МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Донбаська державна машинобудівна академія

МЕТОДИЧНІ ВКАЗІВКИ

до практичних робіт

з дисципліни

«РОЗПОДІЛЕНІ КОМП'ЮТЕРНІ СИСТЕМИ ТА МЕРЕЖІ»

для студентів спеціальності 123

«Комп’ютерна інженерія»

Затверджено
на засіданні кафедри АВП
Протокол № 3
від „06” листопада 2017 року

Краматорськ 2017

УДК 004

Методичні вказівки до практичних робіт з дисципліни «Розподілені комп'ютерні системи та мережі» для студентів спеціальності 123 «Комп’ютерна інженерія» / Укладач О.В. Суботін - Краматорськ: ДДМА, 2017.- 32 с.

Методичні вказівки містять теоретичні відомості, рекомендації до виконання і вимоги до оформлення практичних робіт, а також питання для самостійної роботи студентів спеціальності 123 «Комп’ютерна інженерія» з дисципліни «Розподілені комп'ютерні системи та мережі».

Укладач: О.В. Суботін, доцент.

Відповідальний за випуск: О.В. Суботін, доцент.

Практична робота №1
Хмарні сховища даних. Вибір під конкретну задачу.

Мета роботи: отримати уявлення про характер і рівень можливостей доступних хмарних сховищ даних.

Теоретичні відомості

Хмарне сховище даних - модель онлайн-сховища, в якому дані зберігаються на численних розподілених в мережі серверах, що надаються в користування клієнтам, в основному, третьою стороною. На противагу моделі зберігання даних на власних виділених серверах, придбаних або орендованих спеціально для подібних цілей, кількість або будь-яка внутрішня структура серверів клієнту, в загальному випадку, хоч я знаю. Дані зберігаються, а так само і обробляються, в так званому хмарі, яке представляє собою, з точки зору клієнта, один великий віртуальний сервер. Фізично ж такі сервери можуть розташовуватися віддалено один від одного географічно, аж до розташування на різних континентах.

Переваги хмарних сховищ. Клієнт платить тільки за те місце в сховищі, яке фактично використовує, але не за оренду сервера, всі ресурси якого він може і не використовувати.

Клієнту немає необхідності займатися придбанням, підтримкою і обслуговуванням власної інфраструктури зі зберігання даних, що, в кінцевому рахунку, зменшує загальні витрати виробництва.

Всі процедури по резервуванню і збереженню цілісності даних виробляються провайдером хмарного центру, яка не втягує в цей процес клієнта.

Потенційні проблеми. Безпека при зберіганні і пересилання даних є одним з основних питань при роботі з хмарою, особливо щодо конфіденційних, приватних даних.

Загальна продуктивність при роботі з даними в хмарі може бути нижче такої при роботі з локальними копіями даних.

Надійність і своєчасність отримання і доступності даних в хмарі дуже сильно залежить від багатьох проміжних параметрів, в основному таких як канали передачі даних на шляху від клієнта до хмари, питання останньої милі, питання про належну якість роботи інтернет-провайдера клієнта, питання про доступність самої хмари в даний момент часу.

Хмарні шлюзи. Хмарні шлюзи - технологія, яка може бути використана для більш зручного представлення хмари клієнту. Наприклад, за допомогою відповідного програмного забезпечення, сховище в хмарі може бути представлено для клієнта як локальний диск на комп'ютері. Таким чином, робота з даними в хмарі для клієнта стає абсолютно прозорою. І при наявності гарної, швидкого зв'язку з хмарою клієнт може навіть не помічати, що працює не з локальними даними у себе на

комп'ютері, а з даними, що зберігаються, можливо, за багато сотень кілометрів від нього.

Показники для оцінки хмарних сховищ:

- ціна / безкоштовність різних видів послуг;
- обсяг інформації, що зберігається (в різних конфігураціях);
- стабільність роботи і репутація;
- простота і швидкість здійснення елементарних операцій по приміщенню файлу в сховище;
- гнучкість і розвиненість можливостей з управління доступом;
- простота управління і конфігурації (політика безпеки і розмежування доступу, простота підключення нових послуг);
- наявність шлюзів і клієнтських додатків під різні ОС;
- використовувані алгоритми шифрування і інші засоби управління безпекою;
- наявність (і можливості) API для розробки своїх додатків.

Порядок виконання роботи

1 Вивчіть теоретичні відомості.

2 Ознайомтеся з існуючими хмарними сховищами даних.

3 Підберіть відповідне хмарне сховище даних для виконання завдання відповідно до варіанта індивідуального завдання.

Виберіть декілька найбільш підходящих сховищ з існуючих. Сформулюйте критерії оцінки. Призначте і обґрунтуйте ваги критеріїв з точки зору важливості для вирішення вашого завдання. Проаналізуйте маркетингову документацію обраних сховищ та відгуки про них в інтернеті. На підставі аналізу оцініте кожне з обраних сховищ по кожному з критеріїв. Розрахуйте сумарну оцінку і на її підставі виберіть сховище.

4 Увійдіть на обраному сервісі і зробите найпростіші операції по користуванню сервісом, супроводжуючи процес зняттям скріншотів.

Створіть файл, назва якого збігається з вашим прізвищем. Помістіть його в хмарне сховище. Перевірте його наявність в сховище і скачайте його назад на свій комп'ютер. Зробіть висновки по роботі, оформіть звіт, підготуйтеся до захисту.

Зміст звіту

1 Прізвище, ім'я, по батькові, група, тема, мета.

2 Індивідуальне завдання.

3 Список обраних критеріїв з призначеними вагами, обґрунтування ваг.

4 Список хмарних сервісів для порівняння, короткий опис кожного.

5 Таблиця оцінок сервісів за критеріями. Коротке обґрунтування оцінок. Розрахунок сумарної оцінки для кожного сервісу.

6 Скріншоти реєстрації на сервісі і його використання з коментарями.

7 Висновки по роботі.

Індивідуальні завдання

№	Умова
1	Тільки для особистого користування. Кілька мегабайтів документів повинні бути доступні з дому, с роботи і з мобільного пристрою під Android.
2	Сховище методичних вказівок вузу. Повинна бути система папок з дисциплін, змінювати вміст яких може тільки адміністратор і відповідальний викладач. Доступ на читання повинні мати студенти звідусіль. Бажано, дешевше.
3	Терабайтовий архів робіт невеликого колективу розробників ПЗ з численними версіями і гілками.
4	Сховище великої кількості відео і аудіо інформації стабільною групою осіб в 40 тільки для своїх.
5	Корпоративний файл-сервер для обміну бізнес-інформацією. Важлива стабільність і безпеку.
6	Сховище архівів і зображень для використання сторонніми сайтами за прямими посиланнями через HTTP.
7	Зберігання резервних копій особистої інформації та образів системного диска.
8	Сховище резервних копій корпоративної БД. Оновлюватиметься за допомогою скриптів.
9	Сховище нормативних актів міністерства, відкрите на читання всім бажаючим. Можливо вибуховий збільшення трафіку при оновленнях. Бажана можливість автоматизації пошуку по тексту зберігаються документів.
10	Зберігання великого масиву тестових даних для регулярного тестування нових версій програмного продукту.

Питання для контролю та самостійної роботи

1 Що таке «хмарні технології»? Які цілі і причини їх використання?
2 Які переваги та недоліки хмарних сховищ даних?
3 Які хмарні сховища даних ви знаєте?
4 Які можливості надає кожне з них?
5 Яка їхня порівняльна популярність?
6 Що таке Directory Service (Служба каталогів)? Які операції можуть бути доступні через служби каталогів хмарних сховищ даних?
7 Які хмарні сховища даних надають API для роботи зі своїми службами каталогів?
8 Яким чином можуть бути реалізовані ці API? Які мережеві протоколи для цього використовуються? До яких рівнями OSI відносяться ці протоколи? Чи не порушується ідеологія OSI таким їх використанням?

Практична робота №2
Підключення веб-сервісів.

Мета роботи: Отримати найпростіші поняття про програмування в рамках сервісно-орієнтованої архітектури.

Теоретичні відомості

Сервісно-орієнтована архітектура. Сервісно-орієнтована архітектура - це парадигма, призначена для проектування, розробки та управління дискретних одиниць логіки (сервісів) в обчислювальному середовищі.

Дана концепція грунтується на архітектурному стилі, який визначає модель взаємовідносин між трьома основними сторонами - постачальником, споживачем і посередником сервісу. Постачальник сервісу публікує опис сервісу і забезпечує його реалізацію. Споживач сервісу для знаходження опису сервісу може безпосередньо використовувати універсальний ідентифікатор ресурсу (URI) або ж може знайти опис в реєстрі сервісу, з подальшою прив'язкою і викликом сервісу. Посередник сервісу забезпечує і обслуговує реєстр сервісу.

Метамодель, що представляє ці відносини відображена на рисунку 1.

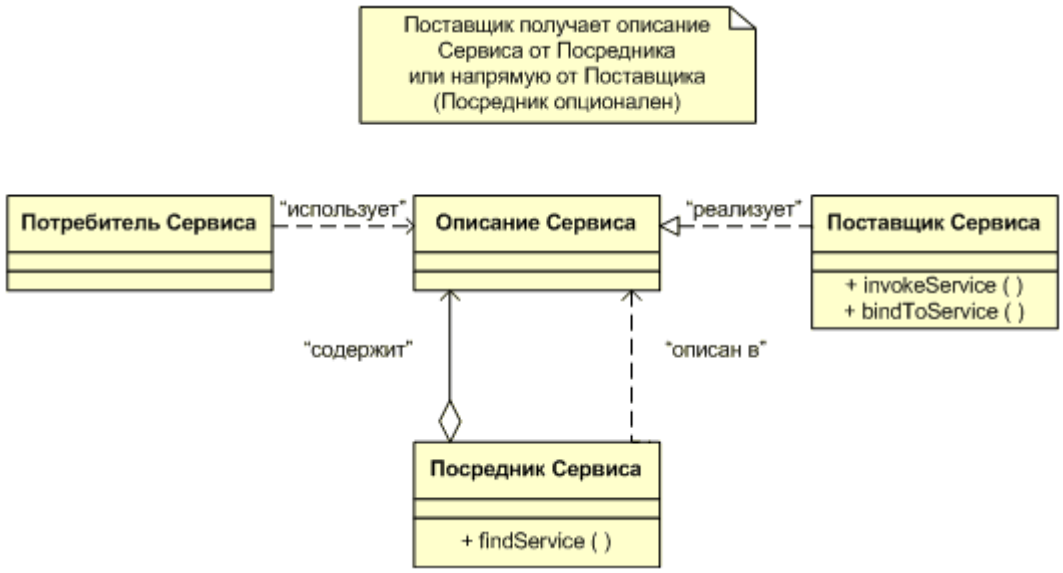


Рисунок 1 - Концептуальна модель архітектурного стилю SOA

Мова опису WSDL. WSDL (англ. Web Services Description Language) - мова опису веб-сервісів і доступу до них, заснований на мові XML. Остання офіційна специфікація версія 2.0 (WSDL Version 2.0 від 26 червня 2007 року).

Кожен документ WSDL можна розбити на наступні логічні частини:

– визначення типів даних (types) - визначення виду відправлених і отриманих сервісом XML повідомлень

– елементи даних (message) - повідомлення, що використовуються web-сервісом

– абстрактні операції (portType) - список операцій, які можуть бути виконані з повідомленнями

– зв'язування сервісів (binding) - спосіб, яким повідомлення буде доставлено

Створення клієнта веб-сервісу в Visual Studio. Створіть проект (підійде і консольний додаток).

Клацанням правої клавіші миші на проекті викличте контекстне меню і виберіть пункт «Додати посилання на сервіс» ("Add Service Reference").

Введіть шлях до файлу WSDL (це може бути URL) і ім'я сервісу (наприклад «YourService») і імпортуйте його. Це створить найпростіший клієнт сервісу. Якщо імпорт проведено успішно, в створеному коді можна буде знайти клас «YourservicenameClient», в якому будуть прописані методи для кожної функції, визначеної в контракті WSDL. Тепер посилання на сервіс і його методи можна буде знайти в інспекторові об'єктів.

Інстанціруйте клієнт і викличте необхідні методи.

```
YourServiceClient client = new YourServiceClient();
client.SayHello("World!");
```

Якщо потрібно використовувати віддалений URL (який не збігається з зазначеним за замовчуванням), можна вказати його в конструкторі клієнта:

```
YourServiceClient client = new YourServiceClient("ConfigName",
"RemoteURL");
```

де "configName" - ім'я кінцевої точки (відомості про кінцевих точках містяться в файлі WSDL, звідти будуть взяті всі настройки, окрім URL), а "remoteURL" - рядок URL для з'єднання (замість зазначеної в config).

Також для генерації коду клієнта на C # можна використовувати утиліту WSDN.EXE доступну на сайті для розробників Microsoft (http://msdn.microsoft.com/ru-ru/library/7h3ystb6.aspx).

Порядок виконання роботи

1. Вивчіть теоретичні відомості.
2. Знайдіть простий і безкоштовний веб-сервіс (наприклад на сайтах free-web-services.com або webservicelist.com, тисячі їх).
3. Напишіть просте додаток, що використовує можливості сервісу.
4. Зробіть висновки по роботі, оформіть звіт, підготуйтеся до захисту.

Зміст звіту

1. Прізвище, ім'я, по батькові, група, тема, мета.
2. Опис функцій сервісу.
3. Вміст WSDL-документа (можливо, частково - опис використаних функцій).
4. Текст програми.
5. Скріншоти тестового прогону.
6. Висновки по роботі.

Питання для самоперевірки

1 Які переваги та недоліки сервісно-орієнтованого підходу при написанні бізнес-додатків?

2 Що нове додає сервісно-оріентірованній підхід в порівнянні з компонентним програмуванням?

3 Які протоколи і формати даних використовуються при організації та використанні веб-сервісів?

4 Чим пояснюється широке використання XML і заснованих на ньому форматів в СОА.

5 Перерахуйте принципи організації СОА.

6 В чому полягає суть слабкого зв'язування, повторного використання, абстракції, комбіновані, автономності в застосуванні до сервісів.

7 Поясніть поняття контракту, кінцевої точки, зв'язку (binding).

Додаткова література

http://www.ibm.com/developerworks/ru/library/ws-soa-design1/
http://serviceorientation.com/static/pdf/SOA_Principles_Poster.pdf
http://www.intuit.ru/department/itmngt/entarc/7/7.html

Практична робота №3
Розробка веб-сервісів.

Мета роботи: Розробити власний веб-сервіс, викликати його локально.

Теоретичні відомості

Створення сервісів в .NET під на C #. Нижче описаний покроковий процес створення найпростішого веб-сервісу в .NET з встановленим SDK. Можете використовувати цей механізм або за бажанням розібратися з роботою WCF, що може виявитися складніше.

Наберіть в будь-якому текстовому редакторі наступний код і збережіть файл як hello.asmx.

```
<% @ WebService Language = "C #" class = "HelloService"%>
using System;
using System.Web.Services;
[WebService (Namespace = "http://abcd.org")]
class HelloService {
 [WebMethod]
 public string SayHello () {
 return "Hello World";
 }
}
```

Директива WebService вказує, що клас буде використаний в якості веб-сервісу. Тут же вказується мова. У нашому випадку це C #.

```
<% @ WebService Language = "C #" class = "HelloService"%>
```

Наступний пункт - підключіть необхідні простору імен, System.Web.Services - обов'язково.

```
[WebService (Namespace = "http://abcd.org")]
```

Цей рядок необов'язкова. Вона служить для запобігання можливих конфліктів. Якщо вона опущена, за замовчуванням значення буде "http://tempuri.org". Однак, якщо сервіс буде дійсно використовуватися через інтернет, її краще прописати.

Нарешті, позначте який-небудь метод вашого класу як [WebMethod] безпосередньо над методом.

Якщо ви помістіть цей файл на ваш локальний веб сервер (передбачається використання IIS), він буде доступний за адресою

http: //localhost/hello.asmx

через браузер. Натисніть на «SayHello», щоб отримати XML-пакет, який є відповіддю на запит сервісу.

Щоб використовувати веб-сервіс, потрібно створити клас-заступник з використанням утиліти wsdl.exe, що входить в комплект засобів розробки платформи .NET (.NET framework SDK), доступний на сайті Microsoft (якщо у вас він ще не встановлений). Наберіть в командному рядку

wsdl "http: //localhost/hello.asmx" /out:hello.cs

Після цього в цій папці у вас з'явиться файл hello.cs. Тепер для використання його потрібно відкомпілювати в DLL:

csc hello.cs / t: library /out:hello.dll
/r:System.Web.Service.dll /r:System.Xml.dll

Не забудьте підключити бібліотеки System.Web.Service.dll і System.Xml.dll.

Помістіть вийшов DLL в папку / bin на вашому веб-сервері. Тепер сервіс можна використовувати, наприклад, за допомогою коду ASP.NET:

```
<% @ Page Language = "C #"%>
<Script language = "C #" runat = "server">
private void Page_Load (Object sender, EventArgs e) {
 HelloService hs = new HelloService ();
 Label1.Text = hs.SayHello ();
}
</ Script>
<Html>
<Head>
<Title> Hello WebService </ title>
</ Head>
<Body>
<Asp: Label id = "Label1" runat = "server" />
</ Body>
</ Html>
```

Порядок виконання роботи

1 Вивчіть теоретичні відомості.

2 Створіть свій веб-сервіс з простою функцією, що повертає строкове значення.

3 Перевірте роботу створеного сервісу.

4 Зробіть висновки по роботі, оформіть звіт, підготуйтеся до захисту.

Додаткова література

http://www.realcoding.net/article/view/2519
http://www.csharpfriends.com/Articles/getArticle.aspx?articleID=35
http://habrahabr.ru/post/116764/

Троелсен Е. / Мова програмування C # 2010 і платформа .NET 4 // Вільямс 2010 ISBN: 978-5-8459-1682-2, 978-1-43-022549-2

Практична робота №4
Підключення веб-сервісів.

Мета роботи: Отримати найпростіші поняття про програмування в рамках сервісно-орієнтованої архітектури.

Теоретичні відомості

Створення сервісів в .NET. Під час використання веб-сервісів загальною проблемою є безпека: повідомлення SOAP передаються в простому тексті по мережі, тому кожен, хто має сніфер, може перехопити повідомлення SOAP та прочитати його. На мою думку, це може статися і з бінарними даними, але, ймовірно, для цього потрібно трохи більше хакерських навичок. Таким чином, рішення - використовувати HTTPS (SSL) замість HTTP, тому зв'язок шифрується.

Для цього нам потрібно зробити наступні дії:

– встановити сертифікат (виданий органом сертифікації) на веб-сервері;

– створіть веб-сервіс за допомогою .net;

– налаштуйте веб-сервіс для використання SSL в IIS;

– зателефонуйте до Клієнта веб-служби;

– встановіть сертифікат.

Зазвичай у виробничому середовищі нам потрібно купувати сертифікат у Verisign або будь-якого іншого добре відомого органу з сертифікації (CA). У середовищі розробки ми можемо використовувати .net SDK MakeCert.exe, який поставляється з .net 2.0 SDK.

Ввести до Exe:

/ Program Files / Microsoft Visual Studio 8 / SDK / v2.0 / bin / MakeCert.exe

Створіть та встановіть сертифікат:

Makecert -sr LocalMachine -ss My -n CN = ServiceModelSamples-HTTPS-Server -sky exchange -sk ServiceModelSamples-HTTPS-Key

Тепер сертифікат створюється та призначається в локальній машині.

Створіть веб-сервіс за допомогою .net.

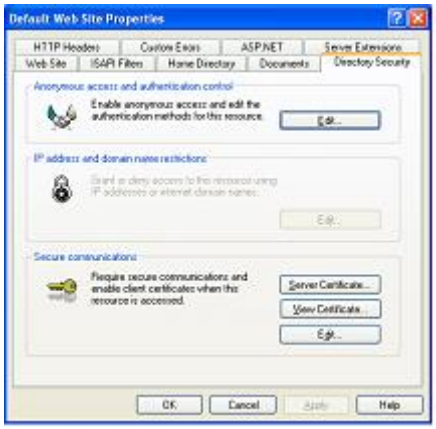
За допомогою VS2005 або 2008 створіть веб-сервіс із зразком веб-методу AddValues, який має два цілих параметри і повертає ціле значення.

```
[WebMethod]public int AddValues (int a, int b).
{
return a + b;
}
```

Тепер якщо ви переглянете Службу з IIS, вона покаже метод AddValues.

Настройте веб-сервіс для використання SSL в IIS:

– введіть inetmgr у діалоговому вікні «Запустити» та відкрийте IIS;
– на властивості веб-сайту за замовчуванням виберіть вкладку. Безпека каталогу.



У розділі Захищене спілкування виберіть Сертифікат сервера.

У спливаючому Майстрі виберіть призначення існуючого сертифіката.

На наступному кроці виберіть наш сертифікат ServiceModelSamples-HTTPS-сервер і закінчіть майстра.

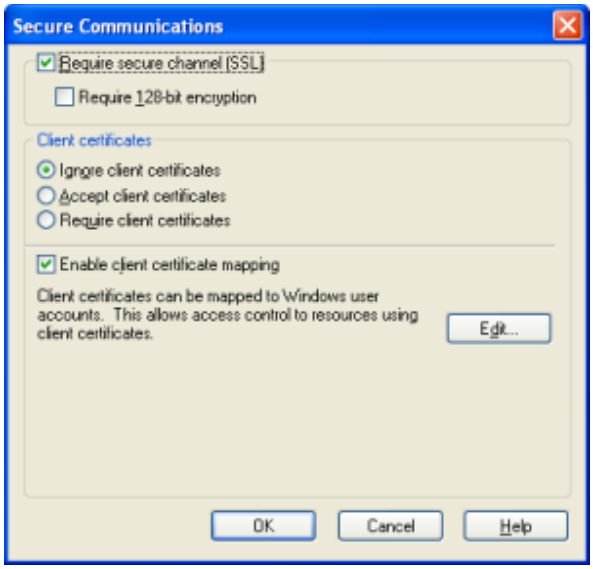
Тепер сертифікат встановлений. Тепер кнопка сертифіката перегляду стає включеною та

ви можете переглянути сертифікат.

Тепер R.click ваш віртуальний каталог веб-служби у IIS та виберіть його властивості.

У діалоговому вікні "Властивості" виберіть вкладку "Захист каталогів".

У розділі Безпечне спілкування натисніть кнопку Редагувати.



Поставте прапорець Потрібний безпечний канал (SSL).
Натисніть OK та вийдіть із властивостей.
Перевірте веб-сервіс на наявність HTTPS.
Тепер у веб-переглядачі перейдіть до своєї веб-служби як

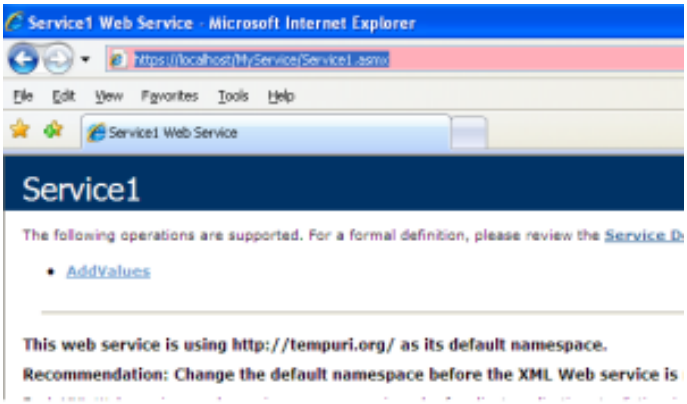
http://localhost/MyService/Service1.asmx

Веб-переглядач покаже "Сторінку потрібно переглядати через захищений канал".
Тепер у браузері змініть адресу адреси на

https://localhost/MyService/Service1.asmx

У браузері відобразиться наступний екран:

Клацніть Продовжити на цьому веб-сайті (не рекомендується).



Створіть клієнта для використання методу веб-сервісу.
Відкрийте Visual Studio і створіть веб-додаток ASP.Net.
У веб-довідці додайте посилання на наш веб-сервіс.
Введіть URL та призначте ім'я посиланню:

https://localhost/MyService/Service1.asmxin

Тепер у завантаженні сторінки зателефонуйте до служби:

```
Service.Service1 myservice = new
ServiceHandler.Service.Service1 ();
int result = myservice.AddValues (10,11);
Response.Write (result.ToString ());
```

Якщо ви запустите програму, ви отримаєте помилку перевірки. "Віддалений сертифікат недійсний відповідно до процедури перевірки."

Resolution:
Add a new Class file to your project with the name MyCertificatePolicy
Add the following using Statement.

```
using System;
using System.Net;
using System.Security.Cryptography.X509Certificates;
```

Inherit System.Net.ICertificatePolicy to the class
Add the method to the class below the constructor

```
public bool CheckValidationResult (ServicePoint sp,
X509Certificate cert,
WebRequest req,
int problem)
{
return true;
}
```

У PageLoad додайте код нижче, перш ніж ініціалізувати веб-службу.

```
System.Net.ServicePointManager.CertificatePolicy =
new MyCertificatePolicy ();
Service.Service1 myservice =
new ServiceHandler.Service.Service1 ();
int result = myservice.AddValues (10,11);
Response.Write (result.ToString ());
```

Тепер запустіть проект. Вихід 21 відобразиться на сторінці.

Порядок виконання роботи

1. Вивчіть теоретичні відомості.
2. Знайдіть простий і безкоштовний веб-сервіс (наприклад на сайтах free-web-services.com або webservicelist.com, тисячі їх).
3. Напишіть просте додаток, що використовує можливості сервісу.
4. Зробіть висновки по роботі, оформіть звіт, підготуйтеся до захисту.

Зміст звіту

1. Прізвище, ім'я, по батькові, група, тема, мета.
2. Опис функцій сервісу.
3. Вміст WSDL-документа (можливо - опис використаних функцій).
4. Текст програми.
5. Скріншоти тестового прогону.
6. Висновки по роботі.

Питання для самоперевірки

1. Які переваги та недоліки сервісно-орієнтованого підходу при написанні бізнес-додатків?

2. Що нове додає сервісно-оріентірованній підхід в порівнянні з компонентним програмуванням?

3. Які протоколи і формати даних використовуються при організації та використанні веб-сервісів?

4. Чим пояснюється широке використання XML і заснованих на ньому форматів в СОА.

5. Перерахуйте принципи організації СОА.

6. В чому полягає суть слабкого зв'язування, повторного використання, абстракції, комбіновані, автономності в застосуванні до сервісів.

7. Поясніть поняття контракту, кінцевої точки, зв'язку (binding).

Додаткова література

http://rmanimaran.wordpress.com/2010/06/24/creating-and-using-c-web-service-over-https-%E2%80%93-ssl-2/

СПИСОК ЛІТЕРАТУРИ

1. Аллен, Э. Типичные ошибки проектирования. Библиотека программиста / Э. Аллен. – СПб.: Питер, 2002. – 424 с.: ил. – ISBN 5-88782-304-6.

2. Астелс, Д. Практическое руководство по экстремальному программированию / Д. Астелс, Г. Миллер. – К.: Диалектика, 2002. – 320 с.: ил. – ISBN 5-8459-0329-7.

3. Бек, К. Экстремальное программирование / К.Бек. – СПб.: Питер, 2002. – 224 с.: ил. – ISBN 5-94723-032-1.

4. Бек, К. Экстремальное программирование: разработка через тестирование / К. Бек. – СПб.: Питер, 2003. – 240 с.: ил. – ISBN 5-8046-0051-6.

5. Брукс, Ф. Мифический человеко-месяц, или Как создаются программные системы / Ф. Брукс ; пер. с англ. – СПб. : Символ-Плюс, 1999. – 324 с.: ил. – ISBN 5-93286-005-7.

6. Влиссидес, Дж. Применение шаблонов проектирования / Дж. Влиссидес. – К.: Диалектика, 2003. – 144 с.: ил. – ISBN 5-8459-0393-9.

7. Гамма, Э. Приемы объектно-ориентированного программирования. Паттерны / Э. Гамма, Р. Хелм. – СПб.: Питер, 2003. – 368 с.: ил. – ISBN 5-272-00355-1.

8. Йордон, Э. Путь камикадзе / Э. Йордон. – М.: Лори, 2001. – 244 с.: ил. – ISBN 5-85582-227-3.

9. Канер, С. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений / С. Канер, Дж. Фолк, Е. К. Нгуен. – К.: ДиаСофт, 2001. – 544 с.: ил. – ISBN 966-7393-87-9.

10. Кендалл, С. Унифицированный процесс. Основные концепции / С. Кендалл. – К.: Диалектика, 2002. – 160 с.: ил. – ISBN 5-8459-0346-7.

11. Коберн, А. Быстрая разработка программного обеспечения / А.Коберн. – М.: Лори, 2002. – 314 с.: ил. – ISBN 0-201-69969-9.

12. Коберн, А. Современные методы описания функциональных требований к системам / А. Коберн. – М.: Лори, 2002. – 263 с.: ил. – ISBN 5-85582-152-8.

13. Ковалёв, А. Управление проектом по созданию Интернет-сайта / А. Ковалёв, И. Курдюмов. – М.: Альпина Паблишер, 2001. – 337с.: ил. – ISBN 5-94599-007-8.

14. Крачтен, Ф. Введение в Rational Unified Process / Ф. Крачтен. – К.: Диалектика, 2002. – 240 с.: ил. – ISBN 5-8459-0239-8.

15. Ларман, К. Применение UML и шаблонов проектирования / К. Ларман. – М.: Вильямс, 2002. – 624 с.: ил. – ISBN 5-8459-0125-1.

16. Леффингуэлл, Д. Принципы работы с требованиями к программному обеспечению. Унифицированный подход / Д. Леффингуэлл. – К.: Диалектика, 2002. – 244 с.: ил. – ISBN 5-8459-0275-4.

17. Орлов, С.А. Технологии разработки программного обеспечения / С. А. Орлов. – СПб.: Питер, 2003. – 480 с.: ил. – ISBN 5-94723-145-X.

18. Петцольд, Ч. Код. Тайный язык информатики / Ч. Петцольд. – М.: Русская редакция, 2001. – 512 с.: ил. – ISBN 5-7502-0159-7.

19. Рамбо, Дж. UML. Специальный справочник / Дж. Рамбо, А. Якобсон, Г. Буч. – СПб.: Питер, 2002. – 656 с.: ил. – ISBN 5-318-00174-2.

20. Себеста, Роберт У. Основные концепции языков программирования / Роберт У. Себеста.– М.: Вильямс, 2001. – 672с.: ил. – ISBN 5-8459-0192-8.

21. Смит, К.У. Эффективные решения: практическое руководство по созданию гибкого и масштабируемого программного обеспечения / К. У. Смит, Л. Уильямс – К.: Диалектика, 2003. – 448 с.: ил. – ISBN 5-8459-0448-X.

22. Соммервилл, И. Инженерия программного обеспечения / И. Соммервилл. – М.: Вильямс, 2002. – 624 с.: ил. – ISBN 5-8459-0330-0.

23. Тейт, Б. Горький вкус Java. Библиотека программиста / Б. Тейт. – СПб.: Питер, 2003. – 333 с.: ил. – ISBN 5-88782-323-2.

24. Трофимов, С.А. Case-технологии: практическая работа в Rational Rose / С. А. Трофимов. – М.: Бином-Пресс, 2002. – 288 с.: ил. – ISBN 5-9518-0001-3.

25. Уилсон, С. Принципы проектирования и разработки программного обеспечения. Учебный курс MCSD / С. Уилсон. – М.: Русская редакция, 2002. – 736 с.: ил. – ISBN 5-7502-0213-5.

26. Фаулер, М. Рефакторинг: улучшение существующего кода / М. Фаулер, К. Бек. – М.: Символ, 2003. – 432 с.: ил. – ISBN 5-93286-045-6.

27. Фаулер, С. UML. Основы / С. Фаулер. – М.: Символ, 2002. – 190 с.: ил. – ISBN 5-93286-032-4.

28. Шаллоуей, А. Шаблоны проектирования. Новый подход к ООП и анализу / А. Шаллоуей, Дж. Тротт. – К.: Диалектика, 2002. – 288 с.: ил. – ISBN 5-8459-0301-7.

29. Шафер, Д.Ф. Управление программными проектами: достижение оптимального качества при минимуме затрат / Д. Ф. Шафер, Р. Т. Фатрелл. – К. Диалектика, 2003. – 1136 с.: ил. – ISBN 5-8459-0413-7.

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ

до практичних робіт

з дисципліни

«РОЗПОДІЛЕНІ КОМП'ЮТЕРНІ СИСТЕМИ ТА МЕРЕЖІ»

для студентів спеціальності 123

«Комп’ютерна інженерія»

Укладач: Суботін Олег Володимирович

.

26/2006 підп. до друку Формат 60х84 / 16
Папір офсетний. Розум. друк.арк. Обл.-від.арк. .
Тираж прим. Зам.№

Видавець и виготівник
«Донбаська державна Машинобудівна академія»
84313, м. Краматорськ, вул. Шкадінова, 72
Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру
серія ДК №1633 від 24.12.2003 р.