогічні переваги в проміжному шарі додатка, а також, що ще більш суттєво, при переносі інформації із проміжного шару в базу даних і назад. Крім того, ці переваги можуть поширюватися на інші джерела даних XML, до яких звертається ваш проміжний шар, наприклад інформаційні стрічки, отримані від Web-сервісів.

Необхідні програмні продукти й технології

- Завантажити програмне забезпечення, необхідне для запуску демонстраційного Web-додатку.
 - Пакет доповнень XML для WebSphere Application Server V7.0: набір стандартів програмування XmleДодатків, що підвищують продуктивність розроблювачів, що й дозволяють вирішувати актуальні на сьогоднішній день завдання. (EN)
 - DB2 Express-C: гнучка й багатифункціональна СУБД, що підтримує реляційні дані й XML. (EN)
 - Apache Derby: відкрита реляційна СУБД, створена повністю на основі JavaТехнологій. (EN)
- Завантажити ознакомительные версии продуктов IBM, а також випробуйте засоби розробки додатків і проміжне програмне забезпечення IBM, зокрема DB2®, Lotus®, Rational®, Tivoli® і Websphere® на сайті online тестування IBM SOA Sandbox. (EN)

3. Складання звіту лабораторної роботи.

Для складання звіту лабораторної роботи необхідно надати наступні дані й відповісти на контрольні питання.

1. Мета
2. Умова завдання
3. Скриншот документа з робочими методами та класами.
4. Текст програми (Опис вхідних/вихідних параметрів+код)
4. Висновок

4. Контрольні питання

1. Що таке блог?
2. Де використовуються коментарі? Наведіть приклади
3. Що таке XML?
4. Поясніть структуру даних в XML?
5. Що таке JDBC4?
6. Які СУБД товаришують з XML, наведіть приклади ?
7. Навіщо потрібні об'єкти Source и Result з JAXP?
8. Що таке проміжний шар в програмному забезпеченні?
9. Що таке IBM WebSpher Application Server? Навіщо потрібні такі сервери?
10. Де можна використовувати подібний додаток?

Список літератури

- 1 Andreas Eide, «Professional ASP.NET Web Services», Wrox Press, 2001
- 2 Michael Stiefel, «Application Development Using C# and .NET» Prentice Hall, 2001
- 3 Andrew Krowczyk, «C# Web Services. Building WS with .Net Remoting and ASP.Net», Wrox Press, 2001
- 4 www.rsdn.ru
- 5 www.wmsdn.com
- 6 www.wmsdntnet.com
- 7 П.Б. Храмцов, С.А.Брик, А.М. Русак, А.И.Сурин Основы WEB-технологий. – М.: ИТУИТ.РУ, 2003. – 512 с.
- 8 Д.Роджерс Программирование на Microsoft JScript.NET. “Вильямс”, 2002. – 352 с.
- 9 В.В. Дунаев JavaScript.— СПб. : “Питер”, 2003 .— 394 с.
- 10 Б. Форта Освой самостоятельно регулярные выражения. 10 минут на урок. – М.: “Вильямс”, 2005. – 184 с.
- 11 К.Мельтцер Разработка CGI-приложений на Perl . М. : “Вильямс”, 2001 .— 395 с.
- 12 И.Шапошников PHP 5.1 : учебный курс — СПб [и др.] : “Питер”, 2007 .— 192 с.
- 13 Д.Шеперд Освой самостоятельно XML за 21 день. – М.: “Вильямс”, 2002. – 432 с.
- 14 А.Старыгин XML: разработка Web-приложений.— СПб. : “БХВ-Петербург”, 2003 .— 585 с.
- 15 Э.Троелсен C# и платформа .NET. Библиотека программиста. – СПб.: “Питер”, 2007.- 796 с.
- 16 М.Беллиньясо Разработка Web-приложений в среде ASP.NET 2.0: с примерами на C#. “Диалектика”, 2007. - 640 с.
- 17 С.Шорт Разработка XML Web-сервисов средствами Microsoft .NET.— СПб. : “БХВ-Петербург”, 2003 .— 480 с.
- 18 Э.Ньюкомер Веб-сервисы : XML, WSDL, SOAP и UDDI. — СПб. : Питер, 2003.
- 19 И.Салмре Программирование мобильных устройств на платформе .NET Compact Framework. “Вильямс”, 2006. - 736 с.
- 20 Д.Крейн, Э.Паскарелло, Д. Джеймс AJAX в действии: технология - Asynchronous JavaScript and XML = AJAX in Action. — М.: “Вильямс”, 2006. — С. 640.
- 21 Wikipedia, the free encyclopedia (http://en.wikipedia.org/wiki/Main_Page).
- 22 Википедия (<http://ru.wikipedia.org/wiki/>)
- 23 W3CSchools Online Web Tutorials (<http://www.w3schools.com/>)
- 24 Интернет Университет информационных технологий (<http://www.intuit.ru/catalog/>)