МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
БЕРДЯНСЬКИЙ ДЕРЖАВНИЙ ПЕДАГОГІЧНИЙ УНІВЕРСИТЕТ

Сергій ОНИЩЕНКО

Методичні рекомендації до виконання лабораторних робіт з дисципліни «WEB-ТЕХНОЛОГІЇ»

Рекомендовано Радою факультету фізико-математичної і технологічної освіти Бердянського державного педагогічного університету Протокол №5 від 23 грудня 2015 року

Бердянськ
«БДПУ»
2015

Рекомендовано Радою факультету
фізико-математичної і технологічної освіти
Бердянського державного педагогічного університету
Протокол №5 від 23 грудня 2015 року

Рецензенти:
Єфименко Ю. О. – кандидат педагогічних наук, доцент Бердянського державного педагогічного університету.
Буянов П. Г. – кандидат педагогічних наук, доцент Бердянського державного педагогічного університету.

Онищенко С. В.
WEB-технології : метод. реком. до вик. лаб. роб. / Сергій Онищенко. – Бердянськ : БДПУ, 2015. – 181 с.

У методичних рекомендаціях подані: навчальна програма, лабораторні роботи з дисципліни "WEB-технології". Призначений для підготовки студентів спеціальності 6.010104 – професійна освіта денної форми навчання.

ВСТУП

WEB-технології є концепція роботи з інформацією. Вона відрізняється наступними особливостями:

- технічна основа веб-технологій – локальні і глобальні мережі, часто Інтернет;
- застосування особливого типу тонких клієнтів: веб-браузерів (типи та історія, сучасний стан браузерів віддаються на самостійне вивчення);
- переважно текстова і статично-графічна подача інформації (ослаблення цієї тенденції пов'язано з розвитком технологій зв'язку і ПЗ, експансією медіаконтенту);
- зміни в інформаційних джерелах миттєво відображаються в публікаціях;
- число споживачів інформації практично не обмежена. Публікатор сам може задати особливі умови на доступ до інформації, що публікується;
- в публікаціях можуть міститися посилання на інші публікації без обмеження на розташування і джерела матеріалів;
- активна робота пошукових машин (історія, сучасний стан та роль пошукових машин віддаються на самостійне вивчення);
- доставка і тиражування контенту практично безкоштовні.

Привабливість WEB-технологій як засобу доставки інформації багато в чому визначає універсальний інтерфейс між людиною і комп'ютером. Кожній людині зрозумілі написи, заголовки, посилання, картинки. WEB-інтерфейс як засіб доступу до інформації інтуїтивно зрозумілий. Наслідком простоти WEB-інтерфейсу є широко використання Інтернету як каналу комунікації. Браузер – програма для перегляду WEB-сторінок і роботи з інформацією в WEB-інтерфейсі. Браузери – програми, якими забезпечені всі сучасні комп'ютери. Теоретично всі браузери повинні відображати всі сайти, зроблені за стандартами, однаково. Практично є безліч тонкощів і складнощів. Найбільш популярні браузери: Internet Explorer, Firefox, Opera, Safari, Chrome.

Значення WEB-технології, як для розробників програмного забезпечення, так і для звичайних користувачів, визначається тим, що це, перш за все – інтеграційна технологія. І важко знайти більш вдалий приклад того, як можна інтегрувати різні джерела інформації і різні її типи. WEB-технології дозволяють створювати прості для освоєння, легкодоступні, вкрай дешеві, швидко обновлювані інформаційні, діалогові, довідкові системи.

Розгляду принципів роботи в мережі, методики розробки WEB-сайтів, WEB-додатків та мобільних додатків присвячений даний навчальний посібник.

1. РОБОЧА ПРОГРАМА ДИСЦИПЛІНИ

Опис навчальної дисципліни

<table>
<tr><td rowspan="2">Найменування показників</td><td rowspan="2">Галузь знань, напрям підготовки, освітньо-кваліфікаційний рівень</td><td colspan="2">Характеристика навчальної дисципліни</td></tr>
<tr><td>денна форма навчання</td><td>заочна форма навчання</td></tr>
<tr><td rowspan="2">Кількість кредитів – 3,5</td><td>Галузь знань
0101 Педагогічна освіта
(шифр і назва)</td><td colspan="2" rowspan="2">Нормативна
(за вибором)</td></tr>
<tr><td>Напрям підготовки
6.010104
Професійна освіта
(шифр і назва)</td></tr>
<tr><td>Модулів – 4</td><td rowspan="5">Спеціальність:
_ професійна освіта _</td><td colspan="2">Рік підготовки</td></tr>
<tr><td>Змістових модулів – 7</td><td>2-й</td><td>-й</td></tr>
<tr><td>Індивідуальне науково-дослідне завдання
_____

(назва)</td><td colspan="2">Семестр</td></tr>
<tr><td rowspan="2">Загальна кількість годин – 105</td><td>4-й</td><td>-й</td></tr>
<tr><td colspan="2">Лекції</td></tr>
<tr><td rowspan="11">Тижневих годин для денної форми навчання:
аудиторних – 3
самостійної роботи студента – 5</td><td rowspan="11">Освітньо-кваліфікаційний рівень:
6.010101 бакалавр
педагогічної освіти</td><td>26 год.</td><td>год.</td></tr>
<tr><td colspan="2">Практичні, семінарські</td></tr>
<tr><td>год.</td><td>год.</td></tr>
<tr><td colspan="2">Лабораторні</td></tr>
<tr><td>26 год.</td><td>год.</td></tr>
<tr><td colspan="2">Самостійна робота</td></tr>
<tr><td>53 год.</td><td>год.</td></tr>
<tr><td colspan="2">Індивідуальні завдання:</td></tr>
<tr><td colspan="2">год.</td></tr>
<tr><td colspan="2">Вид контролю:</td></tr>
<tr><td>екзамен</td><td></td></tr>
</table>

Примітка.

Співвідношення кількості годин аудиторних занять до самостійної і індивідуальної роботи становить (%):

для денної форми навчання – 52:53.

1. Мета та завдання навчальної дисципліни

Метою викладання навчальної дисципліни «Web-технології» є ознайомлення студентів із сучасними Internet засобами, формування системи знань і навичок, необхідних для освоєння спеціальної області знань, пов'язаної з WEB-технологіями.

Основними завданнями вивчення дисципліни «Web-технології» є:

- навчити основним засобам, що використовуються для створення WEB-технологій;
- пояснити структуру побудови всесвітньої мережі Інтернет;
- ознайомити з принципами механізмів взаємодії складових мережі;
- ознайомити з принципами організації інформаційних баз даних у мережі Інтернет;
- ознайомити з принципами організації інформаційних баз даних у мережі Інтернет;
- розвивати логічне мислення у студентів тощо.

Основні результати навчання і компетентності згідно з вимогами освітньо-професійної (освітньо-наукової) програми:

№ з/п	*Компетентності**	*Результати навчання*
1.	Загальні	Знати: структуру й принцип дії WEB-сторінок. Вміти: виконувати розробку базових клієнських та серверних сценаріів та додатків.
2.	Фахові	Знати: структуру й принцип роботи серверних технологій. Вміти: створювати найпростіші статичні та динамічні сайти.

** На кожну дисципліну обсягом до 5 кредитів ЕКТС обирають 2-3 компетентності з освітньо-професійної програми (профілю програми), обсягом до 10 кредитів – 3-4 компетентності, але не більше 4.*

2. Програма навчальної дисципліни

Змістовий модуль 1. Вступ до Web-технологій. Web-додатки.

Тема 1. Вступ до Web-технологій: структура і принципи Web.

Тема 2. Вступ до клієнт-сервісні технології Web. Протокол HTTP.

Тема 3. Клієнтські сценарії та додатки.

Тема 4. Серверні Web-додатки.

Змістовий модуль 2. Сценарії Perl і PHP.

Тема 5. Мови розробки сценаріїв Perl і PHP.

Тема 6. Вступ до C# і платформу Visual Studio.Net.

Змістовий модуль 3. Архітектура Web-додатків.

Тема 7. Архітектура Web-додатків ASP.NET. Розробка Web-додатків на платформі .NET.

Тема 8. Інтерфейси взаємодії Web-додатків з СУБД.

Змістовий модуль 4. Мова XML.

Тема 9. Вступ до XML.
Тема 10. Мови опису схем XML.
Тема 11. DOM XML. Перетворення XML документів.
Змістовий модуль 5. Робота з Web-контентом.
Тема 12. Інтеграція і взаємодія в мережі Web.
Тема 13. Організація процесу розробки Web-контента .CMS/CMF.
Тема 14. Синдикація та аггрегірування Web –контенту.
Змістовий модуль 6. Робота з Web-порталами та розробка Web-додатків.
Тема 15. Web-портали. Класифікація Web-порталів.
Тема 16. Вступ в технологію AJAX. Розробка мобільних Web-додатків.
Змістовий модуль 7. Мережа Web.
Тема 17. Вступ до Web 2.0.
Тема 18. Додатки для соціальних мереж.

3. Структура навчальної дисципліни

Назви змістових модулів і тем	Кількість годин											
	денна форма						Заочна форма					
	усього	у тому числі					усього	у тому числі				
		л	п	лаб	інд	с.р.		л	п	лаб	інд	с.р.
1	2	3	4	5	6	7	8	9	10	11	12	13
Модуль 1												
Змістовий модуль 1. Вступ до Web-технологій. Web-додатки.												
Тема 1. Вступ до Web-технологій: структура і принципи Web.	3	1		1		1						
Тема 2. Вступ до клієнт-сервісні технології Web. Протокол HTTP.	4	1		1		2						
Тема 3. Клієнтські сценарії та додатки.	4	1		1		2						
Тема 4. Серверні Web-додатки.	4	1		1		2						
Разом за змістовим модулем 1	15	4		4		7						
Змістовий модуль 2. Сценарії Perl і PHP.												
Тема 1. Мови розробки сценаріїв Perl і PHP.	7	2		2		3						
Тема 2. Вступ до C# і платформу	8	2		2		4						

Visual Studio.Net.												
Разом за змістовим модулем 2	15	4		4		7						
Усього годин	30	8		8		14						
Модуль 2												
Змістовий модуль 3. Архітектура Web-додатків.												
Тема 1. Архітектура Web-додатків ASP.NET. Розробка Web-додатків на платформі .NET.	7	2		2		3						
Тема 2. Інтерфейси взаємодії Web-додатків з СУБД.	8	2		2		4						
Разом за змістовим модулем 3	15	4		4		7						
Змістовий модуль 4. Мова XML.												
Тема 1. Вступ до XML.	5	1		1		3						
Тема 2. Мови опису схем XML.	5	1		1		3						
Тема 3. DOM XML. Перетворення XML документів.	5	2		2		1						
Разом за змістовим модулем 4	15	4		4		7						
Усього годин	30	8		8		14						
Модуль 3												
Змістовий модуль 5. Робота з Web-контентом.												
Тема 1. Інтеграція і взаємодія в мережі Web.	5	2		2		1						
Тема 2. Організація процесу розробки Web-контента .CMS/CMF.	5	1		1		3						

Тема 3. Синдикація та аггрегірування Web –контенту.	5	1		1		3						
Разом за змістовим модулем 5	15	4		4		7						
Змістовий модуль 6. Робота з Web-порталами та розробка Web-додатків.												
Тема 1. Web-портали. Класифікація Web-порталів.	7	2		2		3						
Тема 2. Вступ в технологію AJAX. Розробка мобільних Web-додатків.	8	2		2		4						
Разом за змістовим модулем 6	15	4		4		7						
Усього годин	30	8		8		14						
Модуль 4												
Змістовий модуль 7. Мережа Web.												
Тема 1. Вступ до Web 2.0.	7	1		1		5						
Тема 2. Додатки для соціальних мереж.	8	1		1		6						
Разом за змістовим модулем 7	15	2		2		11						
Усього годин	15	2		2		11						
Усього годин	**105**	**26**		**26**		**53**						

4. Теми лекцій

№ з/п	Назва теми	Кількість годин
		д/ф
1	Вступ до Web-технологій: структура і принципи Web.	1
2	Вступ до клієнт-сервісні технології Web. Протокол HTTP.	1
3	Клієнтські сценарії та додатки.	1
4	Серверні Web-додатки.	1

5	Мови розробки сценаріїв Perl і PHP.	2
6	Вступ до C# і платформу Visual Studio.Net.	2
7	Архітектура Web-додатків ASP.NET. Розробка Web-додатків на платформі .NET.	2
8	Інтерфейси взаємодії Web-додатків з СУБД.	2
9	Вступ до XML.	1
10	Мови опису схем XML.	1
11	DOM XML. Перетворення XML документів.	2
12	Інтеграція і взаємодія в мережі Web.	2
13	Організація процесу розробки Web-контента .CMS/CMF.	1
14	Синдикація та аггрегірування Web –контенту.	1
15	Web-портали. Класифікація Web-порталів.	2
16	Вступ в технологію AJAX. Розробка мобільних Web-додатків.	2
17	Вступ до Web 2.0.	1
18	Додатки для соціальних мереж.	1
	Усього годин	**26**

5. Теми лабораторних занять

№ з/п	Назва теми	Кількість годин
		д/ф
1	Службові утиліти для роботи в Інтернеті. Вивчення протоколу HTTP.	1
2	Принципи Web-дизайна. Знайомство з Microsoft Expression Web.	1
3	Вступ до JavaScript. Програмний взаємозв'язок з HTML документами на основі DOM API.	1
4	Клієнтські сценарії. Використання регулярних виразів.	1
5	Розробка CGI-додатків на Perl і PHP.	2
6	Знайомство з середовищем розробки Microsoft Visual Studio.NET. Структура програми на C#. Основи мови C#.	2
7	Основи мови C#. Робота з масивами і строками. Інтерфейси та колекції.	2
8	Основи розробки Web-додатків за допомогою ASP.NET.	2
9	Серверні елементи управління ASP.NET.	1
10	Робота з джерелами даних у ASP.NET.	1
11	Структура XML документа. XML схеми.	2
12	Програмна обробка XML документів за допомогою XML DOM.	2
13	Формування та перетворення XML документу за допомогою XSL. XSLT перетворення XML документу.	1
14	Розробка Web-служби в ASP.NET.	1
15	Розробка Web-служби в ASP.NET. Створення проксі-зборки для Web-служби.	2
16	Приклади розробки RSS-джерел та RSS-рідерів.	2

17	Реалізація асинхронної взаємодії Web-браузера з Web- вузлом за допомогою технології AJAX	1
18	Створення мобільних Web-додатків за допомогою ASP.NET.	1
Усього годин		**26**

6. Самостійна робота

№ з/п	Назва теми	Кількість годин
		д/ф
1	Знайомство з HTML та CSS	4
2	Знайомство з текстовими елементами HTML	4
3	Знайомство з основними стильовими правилами CSS	5
4	Знайомство з основними правилами використання, позиціонування та оформлення зображень	5
5	Створення динамічного меню навігації за допомогою списків і правил CSS	5
6	Створення макета сторінки засобами правил CSS	5
7	Обробка подій за допомогою сценаріїв Javascript	5
8	Динамічний HTML	5
9	Створення форми з динамічною перевіркою	5
10	Знайомство з Silverlight	5
11	Знайомство з новими елементами HTML, та прикладами їх використання	5
	Разом	**53**

7. Методи навчання

Лекція, бесіда, презентація, лабораторна робота, самостійна робота.

8. Методи контролю

Для поточного контролю застосовуються тестові завдання.
Для підсумкового контролю – екзамен.

9. Розподіл балів, які отримують студенти для екзамену

Поточне тестування та самостійна робота									Підсумковий тест (екзамен)	Сума
Змістовий модуль 1	Змістовий модуль 2	Змістовий модуль 3	Змістовий модуль 4	Змістовий модуль 5	Змістовий модуль 6	Змістовий модуль 7	Змістовий модуль 8	Змістовий модуль 9	50	100

Т1	Т2	Т3	Т4	Т5	Т6	Т7	Т8	Т9	Т10	Т11	Т12	Т13	Т14	Т15	Т16	Т17	Т18		
2	2	3	3	2	2	3	3	3	3	3	3	3	3	3	3	3	3		

Т1, Т2 ... Т18 – теми змістових модулів.

Шкала оцінювання: національна та ЄКТС

Сума балів за всі види навчальної діяльності	Оцінка за національною шкалою	
	для екзамену, курсового проекту (роботи), практики	для заліку
90 – 100	відмінно	зараховано
65–89	добре	
50–64	задовільно	
1–49	незадовільно	не зараховано

10. Методичне забезпечення

1. Конспекти лекцій.
2. Комп'ютерні презентації.
3. Методичні рекомендації до виконання лабораторних робіт.
4. Методичні рекомендації до виконання самостійної роботи.
5. Критерії оцінювання знань студентів.
6. Питання до екзамену.

Лабораторна робота №1.

Тема «Службові утиліти для роботи в Інтернет. Вивчення протоколу HTTP».

Мета роботи:

1) Вивчення структури IP-адреси.
2) Ознайомлення з найбільш популярними утилітами для діагностики мережевої конфігурації і мережних з'єднань.
3) Ознайомлення з основами протоколу HTTP.

Теоретичний матеріал

1. Структура IP адреси.

IP-адреса складається з двох частин: номера мережі і номера вузла в мережі.

Найпоширенішим є запис IP-адреси у вигляді чотирьох чисел, розділених крапками, кожне з яких представляє значення байту в десятковій формі, наприклад: 213.180.204.11. Запис адреси не передбачає спеціального розмежувального знаку між номером мережі і номером вузла.

Для поділу цих частин зазвичай використовується 2 підходи:

1. За допомогою маски (RFC 950, RFC 1518), що представляє собою число в парі з IP-адресою. За допомогою операції «*логічне І*» над цими двома числами виділяється номер мережі.
2. За допомогою класів адрес (RFC 791).

Вводиться п'ять класів адрес: A, B, C, D, E.

A, B, C – використовуються для адресації мереж, *D* і *E* – мають спеціальне призначення. Ознакою, на підставі якого IP-адреса відносять до того чи іншого класу, є значення декількох перших бітів адреси.

Таблиця 1.1

Розподіл адрес в IP мережах.

Класс	Первые биты	Наименьший номер сети	Наибольший номер сети	Максимальное число узлов в сети
A	0	1.0.0.0 (0 - не используется)	126.0.0.0 (127 - зарезервирован)	2^{24} (3 байта)
B	10	128.0.0.0	191.255.0.0	2^{16} (2 байта)
C	110	192.0.0.0	223.255.255.0	2^{8} (1 байт)
D	1110	224.0.0.0	239.255.255.255	групповые адреса
E	11110	240.0.0.0	247.255.255.255	зарезервировано

В рамках IP протоколу існують обмеження при призначенні IP-адрес, а саме:

- номери мереж і номери вузлів не можуть складатися з довічних нулів або одиниць;

- якщо IP-адреса складається тільки з двійкових нулів, то вона називається невизначеною адресою і позначає адресу того вузла, який згенерував цей пакет;
- якщо в полі номера мережі стоять тільки нулі, то за замовчуванням вважається, що вузол призначення належить тій же самій мережі, що і вузол, який відправив пакет; така адреса може бути використана тільки як адреса відправника;
- якщо всі двійкові розряди IP-адреси дорівнюють 1, то пакет з такою адресою призначення повинен розсилатися всім вузлам, що знаходяться в тій же мережі, що й джерело цього пакету; таку адресу називається *обмеженим широкомовним*, оскільки пакет не зможе вийти за межі мережі;
- якщо в поле адреси призначення в розрядах, які відповідають номеру вузла, стоять тільки одиниці, то пакет розсилається всім вузлам мережі, номер якої зазначений в адресі призначення; такий тип адреси називається *широкомовним*;
- якщо перший октет адреси дорівнює 127, то така адреса називається внутрішньою адресою стека протоколів; вона використовується для тестування програм, організації клієнтської і серверної частин додатків, встановлених на одному комп'ютері;
- групові адреси, що відносяться до класу D, призначені для економічного розповсюдження в Інтернеті, великій корпоративній мережі аудіо- або відеопрограм.

Стандартним класам мереж можна поставити у відповідність наступні значення маски:

- клас A – 255.0.0.0;
- клас B – 255.255.0.0;
- клас C – 255.255.255.0

Розглянемо наступний приклад:

Вихідні дані	*IP адреса*	62.76.167.21
	Маска мережі	255.255.255.0
Логічна операція	I	
Результат	*Адрес мережі*	62.76.167.0
	Номер комп'ютера	21

Для визначення мережевих налаштувань комп'ютеру і мережевого устаткування, діагностики та отримання іншої інформації, що відноситься до інтернет-протоколів, широко використовуються спеціальні утиліти.

2. Утиліта ipconfig.

Ipconfig – це утиліта командного рядка для виводу деталей поточного з'єднання комп'ютеру з мережею і контролю над клієнтським сервісом DHCP. DHCP (Dynamic Host Configuration Protocol) – це мережевий протокол, що

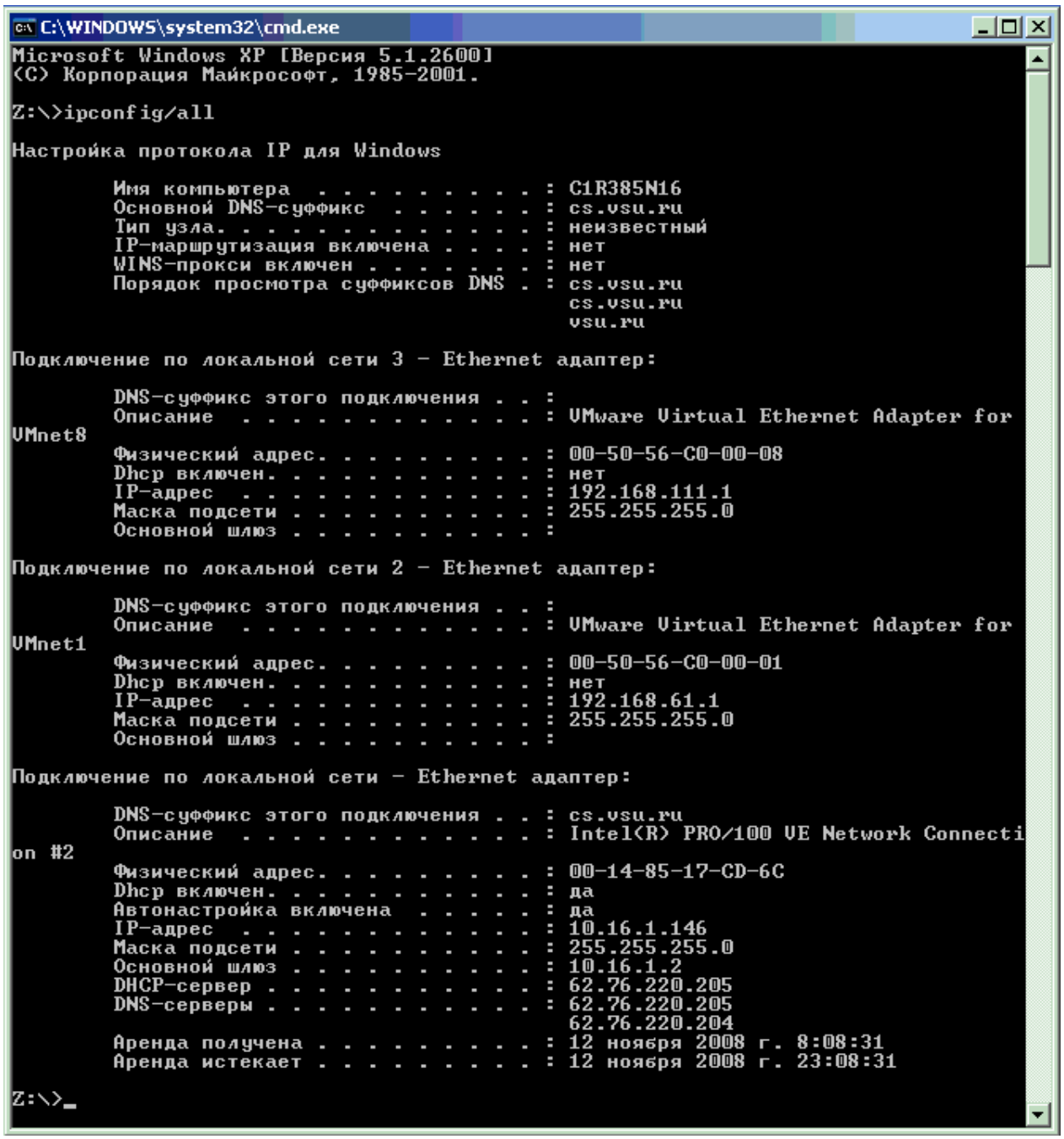
дозволяє комп'ютерам автоматично отримувати IP-адресу та інші параметри, необхідні для роботи в мережі TCP / IP.

Синтаксис команди:

ipconfig/*ключі*

Команда *ipconfig/all* – відображає повну інформацію по всіх мережевих адаптерів.

Приклад виводу для Windows:



3. Утиліта *ping*.

Ping (Packet InterNet Grouper) – це системна програма, призначена для перевірки з'єднань в мережах на основі TCP/IP. Вона відправляє *Echo-Request* запити протоколу ICMP зазначеному вузлу мережі й фіксує відповіді які

надходять (ICMP Echo-Reply). Час між відправленням запиту й одержанням відповіді (RTT, Round Trip Time) дозволяє визначати двосторонні затримки (RTT) за маршрутом і частоту втрати пакетів. Що дозволяє побічно визначати завантаженість каналів передачі даних і проміжних пристроїв. Повна відсутність ICMP-відповідей може також означати, що віддалений вузол (або який-небудь з проміжних маршрутизаторів) блокує ICMP Echo-Reply або ігнорує ICMP Echo-Request.

Синтаксис:

ping - *параметри кінцеве_ім'я*

Кінцеве ім'я – це доменне ім'я або IP-адреса хосту.

Приклад:

```
C:\>ping www.mail.ru

Обмен пакетами с www.mail.ru [194.67.57.26] по 32 байт:

Ответ от 194.67.57.26: число байт=32 время=11мс TTL=118
Ответ от 194.67.57.26: число байт=32 время=11мс TTL=118
Ответ от 194.67.57.26: число байт=32 время=11мс TTL=118
Ответ от 194.67.57.26: число байт=32 время=11мс TTL=118

Статистика Ping для 194.67.57.26:
    Пакетов: отправлено = 4, получено = 4, потеряно = 0 (0% потерь),
Приблизительное время приема-передачи в мс:
    Минимальное = 11мсек, Максимальное = 11 мсек, Среднее = 11 мсек
```

4. Утиліта *traceroute.*

Traceroute (скорочено *tracert*) – це службова програма, призначена для визначення маршрутів прямування пакетів в мережах *TCP/IP*. Робота *traceroute* заснована на протоколі *ICMP*.

Traceroute виконує відправку пакетів вказаною вузла мережі, відображаючи при цьому відомості про всі проміжні маршрутизатори, через які пройшли пакети на шляху до цільового вузла. У разі проблем при доставці пакетів до якого-небудь вузла програма *traceroute* дозволяє визначити, на якій саме ділянці мережі виникли неполадки.

Синтаксис:

tracert - *параметри кінцеве_ім'я*

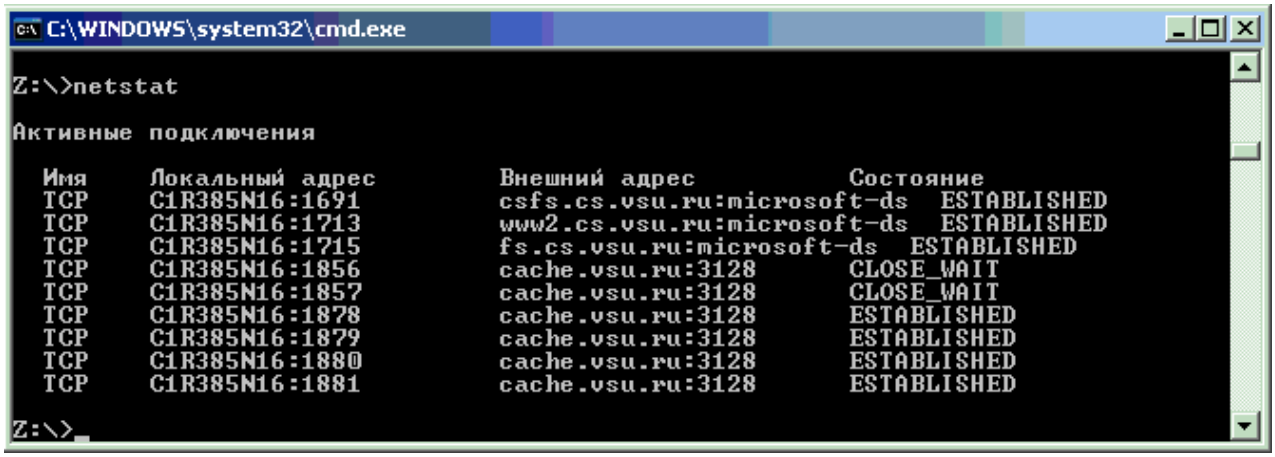
Кінцеве ім'я – це доменне ім'я або IP-адреса хосту.

Приклад:

```
Z:\>tracert www.mail.ru

Трассировка маршрута к www.mail.ru [194.67.57.26]
с максимальным числом прыжков 30:

  1    <1 мс    <1 мс    <1 мс  c1swr1.cs.vsu.ru [10.16.1.2]
  2    <1 мс    <1 мс    <1 мс  c2swr1-ei1000-4-1-ve100.vsu.ru [62.76.168.42]
  3    <1 мс    <1 мс    <1 мс  c2r1-ve5.vsu.ru [62.76.168.50]
  4    27 ms    22 ms    27 ms  RBNet-VSU.vsu.ru [62.76.168.1]
  5    28 ms     *       30 ms  cisco13.Moscow.gldn.net [193.232.244.43]
  6     *        *        *     Превышен интервал ожидания для запроса.
  7    36 ms    45 ms    29 ms  cat07.Moscow.gldn.net [194.186.157.82]
  8    24 ms    46 ms    29 ms  cat01.Moscow.gldn.net [194.186.158.110]
  9    44 ms    27 ms    23 ms  mailru-KK12-1-gw.Moscow.gldn.net [195.239.8.10]

 10    36 ms    47 ms    49 ms  mail.ru [194.67.57.26]

Трассировка завершена.

Z:\>_
```

5. Утиліта *netstat.*

Netstat – службова програма, що відображає статистику протоколу і поточних мережевих підключень TCP/IP:



6. Утиліта *telnet.*

Telnet – мережевий протокол для реалізації текстового інтерфейсу по мережі. Назва «*telnet*» має також утиліта, що реалізує клієнтську частину протоколу. Історично *telnet* служив для віддаленого доступу до інтерфейсу командного рядка операційних систем. Протокол *telnet* може використовуватися для виконання налагодження інших протоколів на основі транспорту TCP.

Утиліта telnet підтримує наступні команди:

- *Close* – закриття поточного з'єднання.
- *Display* – відображення параметрів операції.
- *Open* – підключення до сайту.
- *Quit* – вихід з telnet.
- *Set* – встановлення параметрів.
- *Send* – відправлення рядка на сервер.
- *Status* – виведення відомостей про поточний стан.
- *Unset* - скидання параметрів.

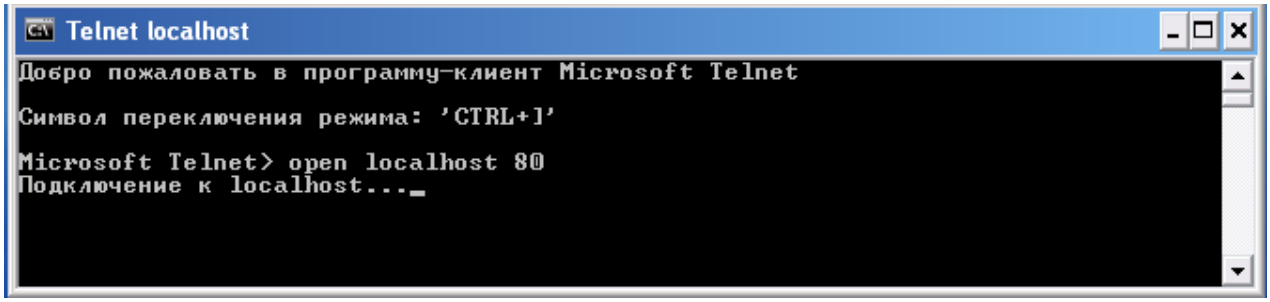
Використовуючи утиліту *telnet* можна, наприклад, вручну відправити запит клієнта і отримати відповідь сервера за протоколом HTTP.

Для цього виконаємо наступну послідовність дій:

1. Запуск утиліти telnet.
2. Встановлення зєднання з веб-сервером за допомогою команди: open *ім'я_хоста 80.*
3. Формування запиту клієнта.
4. Отримання відповіді сервера.

Приклад:

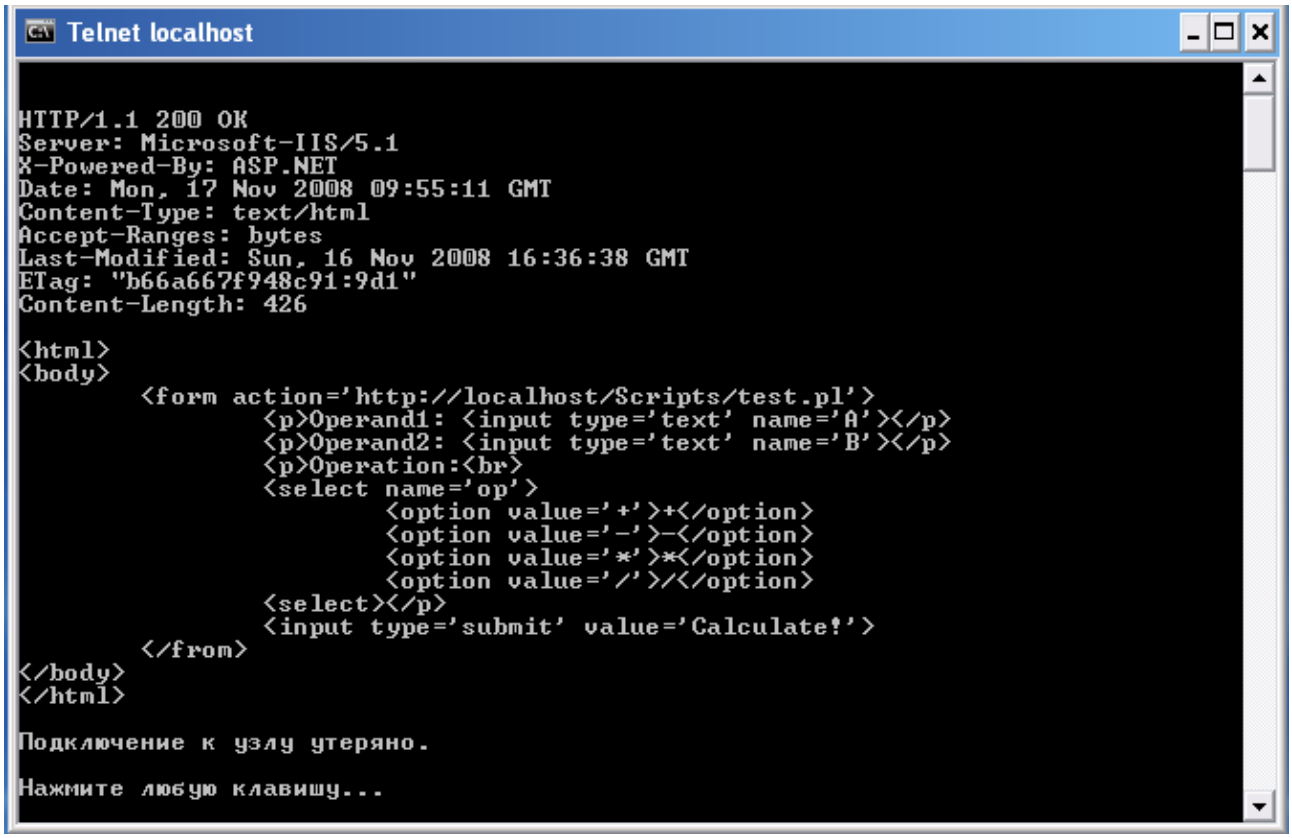
1. Встановлюємо з'єднання: *open localhost 80*



2. Формуємо рядок стану запиту клієнта:

GET HTTP: //LOCALHOST/PERLCALC.HTML HTTP/1.0 <ENTER> <ENTER>

3. Отримуємо відповідь сервера:



Видно, що відповідь веб-сервера *localhost* містить рядок стану (з кодом успішного завершення 200), поля заголовка (*Server*, *Date*, *Content-type* і ін.) і тіло, що містить HTML код запитаного клієнтом документа *http://localhost/perlcalc.html*.

Порядок виконання роботи

Завдання 1.

Переглянути параметри:

1. За допомогою утиліти ***ipconfig*** (запускається в командному рядку командою *ipconfig*) визначте IP-адресу і маску підмережі для свого комп'ютеру.
2. Визначте клас підмережі, в якій знаходиться ваш комп'ютер без використання маски підмережі і по масці підмережі.
3. Визначте адресу підмережі, в якій знаходиться ваш комп'ютер, з використанням функції «Логічне І» над IP-адресою і маскою підмережі. Слід мати на увазі, що операція «Логічне І» повинна проводитися з двійковим поданням операндів.

Завдання 2.

За допомогою утиліти *ping* (запускається в командному рядку командою *ping*) перевірте доступність хостів, мінімальний, середній і максимальний час прийому-передачі ICMP пакетів до них. Можна розглянути хости, наприклад в наступній послідовності:

1. Сервер вашого безпосереднього провайдера або сервера вашої підмережі.
2. Який-небудь сервер вашого регіону.
3. Веб-сервер Інтернет-Університету Інформаційних Технологій: *www.intuit.ru*.
4. Веб-сервер Університету в Кембриджі: *www.cam.ac.uk*.
5. Веб-сервер Університету в Каліфорнії: *www.ucla.edu*.
6. Веб-сервер Університету в Токіо: *www.u-tokio.ac.jp*.
7. Веб-сервер компанії Microsoft: *www.microsoft.com*.

Зверніть увагу, що в останньому випадку ICMP-пакети блокуються веб-сервером.

Завдання 3.

За допомогою утиліти *tracert* (запускається в командному рядку командою *tracert*) визначте маршрути проходження і час проходження пакетів до хостів, наведених у завданні 2.

Завдання 4.

1. За допомогою утиліти *netstat* (запускається в командному рядку командою *netstat*) подивіться активні поточні з'єднання і їх стан на вашому комп'ютері.
2. Запустіть кілька екземплярів веб-браузера, завантаживши в них веб-сторінки з різних веб-серверів. Подивіться з допомогою netstat, які нові мережеві підключення з'явилися в списку.

3. Закривайте браузери і за допомогою *netstat* перевіряйте зміну списку мережевих підключень.

Завдання 5.
Ознайомлення з протоколом HTTP за допомогою утиліти *telnet*.

1. Запустіть сеанс *telnet* (запускається в командному рядку командою *telnet*). При цьому з'явиться підказка *Microsoft Telnet*>. З повним списком команд можна ознайомитися за допомогою команди *help*.
2. Дозвольте режим відображення введених з клавіатури символів за допомогою команди *set localecho*.
3. Відповідно до протоколу HTTP необхідно встановити з'єднання з веб-сервером. Для цього за допомогою команди *open* встановлюється з'єднання, наприклад: *open www.yandex.ru 80*.
4. Сформуйте клієнтський запит. Як мінімум він повинен містити рядок стану, наприклад:

GET HTTP://WWW.YANDEX.RU/INDEX.HTML HTTP/1.0

Якщо поля запиту відсутні, то введення закінчується двома натисканнями клавіші <ENTER> для вставки порожнього рядка після заголовка.

Слід звернути увагу на те, що при введенні не можна допускати помилок, оскільки при спробі їх виправити за допомогою клавіші <BACKSPACE>, її натискання інтерпретується як частина запиту.

5. Вивчіть отриману відповідь сервера. Зверніть увагу на код відповіді в рядку стану відповіді веб-сервера в рядку стану і поля заголовка відповіді.

Якщо відповідь сервера дуже велика (в першу чергу через розмір документа в тілі відповіді), то зміст відповіді сервера у вікні інтерпретатора командного рядка обрізається з початку. У цьому випадку рекомендується для перегляду заголовка замість методу *GET* використовувати метод *HEAD*.

Контрольні завдання

1 За допомогою якої утиліти за заданим доменному імені хосту можна визначити його IP адресу? Визначте IP адресу *хоста www.mail.ru*.

2 За допомогою утиліти *telnet* визначте який веб-сервер встановлений на хості *www.rbc.ru*.

3 Визначте маршрут проходження ICMP пакетів до хосту *www.ttt.com*. Визначте приблизну географічну локалізацію хосту.

Лабораторна робота №2.

Тема «Принципи веб-дизайну. Знайомство з Microsoft Expression Web.»

Мета роботи:

1. Ознайомлення з основними поняттями та принципами веб-дизайну.
2. Знайомство з основними можливостями з розробки веб-сторінок і веб-сайтів за допомогою **Microsoft Expression Web 2**.

Теоретичний матеріал

Веб-дизайн (*web page design*) – це процес проектування, планування, моделювання та реалізації доставки електронного змісту через мережу Інтернет з використанням технологій (на основі мов розмітки), придатних для інтерпретації та візуалізації веб-браузером або іншим графічним веб-інтерфейсом користувача.

Дуже важливим є приведення веб-ресурсу у відповідність стандартам W3C, що забезпечує доступність змісту для людей з обмеженими фізичними можливостями та користувачів портативних пристроїв, а також кросплатформеність коду розмітки ресурсу.

Основними аспектами веб-дизайну (в першу чергу для комерційних веб-сайтів) є наступні:

- Зміст. Інформаційне наповнення має бути релевантним тематиці сайту і орієнтованим на цільову аудиторію сайту.
- Зручність у використанні (*usability*). Сайт повинен мати дружній користувальницький інтерфейс з простою і надійною навігацією.
- Зовнішній вигляд. Графіка і текст повинні бути виконані в одному стилі і узгоджені на всіх сторінках сайту. Стиль оформлення повинен демонструвати професіоналізм, привабливість і релевантність.
- Видимість. Сайт повинен легко знаходитись за допомогою більшості пошукових систем і рекламних майданчиків.

По-суті, веб-сайт – це інформаційна система, яка містить два основних компоненти:

- Компоненту подання (*front-end*). Видимий зміст (розмітка сторінок, графіка, аудіо та текст).
- Компоненту реалізації (*back-end*). Повязана з організацією та ефективною реалізацією вихідних кодів. Включає в себе не відображаємі сценарії, серверні компоненти, що є основою для компонентів подання.

Складанням *технічного завдання* на розробку веб-сайту для фахівців займається *менеджер проекту*. Робота з *замовником* починається зі складання *короткого опису*, в якому викладаються побажання замовника по *візуальному поданні* і структурі сайту. З урахуванням можливостей програмних і дизайнерських засобів і виходячи з короткого опису, *менеджер* складає *технічне завдання*, яке має бути *затверджене замовником*.

Етапи проектування веб-сайту залежать від обсягу сайту, його функціональності і багато чого іншого.

Розробка веб-сайту включає в себе наступні етапи:

- Дизайн головною і типових сторінок сайту. (Виконується зазвичай в графічному редакторі).
- HTML-кодування, в результаті якого створюється код, який можна переглядати за допомогою браузера.
- Програмування сайту. Може здійснюватися як «з чистого аркуша», так і за допомогою спеціального високорівнего пакету – системи управління сайтом (CMS).
- Розміщення сайту в Веб, наповнення контентом і публікація.
- Оптимізація веб-сайту з метою підвищення його видимості в Веб.
- Здача сайту замовнику.

Для дизайну веб-сайту і розробки веб-додатків для нього має широкий спектр інструментальних засобів. Компанія Майкрософт має вже давню традицію створення WYSIWYG-інструментів такого роду. Серед таких інструментів можна назвати, зокрема *Microsoft Visual Studio. NET* і *Microsoft Expression Web*.

Microsoft Expression Web.

Microsoft Expression Web є складовою частиною пакету Microsoft Expression Studio.

Microsoft Expression Studio – пакет графічних програм компанії Майкрософт, призначених для розробників і дизайнерів. Використовує програмний інтерфейс *WinFX*.

WinFX (тепер *NET Framework 3.0*) – прикладний програмний інтерфейс, що входить до складу операційних систем *Windows Vista* і *Windows Server 2008*.

До складу *Microsoft Expression Studio* входять наступні інструменти:

- *Microsoft Expression Blend* – WYSIWYG-інструмент для проектування користувальницького інтерфейсу додатків, створюваних на основі Windows Presentation Foundation.
- *Microsoft Expression Web* – WYSIWYG-інструмент для дизайну веб-сайтів і редактор HTML.
- *Microsoft Expression Media* – векторно/растровий графічний редактор.
- *Microsoft Expression Design* – професійний інструмент для графічного дизайну та підготовки ілюстрацій для настільних і web-додатків.
- *Microsoft Expression Encoder* – професійний мультимедіа-кодек для вмісту у форматі *VC-1* для *Microsoft* Silverlight.

Microsoft Expression **Web 2** дозволяє розробляти веб-сторінки, що інтегрують в рамках веб-сайту такі технології як *XML, CSS 2.1, ASP.NET 2.0, XHTML, XSLT* і *JavaScript.*, А також впроваджувати в сторінки керуючі елементи ASP.NET, Microsoft Silverligth і AJAX. Для роботи з цим інструментом потрібна наявність встановленої компоненти *.NET Framework 2.0.*

На сторінці http://www.microsoft.com/expression/products/overview.aspx?key=web є посилання на навчальне керівництво, яке демонструє основи роботи з цим середовищем розробки. Зокрема там демонструється створення порожнього веб-

сайту, майстер-сторінки, стильового оформлення, створення *.asp сторінки. Також показується яким чином можна додати підтримку RSS на сайті і як створити сторінку, що містить елемент управління Silverlight для перегляду відео. Одна із вправ присвячена впровадженню елементу управління .NET для відображення даних з джерела на основі XML. Показано також, як вставити AJAX елемент на сторінку для асинхронного виклику функції та оновлення індикатора часу.

Необхідно приділити особливу увагу створенню *майстер-сторінок*.

Оскільки сторінки веб-додатки становлять якесь єдине ціле, об'єднане єдиними вимогами до оформлення, змісту, розташуванню елементів управління, то створювані веб-сторінки сьогодні дуже рідко розробляються «з нуля». Зазвичай при створенні сторінок використовуються готові шаблони, або вони створюються в процесі розробки веб-сторінок додатку. Одним із засобів вирішення подібних завдань є *майстер-сторінки* (master pages). Вони реалізують просту модель створення шаблонів форм з можливістю їх повторного використання.

Для реалізації даного механізму існують такі типи сторінок, як майстер-сторінки (master pages) і *сторінки змісту* (content pages).

Майстер-сторінка являє собою шаблон сторінки, який може містити будь-які елементи, допустимі для звичайної сторінки, а також програмний код. Зазвичай майстер-сторінка містить фіксовані елементи, однакові для всіх сторінок, і заповнювач вмісту для іншої частини сторінки. Найбільш типовими фіксованими елементами є верхній і нижній колонтитули, панель навігації, панель меню та інші.

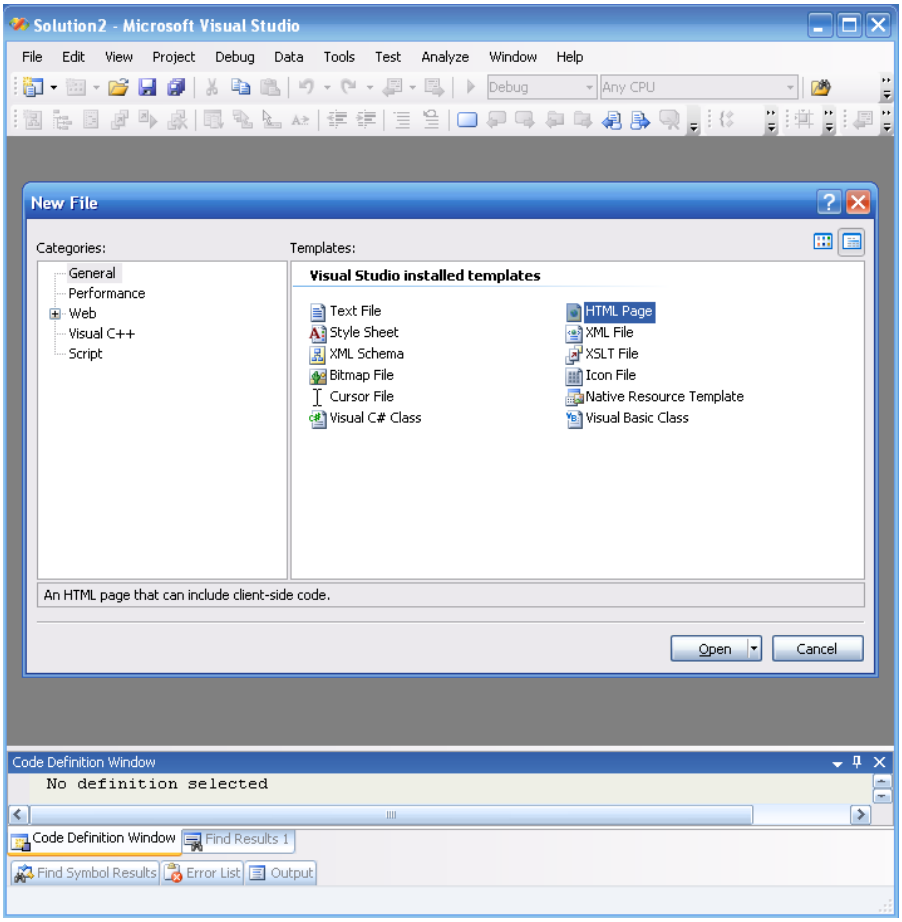
Сторінка змісту включає в себе допустимі елементи управління і за допомогою їх визначає зміст, яким заповнюються спеціальні області майстер-сторінок. Кожна *сторінка змісту* посилається лише на одну майстер-сторінку з якої вона отримує елементи. Крім фіксованих елементів майстер-сторінки вона також надає і додатковий зміст.

Майстер-сторінка повинна містити елемент управління *ContentPlaceHolder*, який призначений для визначення області, куди сторінка змісту може вставляти зміст. При створенні нової майстер-сторінки елемент *ContentPlaceHolder* створюється за замовчуванням. Для того щоб перетворити звичайну сторінку в сторінку змісту, необхідно як значення властивості *MasterPageFile* сторінки вказати ім'я майстер-сторінки, а також додати на сторінку потрібний елемент керування.

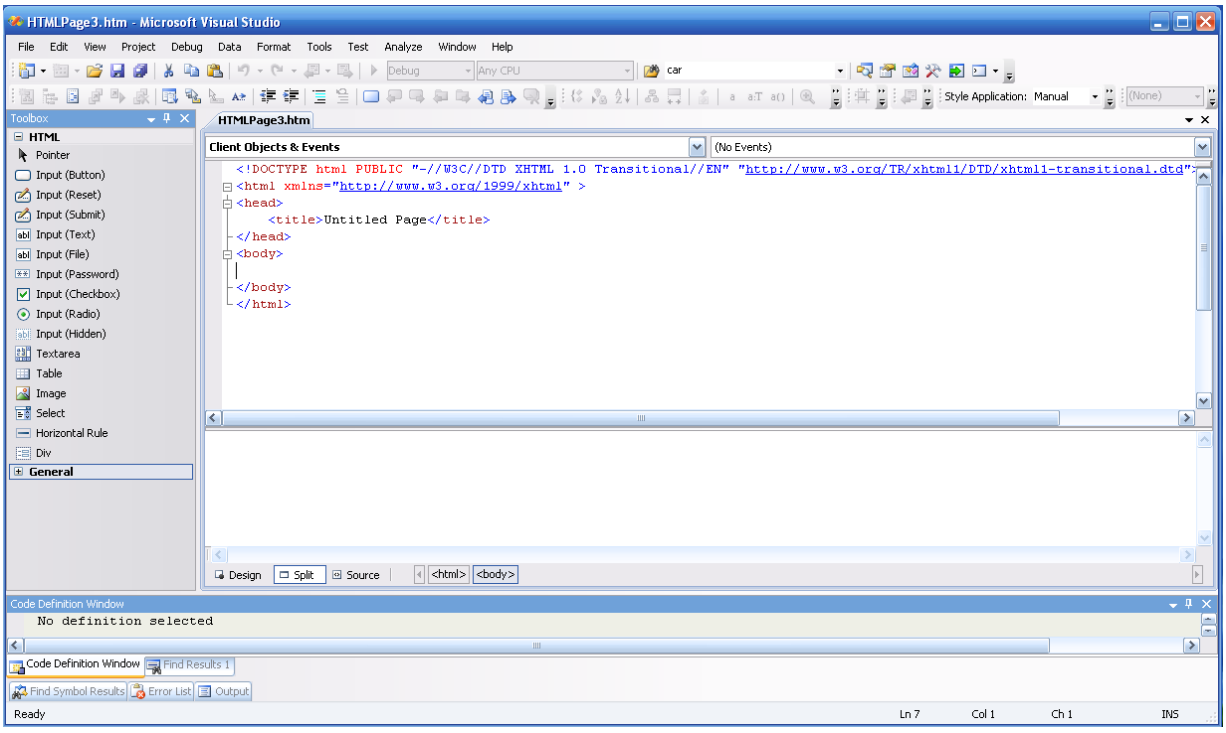
Порядок виконання роботи

1. Створення HTML сторінки в Microsoft Visual Studio. NET.

Для дизайну веб-сторінок в Visual Studio. NET необхідно вибрати розділ меню *File* => *New* і у вікні вибрати тип файлу *HTML Page*:

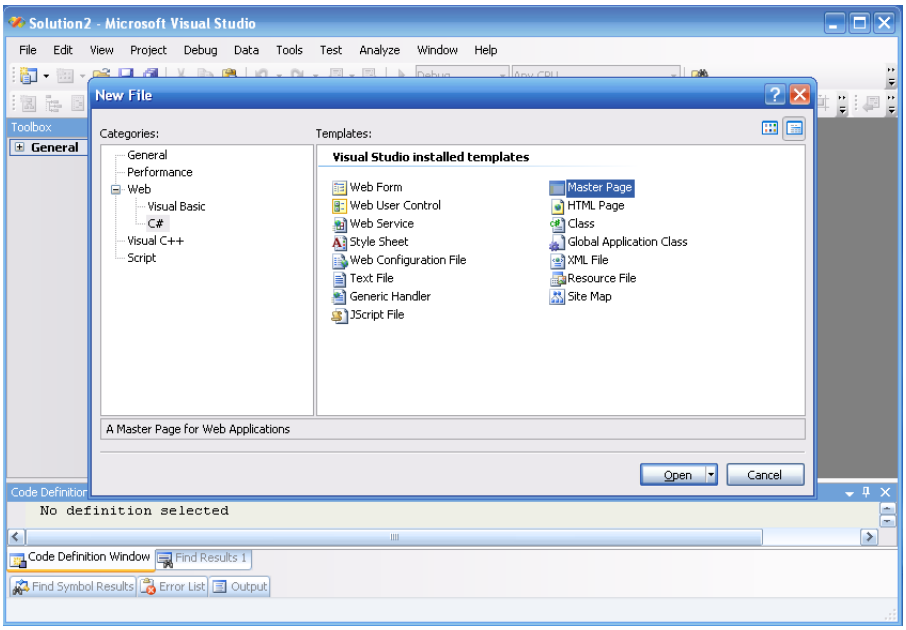


Для дизайну є панель Toolbox, що містить набір основних елементів html сторінки (підключається через меню: *View*> *Toolbox*):

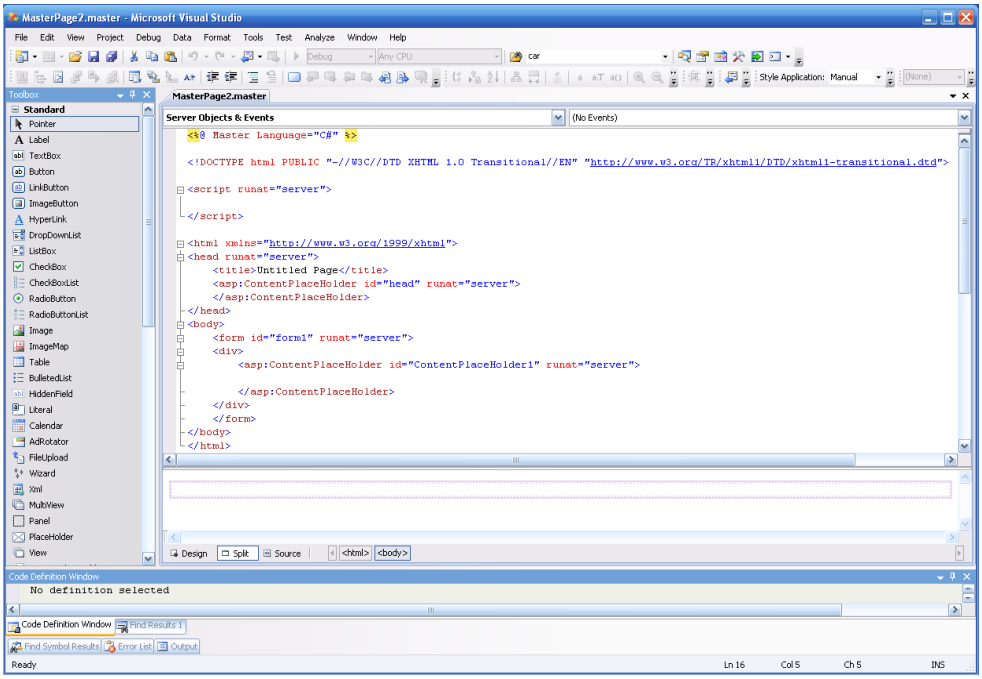


2. Створення майстер-сторінки в *Microsoft Visual Studio. NET*.

Для створення майстер-сторінки необхідно вибрати розділ меню *File* => *New* і у вікні вибрати тип файлу *Web*> *C#*> *Master Page*:



Після цього зявиться вікно майстер-сторінки, що містить порожню форму і два контейнери змісту *ContentPlaceHolder* (в заголовку і в тілі документа):

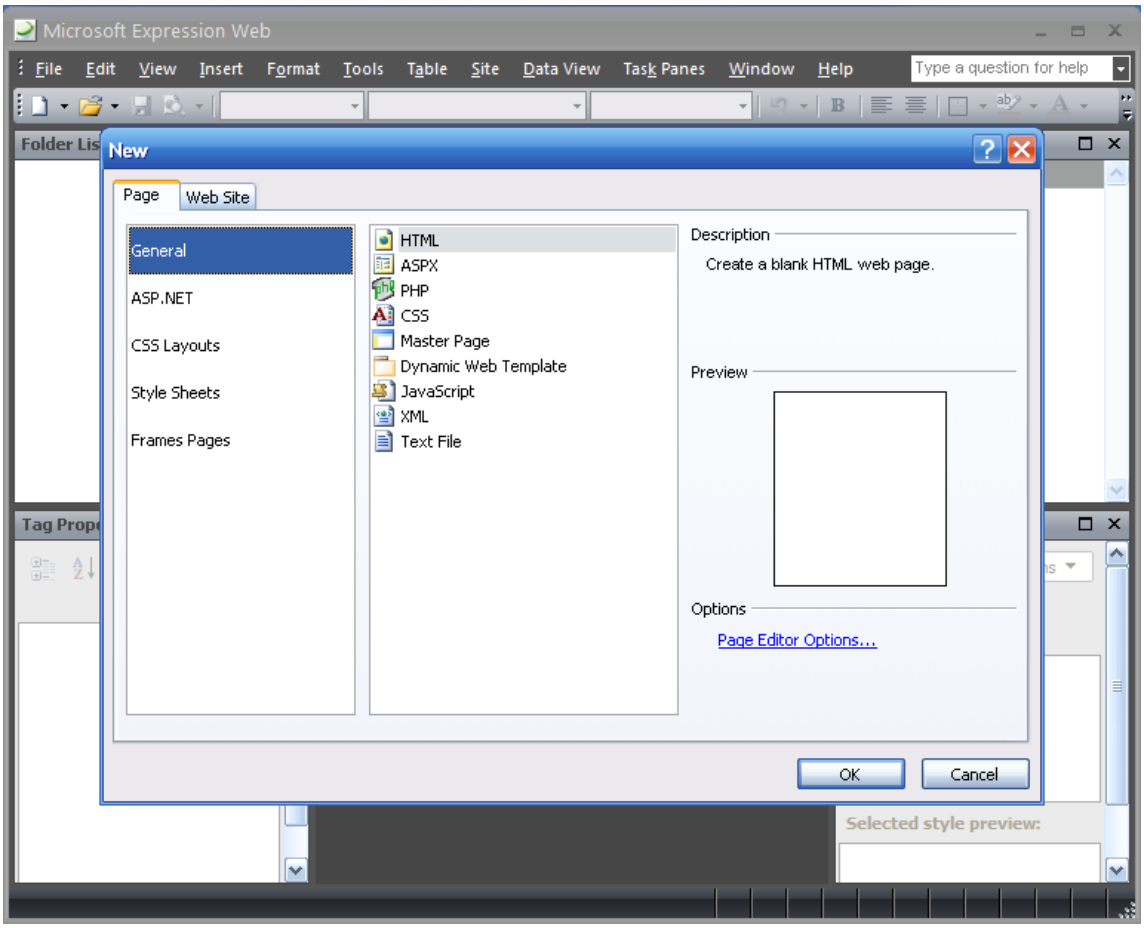


3. Знайомство з *Microsoft Expression Web*

1) Створення нового документа.

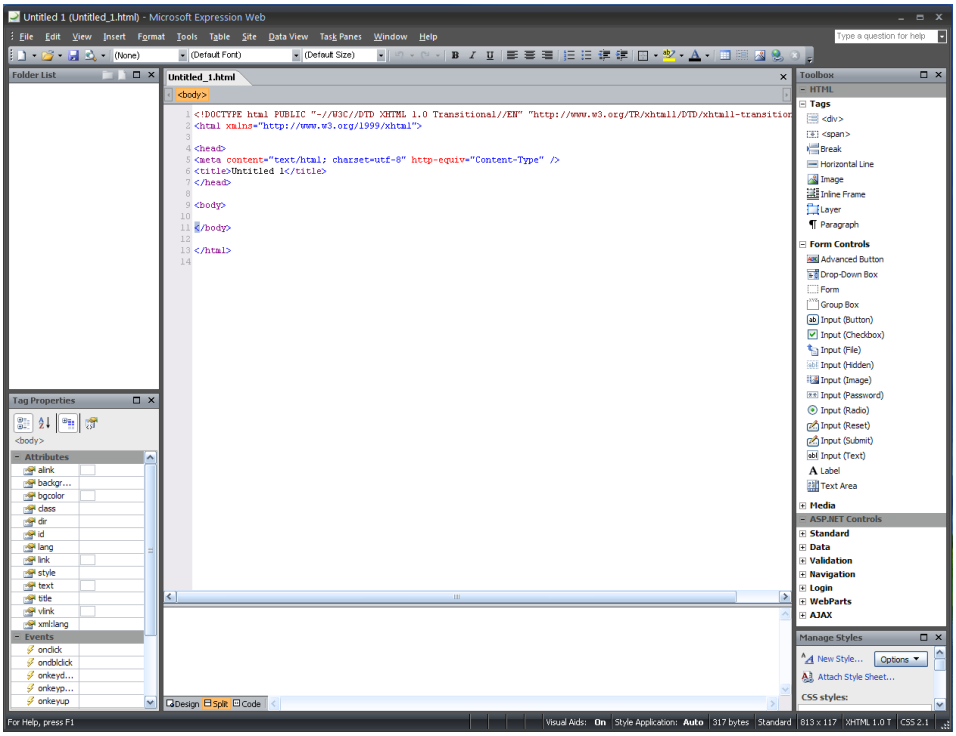
Microsoft Expression Web можна використовувати в якості візуального редактора документів різноманітних форматів: HTML, ASP.NET, PHP, CSS, JavaScript, XML. Підтримується робота з шаблонами, майстер-сторінками та веб-сайтами в цілому.

Вибравши розділ меню *File*> *New*, викличте наступне діалогове вікно:

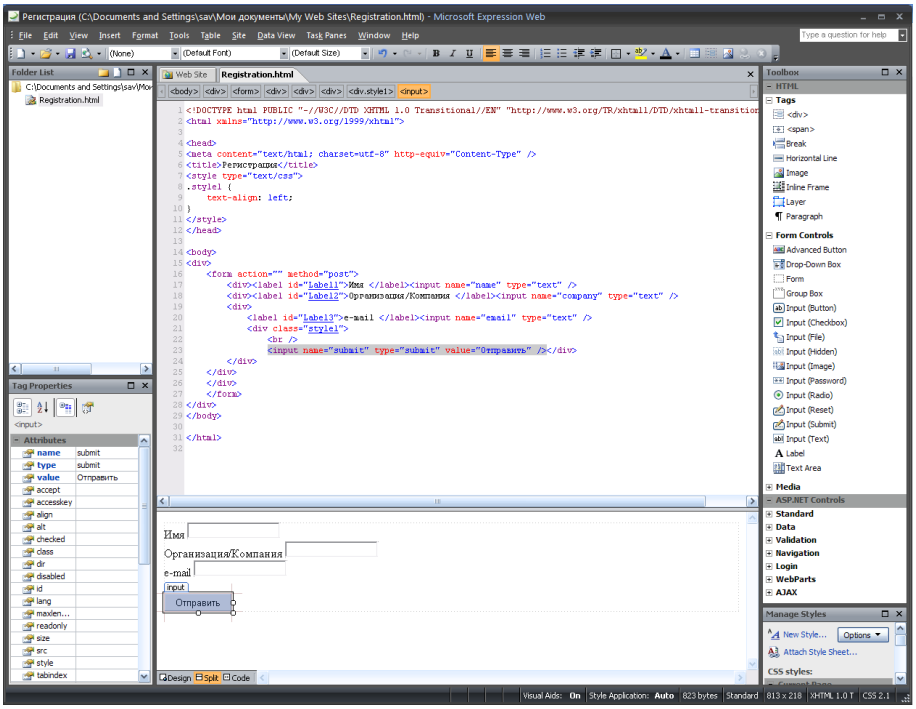


Виберіть формат документа HTML.

В результаті з'явиться наступне вікно, що містить порожню HTML сторінку:

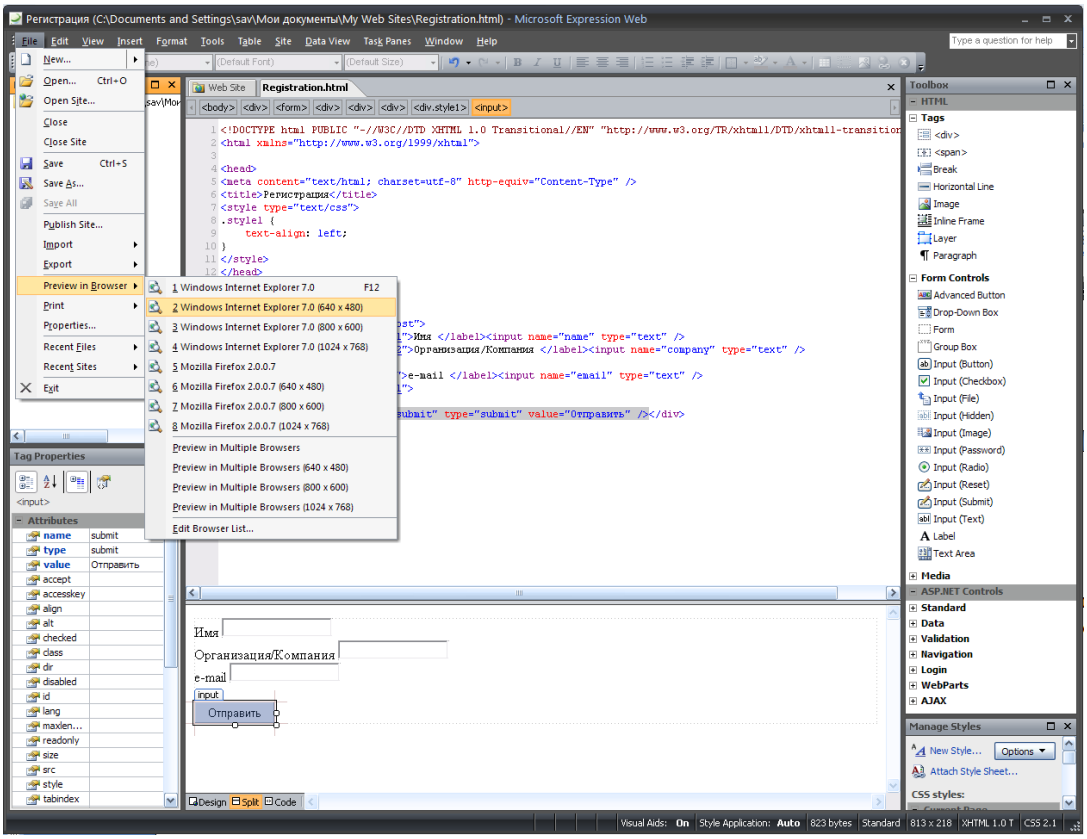


Використовуючи панель інструментів (Toolbox), створіть веб-сторінку для реєстрації користувача наступного виду:



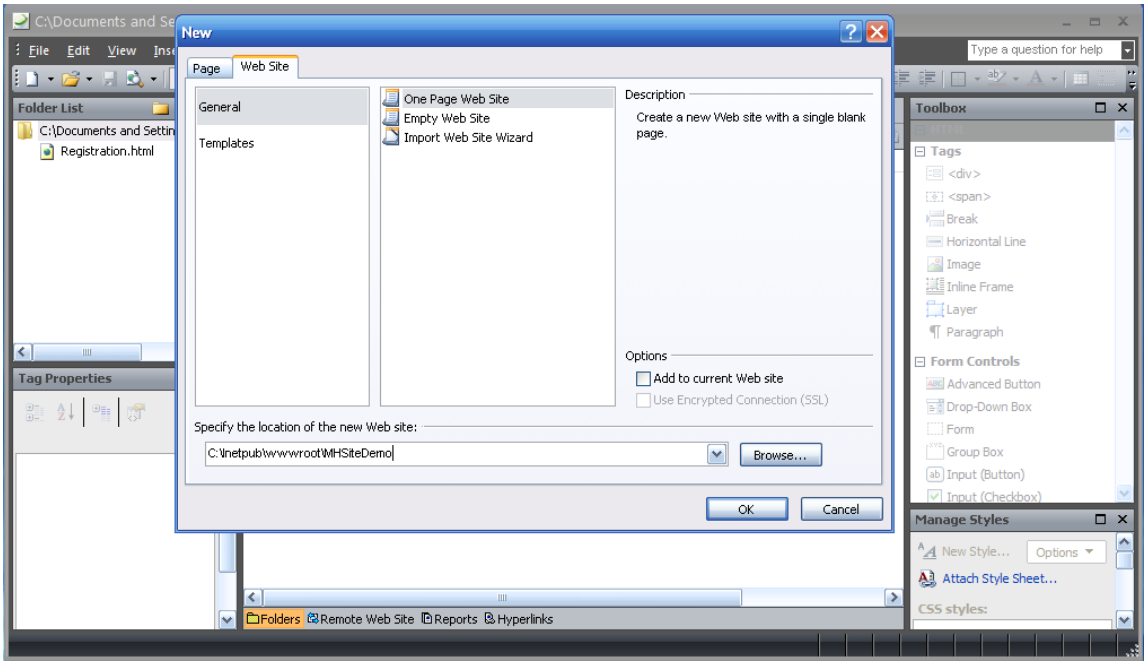
Для зручності вставки елементів форми рекомендується використовувати в якості контейнера елемент * *.

Після збереження документа (*File> Save As*) можна попередньо подивитися сторінку в одному з веб-браузерів:

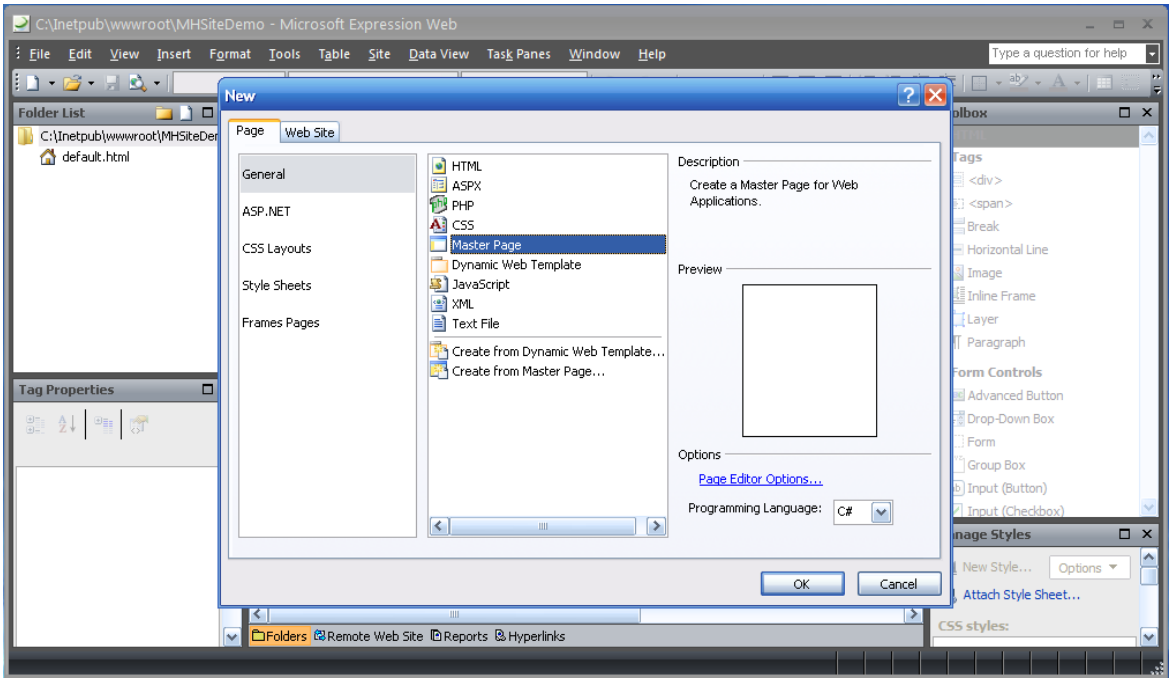


2) Створення веб-сайту.

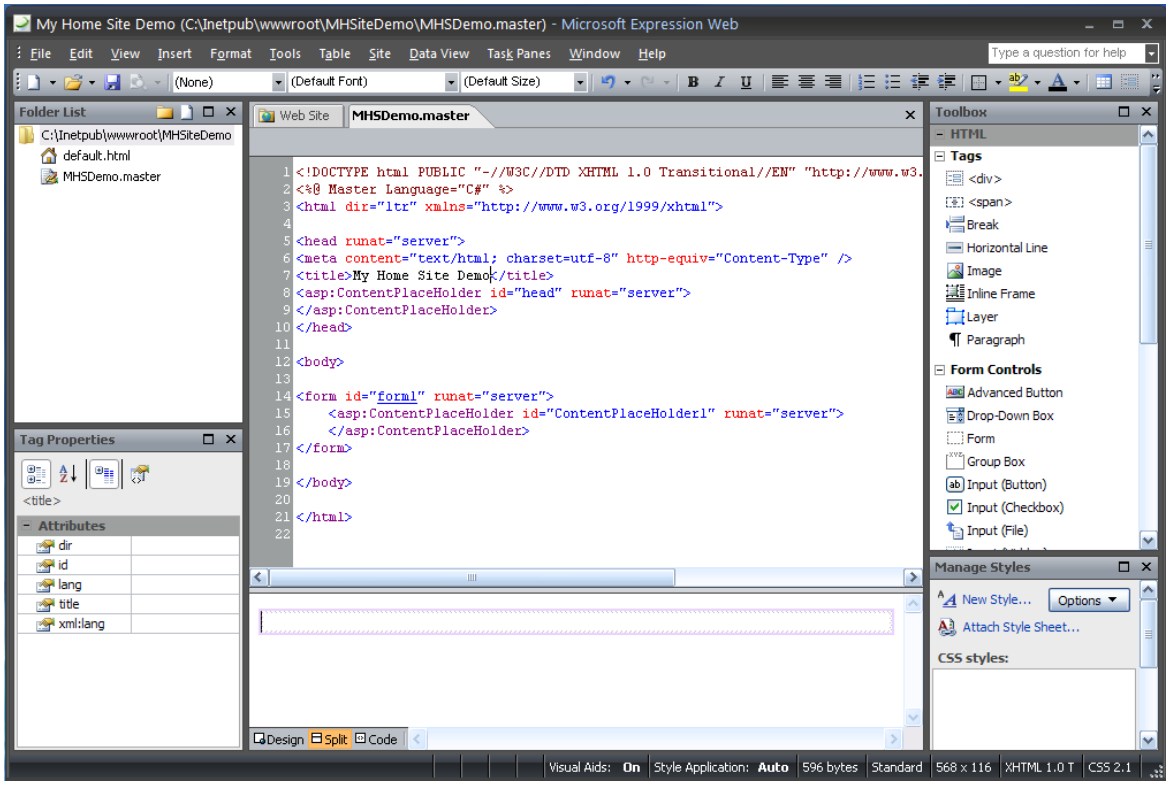
Для створення пустого веб-сайту виберіть розділ меню *File*> *New*, потім на закладці *Web Site* виберіть *Empty Web Site* і вкажіть розташування сайту:



Як шаблон для створення сторінок веб-сайту можна використовувати майстер-сторінку:



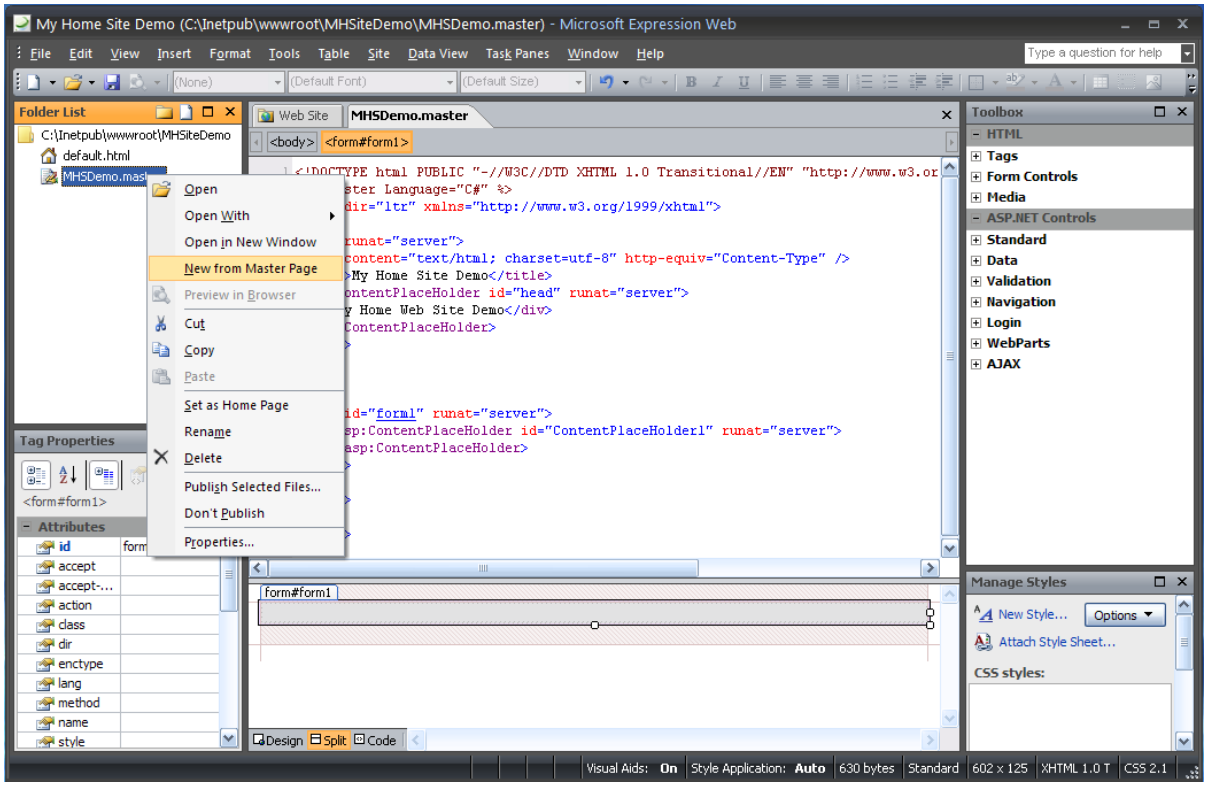
Створена майстер-сторінка містить два контейнери змісту *ContentPlaceHolder*: в заголовку і в тілі документа:



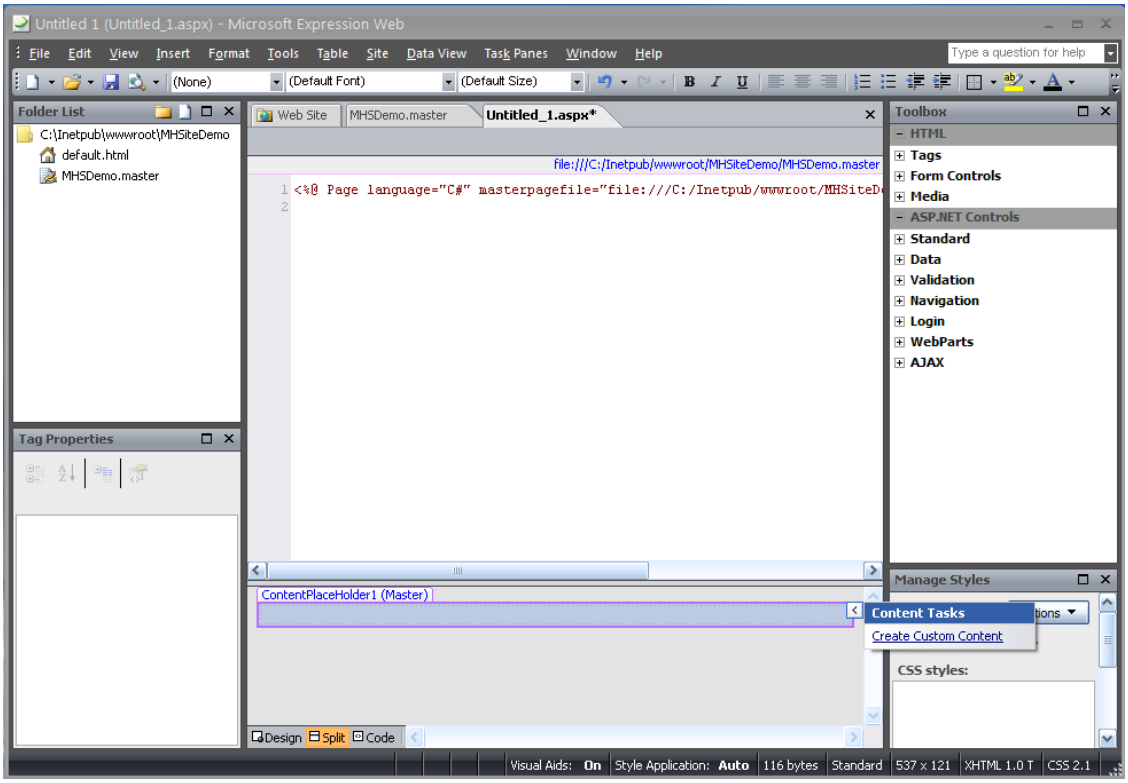
Всі зміни, внесені в цю *майстер-сторінку* будуть автоматично репліцируватися у всіх документах веб-сайту, створених на її основі. Вся специфіка новостворюваних документів повинна бути локалізована в межах контейнера змісту (*ContentPlaceHolder*). Наприклад, у заголовку створимо загальний заголовок «*My Home Web Site Demo*» для всіх сторінок веб-сайту:

```
<head runat="server">
<meta content="text/html; charset=utf-8" http-equiv="Content-Type" />
<title>My Home Site Demo</title>
        <asp:ContentPlaceHolder id="head" runat="server">
                <div>My Home Web Site Demo</div>
        </asp:ContentPlaceHolder>
</head>
```

Для створення нової веб-сторінки на основі майстер-сторінки необхідно вибрати в панелі списку файлів майстер-сторінку і потім в контекстному меню (викликається кліком правої кнопки миші) вибрати розділ «*New from Master Page*»:

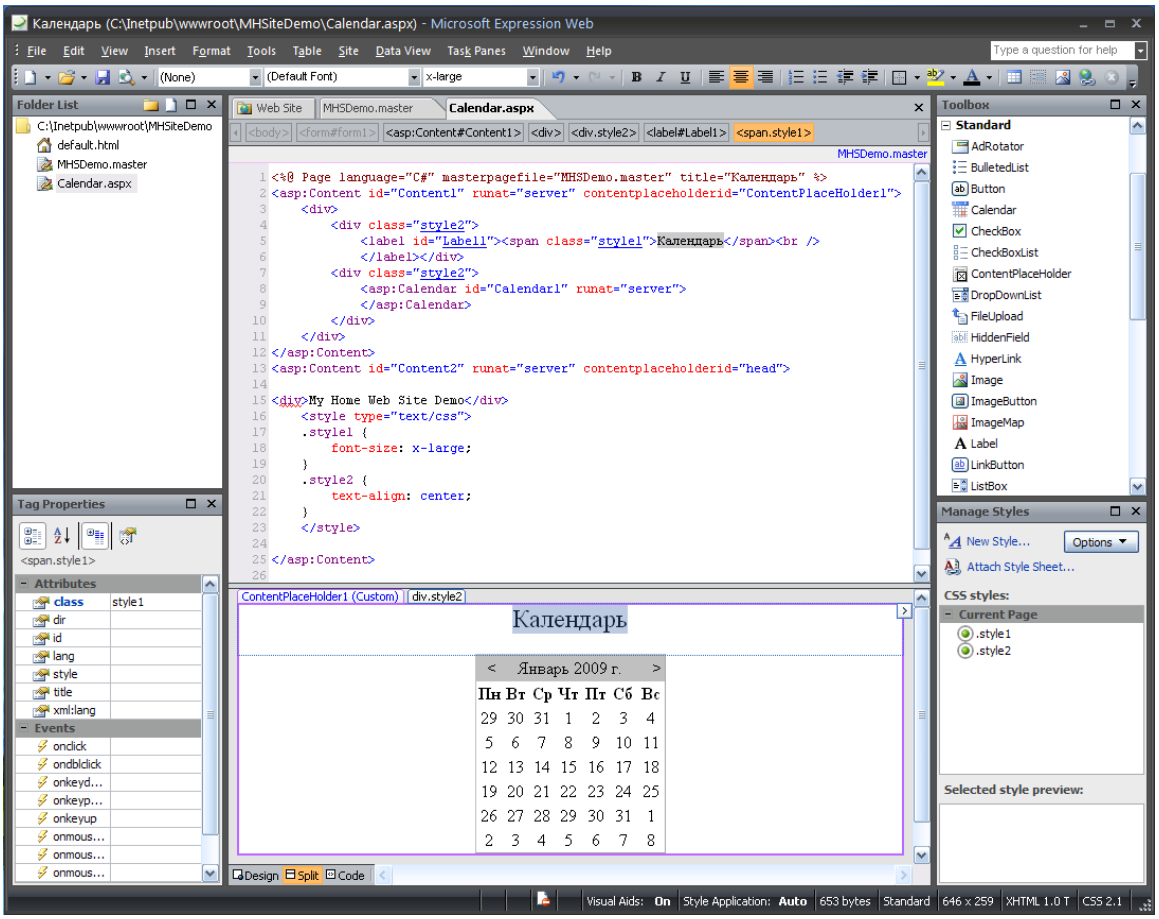


Далі, в режимі відображення «Split» для елемента *«ContentPlaceHolder1»* по стрілці праворуч від нього слід вибрати з контексного меню розділ *«Create Custom Content»:*

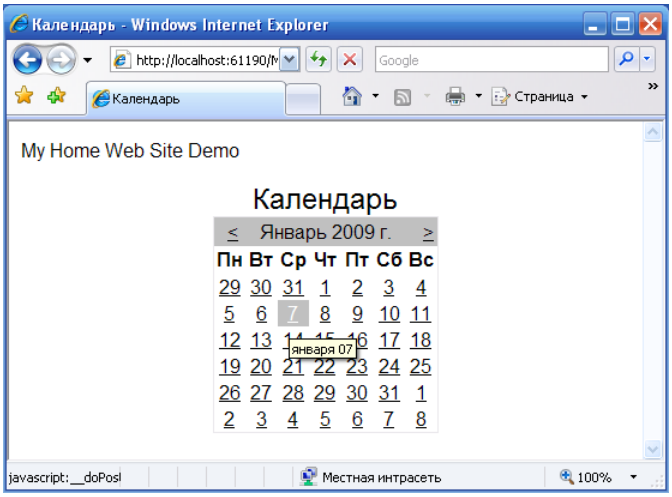


В результаті в коді документа з'являться пара тегів </asp: Content> </ asp: Content>, між якими може бути додано нове зміст сторінки.

Як приклад розглянемо наступний зміст:



Після збереження документа його можна переглянути у веб-браузері:



Контрольні завдання

1. Вставте логотип (можна використовувати будь-яку картинку) на сторінках веб-сайту, створеного в *Microsoft Expression Web*, шляхом винесення необхідних змін у *майстер-сторінки* сайту.

2. Створіть сторінку реєстрації користувача веб-сайту, створеного в *Microsoft Expression Web*, на основі *майстер-сторінки*.

Лабораторна робота №3.

Тема «Введення в JavaScript. Програмна взаємодія з HTML документами на основі DOM API.»

Мета роботи:

1. Ознайомлення з мовою розробки клієнтських веб-сценаріїв JavaScript.
2. Вивчення основ мови та її застосування для автоматизації процесу розмітки і додавання інтерактивних можливостей веб-сторінок.

Теоретичний матеріал

1. Елементи мови JavaScript.

JavaScript дозволяє «оживити» веб-сторінку. Це реалізується шляхом додавання до статичного опису фрагмент виконуваного коду. JavaScript-сценарій може взаємодіяти з будь-якими компонентами HTML-документа і реагувати на зміну їх стану.

JavaScript не є строго типізованою мовою, в змінних можуть зберігатися практично будь-які типи даних.

Як і програма на мові Java, сценарій JavaScript виконується під управлінням інтерпретатора. Однак якщо Java-додаток або Java-аплет компілюється в байтовий код, то сценарій JavaScript інтерпретується на рівні початкового тексту.

Слід зазначити, що мовні конструкції JavaScript збігаються з відповідними засобами C ++ і Java.

1.1.Структура сценарію.

Сценарієм JavaScript вважається фрагмент коду, розташований між дескрипторами <SCRIPT> і </ SCRIPT>:

Текст HTML-документа
<*SCRIPT*>
Код сценарію
</*SCRIPT*>
Текст HTML-документа

1.2. Змінні.

У сценаріях JavaScript змінні можуть зберігати дані будь-яких типів: числа, рядки тексту, логічні значення, посилання на об'єкти, а також спеціальні величини, наприклад «нульове» значення *null* або значення *NaN*, яке повідомляє про неприпустимість операції.

Змінна в мові JavaScript оголошується за допомогою ключового слова **var**. Так, наприклад, вираз

var *selected* = "*first item*";

створює змінну з ім'ям select і присвоює їй як значення рядок символів «*first item*». Змінні можуть оголошуватися також автоматично. Це відбувається при присвоєнні значення змінної, яка не зустрічалася раніше в даному сценарії. Так, у наступному прикладі створюється змінна з імям *rating*, якій присвоюється числове значення, рівне *512.5*:

rating = 512.5;

1.3. Об'єкти.

У мові JavaScript не передбачені засоби для роботи з класами в тому вигляді, в якому вони реалізовані в C ++ або Java. Розробник сценарію не може створити підклас на основі існуючого класу, перевизначити метод або виконати якусь іншу операцію з класом. Сценарію, написаним на мові JavaScript, в основному доступні лише готові об'єкти. Побудову нового об'єкту доводиться виконувати лише в окремих випадках.

Об'єкти містять *властивості* (властивості об'єктів можна порівняти з змінними) і *методи*. Об'єкти, а також їх властивості та методи ідентифікуються *іменами*. Об'єктами є форми, зображення, гіпертекстові посилання та інші компоненти веб-сторінки, HTML-документ, який відображається у вікні браузера, вікно браузера і навіть сам браузер. У процесі роботи JavaScript сценарій звертається до цих об'єктів, отримує інформацію і керує ними.

Крім того, розробнику сценарію на мові JavaScript доступні об'єкти, не пов'язані безпосередньо з HTML-документом. Їх називають *зумовленими*, або *незалежними* об'єктами. За допомогою цих об'єктів можна реалізувати масив, описати дату і час, виконати математичні обчислення і вирішити деякі інші завдання.

Перший об'єкт, з яким необхідно познайомитися, щоб написати найпростіший сценарій, – це об'єкт *document*, який описує HTML документ, я що відображається у вікні браузера. Нижче наведено початковий текст веб-сторінки, що містить сценарій, дії якого зводяться до виведення рядка тексту у вікні браузера.

```
<HTML>
 <HEAD>
 <TITLE>Перший сценарій JavaScript</TITLE></HEAD>
 <BODY>
   <SCRIPT LANGUAGE="JavaScript">
        document.write("Перевірка сценарію JavaScript");
   </SCRIPT>
 </BODY>
</HTML>
```

Імена чутливі до регістрів символів, і якщо ви спробуєте звернутися до поточного документа на ім'я *Document*, інтерпретатор JavaScript відобразить повідомлення про помилку.

Основне призначення сценаріїв JavaScript – створювати динамічно змінювані об'єкти, коректувати зміст HTML-документів в залежності від особливостей оточення, здійснювати взаємодію з користувачем і т.д.

1.4. Операції.

Набір операторів в JavaScript, їх призначення та правила використання в основному збігаються з прийнятими в мові C ++. Винятком є операція, що задається символом «+».

В JavaScript символ «+» визначає як *підсумовування* числових значень, так і *конкатенацію* рядків.

Так, наприклад, в результаті обчислення виразу

sum = 47 + 21;

змінної *sum* буде присвоєно значення *68*, а після виконання операції

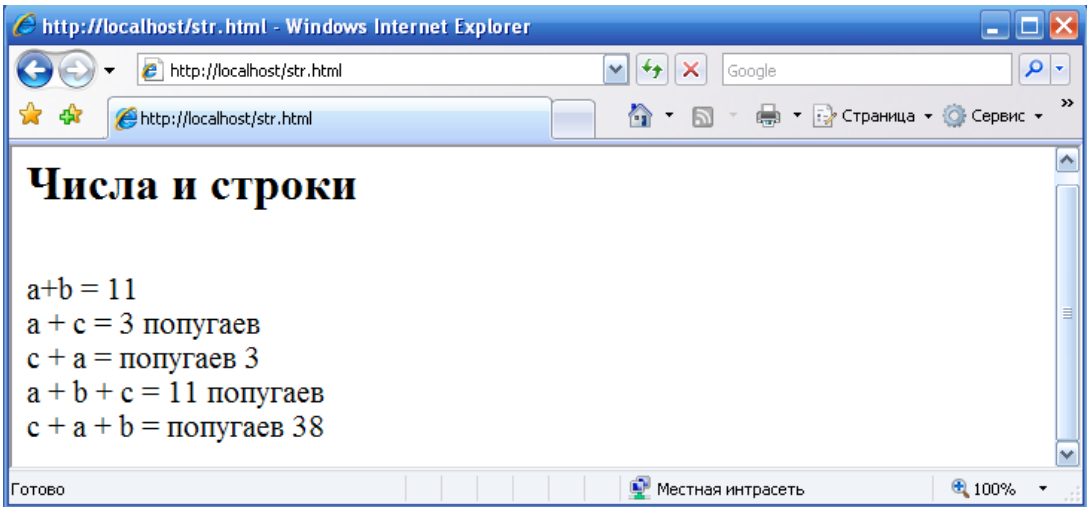
sum = "рядок 1" + "рядок 2";

в змінну *sum* буде записана послідовність символів *"рядок 1 рядок 2"*.

Розглянемо ще один приклад:

```
<HTML>
<BODY>
<H2>Числа и строки</H2><BR>
    <SCRIPT LANGUAGE="JavaScript">

    var a = 3;
    var b = 8;
    var c = " попугаїв ";

    document.write("a+b="); document.write(a + b);
    document.write("<BR>");
    document.write( "a + c = "); document.write(a+c);
    document.write("<BR>");
    document.write("c + a = "); document.write (c + a);
    document.write ("<BR>");
    document.write ("a + b + c = "); document.write(a + b + c);
    document.write("<BR>");
    document.write("c + a + b = "); document.write(c + a + b);
    document.write("<BR>");

    </SCRIPT>
</BODY>
</HTML>
```

У вікні браузера, наведений вище HTML-код виглядає так, як показано на скріншоті:



Перший рядок відображає результат підсумовування двох числових значень, друга і третя - результат конкатенації рядка і символьного представлення числа. Якщо операція підсумовування чисел передує конкатенації, JavaScript обчислює суму чисел, представляє її у символьному вигляді, потім виробляє конкатенацію двох рядків. Якщо ж першою в вираженні вказана операція конкатенації, то JavaScript спочатку перетворює числові значення в символьний вигляд, а потім виконує конкатенацію рядків.

1.5. Керуючі конструкції.

Керуючі конструкції, що використовуються в мові C ++, в основному застосовні і в сценаріях JavaScript.

В JavaScript додатково визначені мовні конструкції, відсутні в C ++, а саме: оператори *for ... in* і *with*.

У *прикладі 1* за допомогою оператора циклу на веб-сторінці формується таблиця множення чисел.

Приклад 1.

```
<html>
<body>
   <table>
   <script language="JavaScript">
        document.write("<tr><td> </td>");
        for (i = 1; i < 10; i++) document.write("<td>"+i+" </td>");
        document.write("</tr>");
        for (i = 1; i < 10; i++)
        {
                document.write("<tr><td>" + i + " </td>");
                for (j = 1; j < 10; j++)
                {
                        document.write("<td  bgcolor='#00ffa0'>" + (i*j) + " </td>");
                        }
                document.write("</tr>");
```

```
        }

    </script>
    </table>
</body>
</html>
```

Окремої уваги заслуговує оператор *new*. Незважаючи на те, що більшість об'єктів уже створені браузером і доступні сценарієм, в деяких випадках доводиться створювати об'єкти в процесі роботи. Це відноситься до зумовленим об'єктам та об'єктам, які визначаються розробником сценарію. Для створення об'єкта використовується оператор *new*, який викликається наступним чином:

змінна = ***new*** *тип_об'єкта (параметри)*

1.6. Функції.

Формат оголошення функції виглядає наступним чином:

function *ім'я функції ([параметри]) тіло функції*

Оголошення функції починається з ключового слова **function**. Так само, як і в мові *C* для ідентифікації функції використовується ім'я, при виклику функції можуть передаватися параметри, а після закінчення виконання повертатися значення. Однак, на відміну від *C*, тип значення і типи установки не задаються. Нижче показані два способи виклику функції:

- *імя_функциі ([параметри]);*
- *змінна = ім'я функції ([параметри]);*

У другому випадку значення, що повертається функцією, присвоюється зазначеної змінної.

1.7. Область видимості змінних.

Робота зі змінними в тілі функції підпорядковується наступним правилам:

1. Якщо змінна оголошена за допомогою ключового слова *var*, доступ до неї здійснюється за правилами, подібним тим, які використовуються в мові *C*.
2. Змінна, оголошена всередині функції, вважається *локальною*. Область видимості такої змінної обмежується тілом функції, в якій вона оголошена.
3. Змінна, оголошена поза функцією, вважається *глобальною*. До неї можна звертатися з будь-якої точки сценарію.
4. Якщо локальна і глобальна змінні мають однакові імена, то в тілі функції локальна змінна «маскує» глобальну.
5. Якщо змінна створюється автоматично, тобто якщо вона не оголошена за допомогою ключового слова *var*, але присутня в лівій частині оператора прямого присвоювання, то вона вважається глобальною і стає доступною з будь-якої точки сценарію.

2. HTML DOM.

DOM (Document Object Model) – являє собою стандарт консорціуму W3C для програмного доступу до документів HTML або XML. Фактично це платформо- та мовно-нейтральний інтерфейс, що дозволяє програмам і сценаріям динамічно звертатися і оновлювати вміст, структуру і стиль документа.

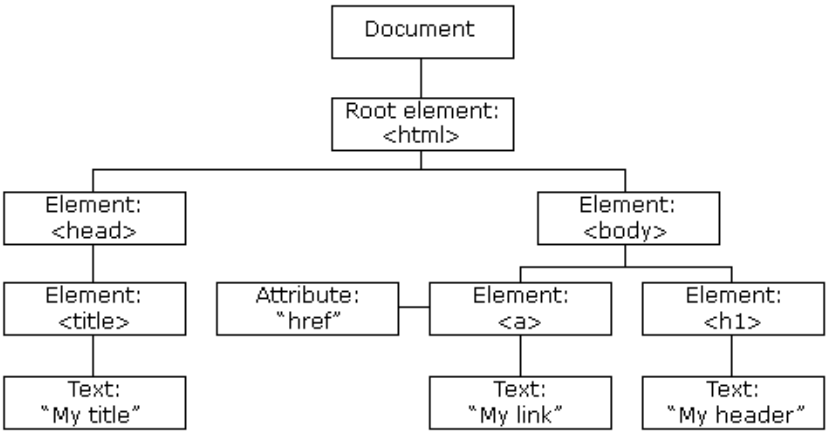
В рамках даного стандарту можна виділити 3 частини:

- Core DOM – стандартна модель будь-якого структурованого документа.
- XML DOM – стандартна модель XML документа.
- HTML DOM – стандартна модель HTML документа.

DOM визначає об'єкти і властивості всіх елементів документа і методи (інтерфейс) для доступу до них.

HTML DOM визначає об'єкти і властивості всіх HTML елементів і методи (інтерфейс) для доступу до них. Інакше кажучи, HTML DOM описує яким чином необхідно отримувати, змінювати, додавати і видаляти HTML елементи.

У відповідності з моделлю DOM все, що міститься всередині HTML документа - є вузлом. Тобто HTML документ представляється у вигляді дерева вузлів, якими є елементи, атрибути та текст.



2.1. Вузли дерева HTML документу.

Згідно моделі DOM:

- Весь документ представляється вузлом документу.
- Кожен HTML тег є вузлом елементу.
- Текст всередині HTML елементів представляється текстовими вузлами.
- Кожному HTML атрибуту відповідає вузол атрибуту.
- Коментарі є вузлами коментарів.

Приклад 2.

```
<html>
 <head>
  <title>HTML документ</title>
 </head>
 <body>
  <h1>Заголовок </h1>
```

```
  <p>Просто текст</p>
 </body>
</html>
```

У цьому прикладі кореневим вузлом є тег <html>. Всі інші вузли утримуються всередині <html>. У цього вузла є два дочірніх вузла: <head> і <body>. Вузол <head> містить вузол <title>, а вузол <body> містить вузли <h1> і <p>.

Слід звернути особливу увагу на те, що текст, розташований у вузлі елемента відповідає текстовому вузлу. У прикладі ***<title>HTML документ</title>*** вузол елемента <title> містить текстовий вузол *«HTML документ»*, тобто *«HTML документ»* не є значенням елемента <title>. Тим не менш, в рамках HTML DOM значення текстового вузла може бути доступно допомогою властивості *inner*HTML.

Всі вузли HTML документа можуть бути доступні за допомогою дерева, при цьому їх вміст може бути змінено або видалено, а також можна додати нові елементи.

Всі вузли дерева знаходяться в ієрархічних відносинах між собою. Для опису цих відносин використовуються терміни *батько, дочірній елемент* і *нащадок*. Батьківські вузли мають дочірні вузли, а дочірні елементи одного рівня називаються нащадками (братами або сестрами).

Відносно вузлів дерева дотримуються наступні принципи:

- Самий верхній вузол дерева називається кореневим.
- Кожен вузол, за винятком кореневого, має рівно один батьківський вузол.
- Вузол може мати будь-яке число дочірніх вузлів.
- Кінцевий вузол дерева не має дочірніх вузлів.
- Нащадки мають загального батька.

2.2. Програмний інтерфейс HTML DOM.

В рамках DOM моделі HTML можна розглядати як безліч вузлових *об'єктів*. Доступ до них здійснюється за допомогою *JavaScript* або інших мов програмування. Програмний інтерфейс DOM включає в себе набір стандартних *властивостей* і *методів*.

Властивості представляють деякі сутності (наприклад, <h1>), а *методи* – дії над ними (наприклад, *додати* <a>).

До типових властивостях DOM відносяться наступні:

- *x.innerHTML* – внутрішнє текстове значення HTML елемента x;
- *x.nodeName* – ім'я **x**;
- *x.nodeValue* – значення **x**;
- *x.parentNode* – батьківський вузол для **x**;
- *x.childNodes* – дочірній вузол для **x**;
- *x.attributes* – вузли атрибутів **x**.

Вузловий об'єкт, який відповідає HTML елементу підтримує такі методи:

- *x.getElementById* (id) – отримати елемент з вказаним **id**;

- *x.getElementsByTagName* (name) – отримати всі елементи з вказаним ім'ям тега (**name**);
- *x.appendChild* (node) – вставити дочірній вузол для **x**;
- *x.removeChild* (node) – видалити дочірній вузол для **x**.

Приклад 3.

Для отримання тексту з елемента <*p*> зі значенням атрибута *id «demo»* в HTML документі можна використовувати наступний код:

```
txt = document.getElementById("demo").innerHTML
```

Такий самий результат може бути отриманий по-іншому:

```
txt=document.getElementById("demo").childNodes[0].nodeValue
```

В рамках DOM можливі 3 способи доступу до вузлів:

1. За допомогою методу *getElementById*(ID). При цьому повертається елемент із зазначеним **ID**.
2. За допомогою методу *getElementsByTagName* (name). При цьому повертаються всі вузли з вказаним імям тега (у вигляді індексованого списку). Перший елемент в списку має нульовий індекс.
3. Шляхом переміщення по дереву з використанням відносин між вузлами.

Для визначення довжини списку вузлів використовується властивість *length*.

Приклад 4.

```
x = document.getElementsByTagName("p");
for (i = 0; i < x.length; i++)
{
        document.write(x[i].innerHTML);
        document.write("<br/>");
}
```

У даному прикладі всередину HTML документа вставляється у вигляді списку текстовий вміст всіх елементів відповідних тегом <p>.

Для навігації по дереву навколо поточного вузла можна використовувати такі властивості:

- *parentNode;*
- *firstChild;*
- *lastChild.*

Для безпосереднього доступу до тегів можна використовувати 2 спеціальні властивості:

- *document.documentElement* – для доступу до кореневого вузла документа;
- *document.body* – для доступу до тегу <*body*>.

2.3. Властивості вузлів.

В HTML DOM кожен вузол є об'єктом, який може мати методи (функції) і властивості. Найбільш важливими є такі властивості:

- *nodeName;*
- *nodeValue;*
- *nodeType.*

Властивість *nodeName* вказує на ім'я вузла. Ця властивість має такі особливості:

- властивість *nodeName* призначена тільки для читання;
- властивість *nodeName* вузла елемента точно відповідає імені тега;
- властивість *nodeName* вузла атрибуту відповідає імені атрибута;
- властивість *nodeName* текстового вузла завжди дорівнює **#text;**
- **в**ластивість *nodeName* вузла документа завжди одно **#document.**

Зауваження: *nodeName* завжди містить ім'я тегу HTML елемента у верхньому регістрі.

Властивість *nodeValue* вказує на значення вузла. Ця властивість має такі особливості:

- властивість *nodeValue* вузла елемента не визначено;
- властивість *nodeValue* текстового вузла вказує на сам текст;
- властивість *nodeValue* вузла атрибуту вказує на значення атрибута.

Властивість *nodeType* повертає тип вузла. Ця властивість призначена тільки для читання.

Найбільш важливими типами вузлів є наступні:

Тип елемента	**Тип вузла**
Element	1
Attribute	2
Text	3
Comment	8
Document	9

2.4. Зміна HTML елементів.

HTML елементи можуть бути змінені за допомогою використання JavaScript, HTML DOM і подій.

У прикладі 5 показано, як можна динамічно змінювати текстовий зміст тегу <p>:

Приклад 5.

```
<html>
  <body>
    <p id="p1">Hello World!</p>
    <script type="text/javascript">
        document.getElementById("p1").innerHTML="New text!";
    </script>
  </body>
</html>
```

3. Діалогові елементи.

В JavaScript підтримується робота з наступними діалоговими елементами інтерфейсу:

1) **Alert**. Застосовується для повідомлення користувача, що працює з веб-браузером.

Синтаксис:

alert ("повідомлення");

2) **Confirm**. Застосовується для вибору користувачем одного з двох варіантів відповіді «Так / Ні». Відповідно *Confirm* повертає значення *true / false*.

Синтаксис:

confirm ("питання");

3) **Prompt**. Застосовується для введення користувачем значення. При натисканні «OK» повертається введене значення, у випадку «Cancel» повертається значення *null*.

Синтаксис:

prompt ("питання / запит", "значення за замовчуванням");

Нижче наводиться код веб-сторінки, в якій користувач має можливість вибрати колір тексту за допомогою діалогового елементу.

Приклад 6.

```
<html>
 <body>
 // здесь будет отображаться текст
   <div id="c" style="color:blue">Вы выбрали цвет текста: черный</div>

   <script language="JavaScript">
   // пользователь выбирает цвет текста
      var tcolor = prompt("Выберите цвет текста: red, blue, green, yellow, black","black");
      // задается текст
       document.getElementById("c").innerHTML = "Вы выбрали цвет текста: " + tcolor;
      // задается цвет текста
       document.getElementById("c").style.color = tcolor;
```

```
    </script>
  </body>
</html>
```

Порядок виконання роботи

Використовуючи *Microsoft Expression Studio 2* (або інший редактор), підготуйте на основі прикладів 1-2, 4-6 відповідні веб-сторінки і перегляньте їх за допомогою веб-браузера.

Контрольне завдання

Створення таблиці випадково вибраних кольорів.

Взявши за основу сценарій побудови таблиці множення, побудуйте таблицю випадково вибраних кольорів. Колір осередки таблиці задається за допомогою атрибута *bgcolor*. Колір комірки описується в рамках трикомпонентної моделі RGB, наприклад: <*td bgcolor* = "# *c0a145*">. Для генерації кожної компоненти можна використовувати генератор випадкових чисел за допомогою методів об'єкту *Math* і перетворення в шістнадцятковий формат:

*color = Math.round (255.0 * Math.random ());*
r = color.toString (16);

Результуючий колір утворюється шляхом конкатенації компонентів:

color = r + g + b;

Зразковий вид результату роботи сценарію:

6852a0	1e6eac	2fa1a8	a378b5	34b1e0	238a8b	378c56	619f8d	581d8
11392a	59d966	ba33aa	631033	a33c65	636319	2fae43	4611f8	7f7794
ba7a67	cacb60	6d7160	3b1cb3	265979	b6bc2e	ff26ce	2d59d	6e4e1
7b677a	c4a6bc	bec14f	85437d	1a106f	583c37	4ea14	852213	59f120
909eb6	f395f	3b130	6c083	33310	41a531	d93bf1	1a3ec9	aab213
325f4	8fed	3693	d054dc	f2c484	ac8ef	3aea6	80afbe	e4bcd6
b03872	b13fe	5d7966	71585a	9c845e	233be1	8fded	ab4870	18ccf4
2e4287	cdf15f	927c81	8b7d6e	33c468	eec091	feba0	7b739c	2df9df
90a82d	a66dcf	a7a5f0	5a77f7	b1e64a	9658c7	d5cb45	ea2e82	b1b02d

Лабораторна робота №4.

Тема «Клієнтські сценарії. Використання регулярних виразів.»

Мета роботи:

1. Отримати уявлення про загальні принципи обробки в JavaScript подій, пов'язаних з вікном веб-браузера, веб-сторінкою, що міститься в браузері і елементами документа.
2. Навчитися використовувати найпростіші елементи регулярних виразів для пошуку підрядків, структура і зміст яких описується нетривіальним шаблоном.

Теоретичний матеріал

1. Обробка подій в JavaScript.

Популярність JavaScript багато в чому обумовлена саме тим, що написаний на ньому сценарій може реагувати на дії користувача і інші зовнішні події. Кожна з подій пов'язана з тим чи іншим об'єктом: формою, гіпертекстовим посиланням або навіть з вікном, що містить поточний документ.

Як приклади зовнішніх подій, на які можуть реагувати об'єкти JavaScript, можна навести наступні.

- закінчення завантаження документа у вікно (або закінчення завантаження документів у всі фрейми вікна). Ця подія пов'язана з об'єктом *window*;
- клацання мишею на об'єкт. Ця подія може бути пов'язана з інтерактивним елементом форми або з гіпертекстовим посиланням;
- отримання об'єктом фокусу введення. Ця подія може бути пов'язана з об'єктами типу Text, Password і з іншими інтерактивними елементами;
- передача на сервер даних, введених користувачем за допомогою інтерактивних елементів. Зв'язується з формою.

Обробка події проводиться за допомогою спеціально призначеного для цього фрагмента коду, званого *обробником події*. Для кожної події JavaScript надає свій обробник. Однак при побудові сценарію можна створювати власний обробник події і використовувати його замість обробника, заданого за замовчуванням.

Ім'я обробника визначає, яку подію він повинен обробляти. Так, для того щоб сценарій потрібним чином відреагував на клацання мишею, використовується обробник з ім'ям *onClick*, для обробки події, що полягає в отриманні фокусу введення, – обробник *onFocus*.

Для того щоб вказати інтерпретатору JavaScript на те, що обробкою події повинен займатися обробник, необхідно включити в HTML-дескриптор такий вираз:

Ім'я_обробника="*команди_ обробника*"

Цей вираз включається в тег, який описує об'єкт, з яким пов'язана подія.

Наприклад, якщо необхідно обробити подію, яка полягає в отриманні фокусу полем введення, дескриптор, що описує цей інтерактивний елемент, повинен мати приблизно такий вигляд:

<*Input type* ="*text*"*name*="*Inform*" *onFocus*="*handleFocus();*">

Ім'я обробника є одним з атрибутів HTML-дескриптора, а команди, призначені для обробки події, виступають в ролі значення цього атрибута. В даному випадку обробка події проводиться в тілі функції *handleFocus().* В принципі, оброблювачем може бути не тільки функція, але і будь-яка послідовність команд JavaScript у вигляді складеного оператора.

Наступний приклад демонструє обробку події, яка пов'язана з наведенням курсору миші на гіперпосилання:

```
<a href = ”http://www.myhp.edu” onmouseover="alert('An onMouseOver event'); return false">
        <img src="my.gif" width="50" height="50">
</a>
```

Нижче наводиться повний текст HTML документа с JavaScript сценарієм, в якому обробляється подія натиснення кнопки миші, і визначається, яка саме з них була натиснута:

Приклад 1.

```
<html>
        <head>
        <script language = "javascript">
        function whichButton(event)
        {
                if (event.button == 2)
                  {
                        alert("Ви клацнули правою кнопкою миші!");
                  }
                else
                  {
                        alert("Ви клацнули лівою кнопкою миші!");
                  }
        }
        </script>
        </head>

        <body onmousedown="whichButton(event)">
           <p> Клацніть клавішею миші в будь-якому місці документа </p>
        </body>

</html>
```

Таким чином, для того щоб обробити якусь стандартну подію в браузері, необхідно додати у відповідний HTML-тег атрибут, який відповідає цій події, вказавши як значення атрибуту ім'я JavaScript функції. Список атрибутів, які визначені для HTML-тегів наводиться нижче:

IE: *Internet Explorer*, **F**: *Firefox*, **O**: *Opera*, **W3C**: *стандарт*

Атрибут	Опис	Номер версії браузера			W3C
		IE	F	O	
onabort	Перерване завантаження зображення	4	1	9	Так
onblur	втрата фокусу елементом	3	1	9	Так
onchange	Зміна вмісту в поле вводу	3	1	9	Так
onclick	Клацання миші на об'єкті	3	1	9	Так
ondblclick	Подвійне клацання миші на об'єкті	4	1	9	Так
onerror	Помилка при завантаженні зображення або документа	4	1	9	Так
onfocus	Отримання фокуса елементом	3	1	9	Так
onkeydown	Натискання клавіші	3	1	Ні	Так
onkeypress	Клавіша натиснута	3	1	9	Так
onkeyup	Відтискання клавіші	3	1	9	Так
onload	Завершення завантаження сторінки або зображення	3	1	9	Так
onmousedown	Натискання кнопки миші	4	1	9	Так
onmousemove	Переміщення курсору миші	3	1	9	Так
onmouseout	Зсув курсору миші з об'єкта	4	1	9	Так
onmouseover	Наведення курсору миші на об'єкт	3	1	9	Так
onmouseup	Відтискання кнопки миші	4	1	9	Так
onreset	Кнопка «Reset» натиснута	4	1	9	Так
onresize	Зміна розміру вікна	4	1	9	Так
onselect	Виділення тексту	3	1	9	Так
onsubmit	Кнопка «Submit» натиснута	3	1	9	Так
onunload	Вихід з веб-сторінки	3	1	9	Так

2. Регулярні вирази

Регулярні вирази – система пошуку текстових фрагментів в електронних документах, заснована на спеціальній системі запису зразків для пошуку.

Зразок, що задає правило пошуку, називається «*шаблоном*». Застосування регулярних виразів принципово змінило технології електронної обробки текстів.

За допомогою регулярних виразів можна задавати структуру шуканого шаблону і його позицію всередині рядка (наприклад, на початку або в кінці рядка, на кордоні або не на межі слова).

При описі структури шаблону використовуються:

- гнучка система квантифікаторів (операторів повторення);

– оператори опису наборів символів та їх типу (числові, нечислові, спеціальні).

Для того, щоб задати положення шуканого фрагменту усередині рядка, можна використовувати один з наступних операторів:

Подання	Позиція
^	Початок рядка
$	Кінець рядка
\b	Границя слова
\B	Не границя слова
(?=шаблон)	Шуканий рядок випливає після зазначеного рядка (з переглядом вперед)
(?!=шаблон)	Шуканий рядок не випливає після зазначеного рядка (з переглядом вперед)
(?<=шаблон)	Шуканий рядок випливає після зазначеного рядка (з переглядом назад)
(?<!шаблон)	Шуканий рядок не випливає після зазначеного рядка (з переглядом назад)

Крім того, мова регулярних виразів надає набір квантифікаторів, що дозволяють вказати кількість повторень шаблону:

Подання	Число повторень
{*n*}	Рівне *n*
{*m*, *n*}	Від *m* до *n* включно
{*m*,}	Не менеше *m*
{,*n*}	Не більше *n*

Є і більш прості квантифікатори:

Подання	Число повторень	Еквівалент
*	Нуль або більше	{*0*,}
+	Одне або більше	{*1*,}
?	Нуль або одне	{*0*,*1*}

Для того, щоб задати всередині шаблону групи символів можна використовувати наступні оператори:

Оператор	Опис
[xyz]	Будь-який символ із зазначеної множини
[^xyz]	Будь-який символ що не входить у вказану множину
[x-z]	Будь-який символ з зазначеного діапазону
[^x-z]	Будь-який символ що не входить у вказаний діапазон
. (*крапка*)	Будь-який символ, окрім символів розриву або переносу рядка
\w	Будь-який буквено-цифровий символ, включаючи символ підкреслення
\W	Будь-який не буквений символ

\d	Будь-яка цифра
\D	Будь-який нецифровий символ
\s	Будь-який не відображаємі символ
\S	Будь-який символ (крім не відображаємих символів)

Для угруповання окремих частин шаблону можна використовувати наступні оператори:

Оператор	Опис
( )	Пошук групи символів всередині дужок і збереження знайденої відповідності
(?:)	Пошук групи символів всередині дужок без збереження знайденої відповідності
|	Комбінування частин в одне вираження з подальшим пошуком кожної із частин окремо. Аналогічно оператору «АБО».

Якщо шаблон пошуку включає спеціальні (як правило не відображаємі) символи, для їх опису можна використовувати такі позначення:

Позначення	Опис
\0	Символ з нульовим кодом
\n	Символ нового рядка
\r	Символ початку рядка
\t	Символ табуляції
\v	Символ вертикальної табуляції
\xxx	Символ, що має заданий восьмеричний ASCII код *xxx*
\xdd	Символ, що має заданий шістнадцятковий ASCII код *dd*
\uxxxx	Символ, що має ASCII код виражений Юнікоді *xxxx*

Квантифікаторам в регулярних виразах відповідає максимально довгий рядок з можливих (тобто квантифікатори є «жадібними»). Це може призводити до деяких проблем. Наприклад, шаблон (<. *>), що описує на перший погляд HTML-теги, насправді буде виділяти більші фрагменти в документі.

Наприклад, рядок виду

<P> <font color = 'blue'> <i> Регулярні вирази <i> </ font> - зручний інструмент для пошуку в рядках </ p>, який формально відповідає зазначеному вище шаблону

Для вирішення даної проблеми можна використовувати два підходи:

1. У регулярному виразі враховуються символи, що не відповідають бажаному зразком (наприклад, <[^>] *> для вищеописаного випадку).
2. Визначення квантифікатору як нежадібного (ледачого) – більшість реалізацій дозволяють це зробити, додавши після нього знак питання.

Наприклад, за шаблоном (<. *?>) будуть знайдені всі теги з розглянутого рядка.

Таким чином, виходять такі «нежадібні» модифікації квантифікаторів:

Квантифікатор	Опис
*?	«Не жадібний» еквівалент *
+?	«Не жадібний» еквівалент +
{n,}?	«Не жадібний» еквівалент {n,}

Слід, однак, мати на увазі, що використання «ледачих» квантифікаторів може призвести до ситуації, коли виразу відповідає занадто короткий, зокрема, порожній рядок.

3. Використання регулярних виразів в JavaScript

При пошуку по тексту можна використовувати шаблон, що описує підрядок. В JavaScript такий шаблон може бути описаний за допомогою об'єкту *RegExp*. У найпростішому випадку такий шаблон описує окремий символ, однак має сенс його використовувати для регулярних виразів.

Наступний нижче код описує *RegExp* об'єкт з ім'ям *pttn*, що містить регулярний вираз, який описує ціле десяткове число:

```
var pattn = new RegExp("/[0-9]+/");
```

Об'єкт *RegExp* має 3 вбудованих методи: *test()*, *exec()* і *compile()*.

Метод *test()* виконує пошук за шаблоном:

```
var pattn = new RegExp("[0-9]+");
document.write(pattn.test("38 попугаїв"));
```

Результат:

```
true
```

Метод *exec()* виконує пошук підрядка за шаблоном і повертає знайдені відповідності; якщо відповідностей немає, повертається значення *null*:

```
var pattn=new RegExp("[0-9]+");
document.write(pattn.exec("38 попугаїв"));
```

Результат:

```
38
```

Якщо необхідно знайти всі відповідності, то при виклику конструктора *RegExp* слід вказати додатковий параметр "g", який вказує на необхідність глобального пошуку.

Приклад 2.

```
var pattn = new RegExp("[0-9]+", "g");
do
{
        result  = pattn.exec("1 попугай, 2 попугая,…, 38 попугаїв");
        document.write(" " + result);
}
while (result != null)
```

Результат:

```
1 2 38 null
```

Метод *compile()* застосовується для зміни раніше створеного шаблону.
Приклад 3.

```
var pattn = new RegExp("[0-5]+");
document.write(pattn.exec("38 попугаїв"));

pattn.compile("[6-9]+");
document.write(";" + pattn.exec("38 попугаїв"));
```

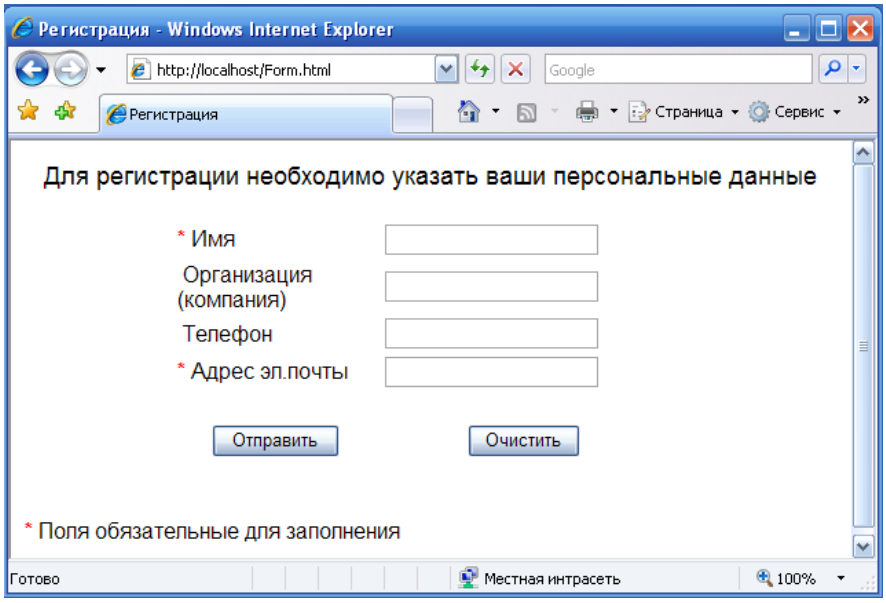
Результат:

```
3;8
```

Порядок виконання роботи

Перевірка значень, введених користувачем в поля форми для реєстрації.

1) Для виконання лабораторної роботи необхідно створити веб-сторінку, що містить форму з полями, наступного виду:



2) У тегу <form> додайте обробник події відправки даних виду:

```
onSubmit = "CheckData(); return false;"
```

У даному випадку вказана функція обробник *CheckData()*. Оператор *return false;* запобігає автоматичне відправлення даних після виконання функції-обробника. Відправка даних буде виконуватися з обробника.

3) Додайте на сторінці секцію JavaScript коду, що описує функцію-обробника:

```
function CheckData()
{
  var ans;
  ans = confirm("Вы уверены, что хотите отправить введенные данные ?");
  if (ans) submit();
}
```

Як це видно з коду, функція *CheckData()* у разі підтвердження з боку користувача самостійно викликає метод *submit()* для передачі даних з форми.

4) Тепер необхідно додати перевірку значень, введених в поля форми користувачем.

Перш за все, необхідно переконатися в тому, що заповнені всі поля, обов'язкові для введення. Для цього можна використовувати перевірку на рівність нулю довжини рядка (властивість *length*), що є значенням вузла дерева документу, що відповідає полю введення, наприклад: *document.getElementById ("uname"). Value.length.*

Наступна перевірка повинна контролювати структуру і зміст полів. Для цього можна використовувати об'єкт *RegExp*, наприклад:

```
var validEMail, pattn;

pattn = new RegExp("^[\.\-_A-Za-z0-9]+?@[\.\-A-Za-z0-9]+?\.[A-Za-z0-9]{2,6}$");

validEMail = pattn.test(document.getElementById("email").value));
```

У цьому фрагменті описана перевірка структури електронної адреси з поля форми з ідентифікатором "*email*". Для перевірки був використаний шаблон на основі регулярного виразу.

Контрольне завдання

Самостійно побудуйте регулярний вираз, що описує шаблон для перевірки номера телефону, і внесіть всі необхідні зміни і доповнення в функцію *CheckData()*.

Лабораторна робота №5.

Тема «Розробка CGI-додатків на Perl і PHP.»

Мета роботи:
Ознайомлення з:

1. основами мов розробки веб-сценаріїв на мовах Perl і PHP;
2. синтаксисом мов Perl і PHP;
3. реалізацією обробки даних, отриманих від клієнтського додатка, на стороні веб-сервера.

Теоретичний матеріал

1. Основи розробки сценаріїв на мові Perl

Програма на мові Perl складається з *декларацій* і *операторів*. Будь-який текст, починаючи з символу «#» і до кінця рядка, вважається коментарем і ігнорується.

Для *змінних* декларацій не потрібно. До тих пір доки їм не буде присвоєно яке-небудь конкретне значення, вони просто містять невизначене значення *undef*. *Декларації* можуть розташовуватися в будь-якому місці програми, так як обробляються на етапі компіляції, що передує етапу виконання програми.

Оператори мови Perl підрозділяються на *прості* і *складові*. *Складові* оператори складаються з блоків, укладених у фігурні дужки. На відміну від мови С, фігурні дужки в складових операторах обов'язкові, навіть якщо в них укладений тільки один оператор. Оператори розділяються крапкою з комою.

Знайомство з Perl можна починати зі сценарію 1.

Сценарій 1. *Виведення рядка вітання на Perl*.

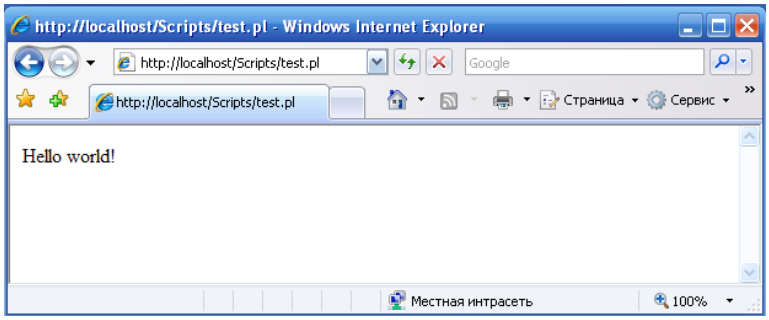
```
#!c:/perl/bin/perl
print "Content-type:text/html\n\n";
print "Hello world!";
```

Перший рядок сценарію, оформлений у вигляді коментаря, що вказує на фізичне розміщення інтерпретатора мови Perl. Решта рядків фактично формують відповідь веб-сервера.

У другому рядку у вихідний потік передається поле *Content-type* заголовка відповіді сервера, і врешті вставляється порожній рядок, що відокремлює заголовок від тіла відповіді сервера.

В останньому рядку поміщається зміст тіла відповіді сервера.

У результат виконання сценарію даного сценарію отримаємо наступну сторінку:



У наступному прикладі буде розглянуто сценарій, що зчитує і обробляє дані, отримані веб-сервером із запиту клієнта. Вихідні дані повинні вводитися користувачем в поля форми веб-сторінки, завантаженої в веб-браузері.

Сценарій 2. *Виведення списку параметрів, отриманих сервером в запиті від клієнта.*

```
#!c:/perl/bin/perl

print "Content-type: text/html\n\n";
print "<HTML><BODY>\n";

$method = $ENV{'REQUEST_METHOD'};

if ($method eq 'POST')
{
        $length = $ENV{'CONTENT_LENGTH'};
        read(STDIN, $qstr, $length);
}
else
{
        if ($method eq 'GET') { $qstr = $ENV{'QUERY_STRING'}; }
        else
        {
                print "Method ".$method." is not supported </BODY></HTML>";
                #exit(0);
        }
}

print "<P>Метод = ", $method;
print "<p>Строка параметров: <p>\n";
print $qstr;

# зворотнє перекодування

$qstr =~ tr/+/ /;
$qstr =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;

# виділення списку параметрів

print "<p>А теперь отдельные поля:<p>";
@pars = split(/&/, $qstr);

# визначення розміру масиву

$n_pars = @pars;

# виділення імені і значення для кожного параметра

for ($i=0; $i<$n_pars; $i++)
{

# виділення списку з двох змінних $name та $value

        ($name, $value) = split(/=/, $pars[$i]);
        print "Параметр <B>", $name, "</B> дорівнює <I>", $value, "</I><br>";
```

```
}
print "</HTML></BODY>\n";
```

Даний сценарій орієнтований на передачу даних з веб-форми одним з основних методів: *GET* або *POST*. Оскільки передача даних у цих методах відрізняється, то в сценарії спочатку визначається метод передачі даних шляхом звернення до змінної оточення *REQUEST_METHOD*, значення якої є сценарієм через однойменний елемент асоціативного масиву (хешу) *ENV*, що містить значення всіх змінних оточення.

Після визначення, що використовувався в запиті клієнта методу, вибирається адекватний спосіб читання параметрів, отриманих з веб-форми:

- У випадку методу *POST* визначається загальний розмір переданих даних (в байтах) із змінної оточення *CONTENT_LENGTH*, а потім блок даного розміру зчитується з вхідного потоку *STDIN* за допомогою функції *read*.
- У випадку методу *GET* дані доступні в змінній оточення *QUERY_STRING*.
- Якщо метод запиту не збігається ні з одним з розглянутих вище або його значення не визначено, то відбувається примусове завершення сценарію з видачею відповідного повідомлення.

Оскільки в рамках протоколу HTTP символи, відмінні від латинських букв і цифр передаються у вигляді шістнадцяткових кодів (прогалини замінюються на «+»), потрібно попереднє зворотне перетворення отриманих даних за допомогою операторів заміни ***s///*** і ***tr//*** з використанням шаблонів у вигляді регулярних виразів.

Далі за допомогою функції *split* відбувається поділ блоку символів на *підрядки* по заданому *роздільнику*:

- *Пари*, що описують ім'я параметра і його значення поділяються за допомогою символу «&». Результат поділу поміщається в скалярний масив @pars.
- Всередині пари *ім'я* і *значення* розділяються символом «=».

Розглянутий приклад може бути взятий за основу для будь-якого сценарію, що обробляє дані з веб-форми, оскільки дозволяє отримати вихідні дані, отримані від клієнта. Розширення сценарію зводиться до додавання коду обробки отриманих даних і формування підсумкового документа, що повертається веб-сервером клієнтові.

Розглянемо для прикладу сценарій, що виконує чотири арифметичні операції над цілими числами, які користувач вводить через поля веб-форми в браузері.

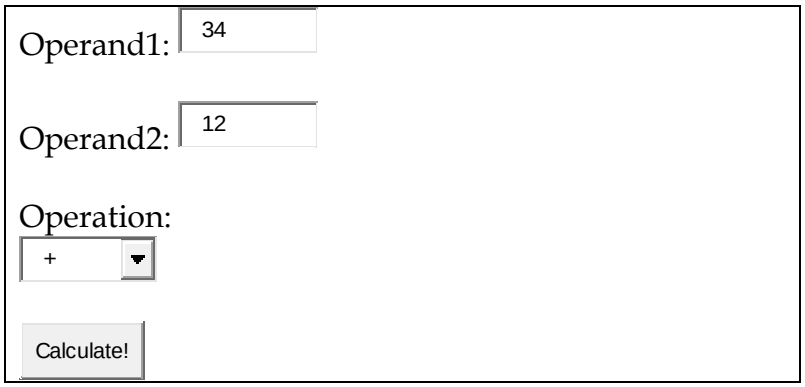
Сценарій 3. *Калькулятор арифметичних операцій для цілих операндів*.

```
#!c:/perl/bin/perl

print "Content-type: text/html\n\n";
print "<HTML><BODY>\n";
```

```
$method = $ENV{'REQUEST_METHOD'};

if ($method eq 'POST')
{
        $length = $ENV{'CONTENT_LENGTH'};
        read(STDIN, $qstr, $length);
}
else
{
        if ($method eq 'GET') { $qstr = $ENV{'QUERY_STRING'}; }
        else
        {
                print "Method ".$method." is not supported </BODY></HTML>";
                exit(0);
        }
}

$qstr =~ tr/+/ /;
$qstr =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;

@pars = split(/&/, $qstr);
$n_pars = @pars;

foreach $par (@pars)
{
        ($name, $value) = split(/=/, $par);
        $opers{$name} = $value;
}

$op1 = int($opers{'A'});
$op2 = int($opers{'B'});
$op = $opers{'op'};

switch:
        {
                if ($op eq '+') { $res = $op1 + $op2; last switch; }
                if ($op eq '-') { $res = $op1 - $op2; last switch; }
                if ($op eq '*') { $res = $op1 * $op2; last switch; }
                if ($op eq '/') {
                                        if ($op2 == 0)
                                        {
                                           print " Divide by zero! </BODY></HTML>";
                                           exit(0);
                                        }
                                        else
                                        {
                                           $res = $op1 / $op2; last switch;
                                        }
                                       }
```

```
                {
                  print "Operator ".$op." is not supported </BODY></HTML>";
                  exit(0);
                }
        }

print "Result: ".$op1.$op.$op2.' = '.$res;
print "<p><a href=\'".$ENV{'HTTP_REFERER'}."\'>Back</a></p>";

print "</HTML></BODY>\n";
```

У даному прикладі з рядка запиту витягуються 3 параметри з іменами «A», «B» і «op», після чого в залежності від значення третього параметра виконується відповідна арифметична операція над першими двома параметрами.

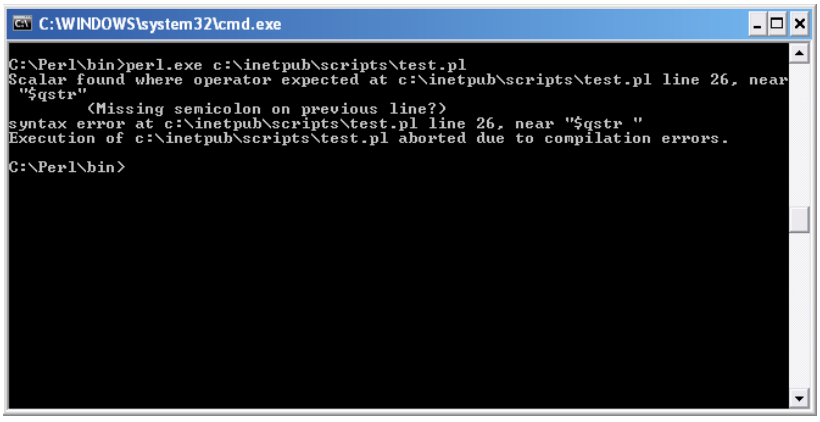


У результаті формується *html* сторінка з результатом обчислення і зворотнім посиланням на вихідну сторінку з формою. Для визначення адреси вихідної сторінки використовується значення змінної оточення *HTTP_REFERER*:

Result: 34*12 = 408

Back

Для пошуку синтаксичних помилок в сценаріях на Perl можна використовувати безпосередній запуск інтерпретатора в командному рядку, який знаходиться в установчому каталозі Perl:



2. Основи розробки сценаріїв на мові PHP

PHP – сценарії можуть розміщуватися в окремому файлі (з розширенням *.php*) або вбудовуються безпосередньо в HTML документ.

Існує кілька способів впровадження коду PHP в HTML документи:

- За допомогою відкриваючого тегу ***<? Php*** і закриваючого тегу ***?>***.
- За допомогою коротких тегів *<?* та *?>*. Дана можливість доступна тільки при спеціальному налаштуванні.
- За допомогою тегів *<script language = "php">* та *</ script>*.
- Шляхом використання *echo* тегів в стилі ASP: *<%* та *%>*. Така можливість доступна при відповідному конфігураційному налаштуванні.

Надалі в прикладах буде використовуватися перший з варіантів впровадження PHP коду.

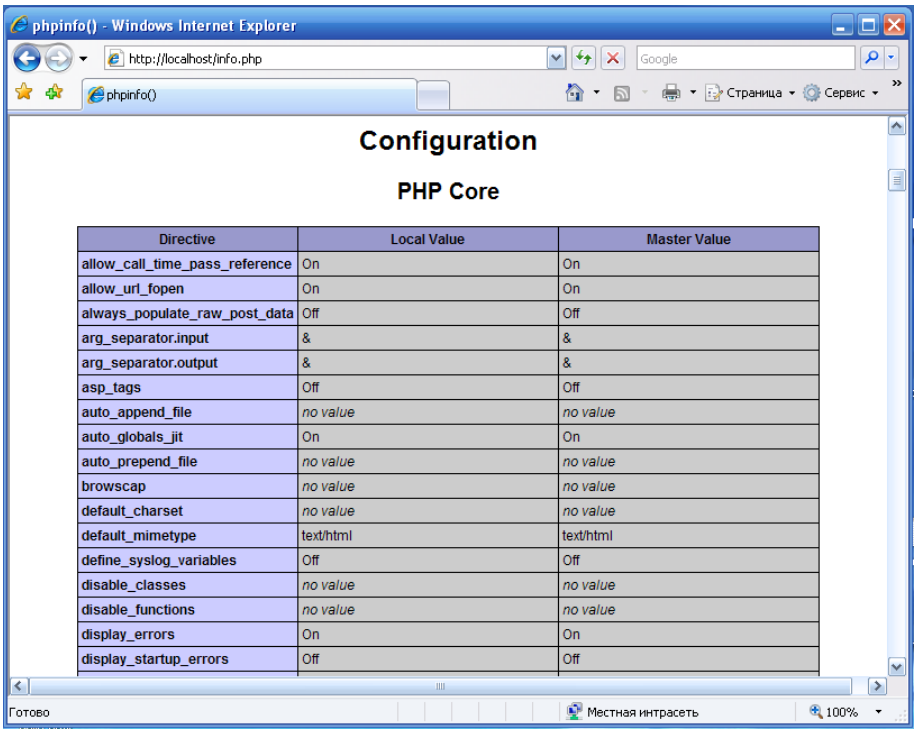
Код, який знаходиться всередині зазначених тегів, обробляється інтерпретатором PHP, весь інший код залишається незмінним.

Для того, щоб побачити поточні налаштування PHP, і для перевірки його працездатності корисно використовувати спеціальну функцію *phpinfo()*:

Сценарій 4. *Використання php-функції* ***phpinfo()***.

```
<?php

        phpinfo();
?>
```

Після виконання цього коду, в веб-браузері можна буде побачити приблизно наступне (показана невелика частина):



Directive	Local Value	Master Value
allow_call_time_pass_reference	On	On
allow_url_fopen	On	On
always_populate_raw_post_data	Off	Off
arg_separator.input	&	&
arg_separator.output	&	&
asp_tags	Off	Off
auto_append_file	*no value*	*no value*
auto_globals_jit	On	On
auto_prepend_file	*no value*	*no value*
browscap	*no value*	*no value*
default_charset	*no value*	*no value*
default_mimetype	text/html	text/html
define_syslog_variables	Off	Off
disable_classes	*no value*	*no value*
disable_functions	*no value*	*no value*
display_errors	On	On
display_startup_errors	Off	Off

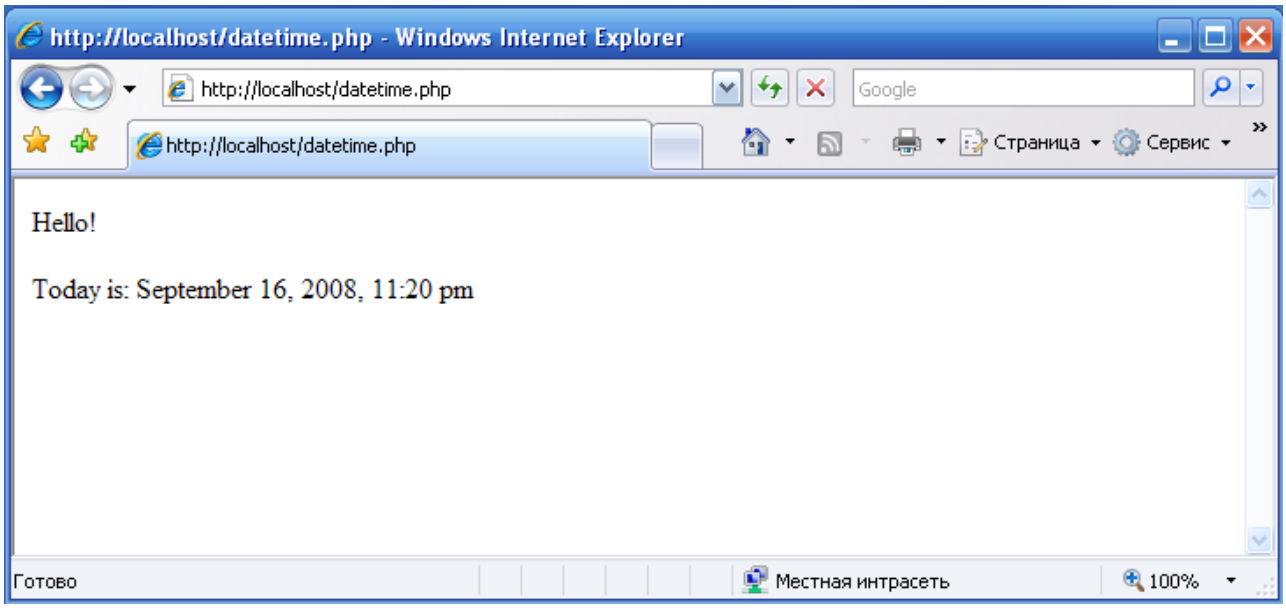
В цілому ж лістинг містить інформацію про встановлені опції і розширення PHP, версії PHP, інформацію про веб-сервер і змінних оточення, інформацію про версію ОС, шляхах, налаштуваннях конфігураційних змінних, полях заголовка HTTP і PHP ліцензії.

Наступний приклад демонструє варіант з впровадженням PHP коду в HTML:

Сценарій 5. *Впровадження PHP коду в HTML документ.*

```
<html>
<body>
<p>Hello! </p>
<p>Today is:
<?php
        $today = date("F j, Y, g:i a");
        echo($today);
?>
</p>
</body>
</html>
```

Результат обробки цього документа представлений нижче:



Одним з головних завдань, що вирішуються за допомогою PHP, є обробка даних, одержуваних від користувача через веб-форми. Розглянемо, яким чином в PHP реалізується така обробка.

Сценарій 6. *Виведення списку параметрів, отриманих сервером в запиті від клієнта.*

```
<?php
        $method = $_SERVER["REQUEST_METHOD"];

        if ($method == "GET") $query = "_GET";
        elseif ($method == "PUT") $query = "_PUT";
        else die("$method is not supported!");

        print "<p><b>Method</b>: $method </p>";
        print "<p><u>Params:</u></p>";

        foreach ($$query as $name => $value)
        {
                print "<b>$name</b> = <i>$value</i> <br>";
        }
?>
```

У першому рядку сценарію за допомогою змінної оточення *REQUEST_METHOD* з глобальної асоціативного масиву *$_SERVER* визначається метод передачі даних в запиті клієнта. Залежно від обраного методу передані дані будуть вилучатись або з глобальної змінної *$ _GET* або з *$_PUT*. Якщо метод відрізняється від *GET* або *PUT*, або невизначений, то відбувається примусове завершення роботи сценарію з видачею повідомлення через виклик функції *die()*. Загалом, можна також використовувати глобальний масив *$ _REQUEST*, що містить усередині себе масиви *$_GET*, $_POST і *$_COOKIE*, що дозволяє позбавитися від перевірки методу передачі.

Конструкція виду *$$query* демонструє непряме звернення до змінної, тобто змінна *$query* містить ідентифікатор іншої змінної, і для звернення до неї необхідно додати ще один знак *$*.

Обидві змінні *$_GET* і *$_POST* є асоціативними масивами, тому для перебору елементів був використаний спеціальний оператор:

foreach (ім'я_масиву **as** *ключ* => *значення*)

Слід звернути увагу на те, що всередину строкових констант, обмежених символами " " можна вставляти змінні. Після обробки такого рядка інтерпретатором замість змінної вставляється її фактичне значення. Також для конкатенації рядків можна використовувати оператор '.' В цілому можна сказати, що в PHP можна використовувати оператори розгалуження, вибору і циклів аналогічні тим, що використовуються в мові С.

Код сценарію, що реалізовує чотири арифметичні операції над цілими числами представлений в лістингу нижче:

Сценарій 7. *Калькулятор арифметичних операцій для цілих операндів*.

```
<?php
        $method = $_SERVER["REQUEST_METHOD"];
```

```
        if ($method == "GET") $query = "_GET";
        elseif ($method == "PUT") $query = "_PUT";
        else die("$method is not supported!");

        $q = $$query;

        $a = $q["A"];
        $b = $q["B"];
        $op = $q["op"];

        switch ($op)
        {
          case '+': $result = (int)$a + (int)$b; break;
          case '-': $result = (int)$a - (int)$b; break;
          case '*': $result = (int)$a * (int)$b; break;
          case '/': {
                        if ($b == '0') die("divide by zero!");
                    }
          default: die("operator $op is not defined");
        }
        print "<p>Result: $a $op $b = $result </p>";
        print "<p><a href=".$_SERVER['HTTP_REFERER'].">Back</a></p>";
?>
```

У PHP є широкий діапазон функція для роботи з файлами. Наприклад, наступний приклад демонструє читання файлу, в якому знаходиться виконуваний PHP код:

Сценарій 8. *Читання файлу, що містить PHP код, виконуваного сценарію.*

```
<?php
        $fh = fopen("read.php","r");
        if (!$fh) die("Cannot open file");

        while (!feof ($fh))
        {
           $line = fgets($fh);
           echo $line,"<br>";
        }
        fclose($fh);
?>
```

У даному прикладі використовуються практичні ті ж самі функції, що і в мові С:

- **fopen** (*шлях_до_файлу, тип_доступу*) – відкриття файлу;
- **feof** (*указатель_на_файл*) – перевірка на наявність ознаки кінця файлу;
- **fgets** (*указатель_на_файл*) – читання рядка з файлу;
- **fclose** (*указатель_на_файл*) – закриття файлу.

Для запису даних у файл можна використовувати функцію **fputs** (*указатель_на_файл, рядок*). У наступному прикладі згенеровані функцією *rand()* псевдовипадкові числа зберігаються у файлі *rand.dat*:

Сценарій 9. *Збереження у файлі послідовності псевдовипадкових чисел.*

```
<?php

        $n = 10;

        $fh = fopen("rand.dat","w");

        if (!$fh) die("Cannot open file");

        srand();
        for ($i=0; $i < $n; $i++)
        {
                $d = rand(0,100);
                fputs($fh,"$d\n");
        }
        fclose($fh);
?>
```

Перед виконанням даного сценарію слід правильно налаштувати права доступу для веб-сервера до директорії, в якій зберігається файл *rand.dat*.

Порядок виконання роботи

Для практичного вивчення прикладів в даній лабораторній роботі необхідна наявність встановлених і правильно сконфігурованих інтерпретаторів мов *Perl* і *PHP*.

Частина 1. Мова Perl.

1. Створіть файл з текстом *сценарію 1*. Файл повинен мати розширення pl, і розміщуватися в директорії *Scripts* (або *cgi-bin*). Перевірте налаштування доступу до папки для веб-сервера, який повинен мати право на виконання сценаріїв в цій папці.

При збереженні файлу в редакторі також слід також вибрати правильне кодування символу переносу рядка. Крім того, перший рядок сценарію повинен містити правильний шлях до директорії, в якій встановлено інтерпретатор мови Perl (зазвичай це файл *perl.exe* або подібний йому).

Якщо текст сценарію містить синтаксичні помилки, то після спроби його запустити на виконання веб-сервером, останній поверне клієнтові відповідь, що містить код внутрішньої помилки сервера. Тому перед запуском сценарію рекомендується виконати його перевірку. Для перевірки сценарію на наявність синтаксичних помилок зручно використовувати безпосередній запуск інтерпретатора вручну. Для цього в командному рядку з директорії, зазначеної в

першому рядку сценарію, необхідно запустити виконуваний модуль (зазвичай *perl.exe*) з аргументом, що є ім'ям файлу (із зазначенням шляху), що містить текст сценарію. При наявності синтаксичних помилок в сценарії інтерпретатор видасть повідомлення із зазначенням відповідних номерів рядків у файлі, в яких ці помилки виявлені. Виправляйте помилки доти, поки інтерпретатор не перестане видавати повідомлення про помилки.

Для того, щоб подивитися роботу сценарію, необхідно в браузері набрати його URL по HTTP-протоколу.

2. Підготуйте файл з текстом *сценарію 2*.

Для перевірки працездатності даного сценарію можна виконати його безпосередній запуск у веб-браузері через його URL з додаванням рядка параметрів, наприклад:

http://localhost/Scripts/test.pl?a=2&b=14

Підготуйте HTML сторінку, що містить форму з полями для введення даних. Вставте в тегу <*FORM*> атрибут *ACTION* зі значенням, рівним URL сценарію, як метод виконання запиту вкажіть в атрибуті *METHOD* значення *GET*. Перевірте роботу форми. Зробіть те ж саме, але для методу *POST*.

3. Підготуйте файл з текстом *сценарію 3* і HTML сторінку, що містить форму з полями для введення операндів (прості поля для введення тексту з іменами «*A*» і «*B*») і вибору арифметичної операції (поле типу «*select*» з ім'ям «*op*»). Додайте кнопку типу «*submit*» і атрибут *ACTION* зі значенням, рівним URL сценарію, в тегу <*FORM*>:

```
<html>
<body>
        <form action='http://localhost/Scripts/test.pl'>
                <p>Operand1: <input type='text' name='A'></p>
                <p>Operand2: <input type='text' name='B'></p>
                <p>Operation:<br>
                <select name='op'>
                        <option value='+'>+</option>
                        <option value='-'>-</option>
                        <option value='*'>*</option>
                        <option value='/'>/</option>
                <select></p>
                <input type='submit' value='Calculate!'>
        </from>
</body>
</html>
```

Перевірте роботу сценарію.

Частина 2. Мова PHP.

1. Підготуйте текстовий файл з розширенням *PHP* і розмістіть його в директорії відповідно до конфігураційними налаштуванням інтерпретатора PHP (параметр *doc_root*).

Після запуску сценарію ви побачите сторінку конфігурації, в якій ви побачите значення *змінних ядра* PHP, встановлені бібліотеки функцій, значення *змінних оточення* і *глобальних змінних* PHP.

Даний сценарій можна використовувати для перевірки списку переданих даних від клієнта, якщо URL сценарію використовувати у формі (атрибут *ACTION*) або викликати його безпосередньо у веб-браузері.

2. Підготуйте файл з текстом *сценарію 6* і HTML сторінку з формою для перевірки його роботи.

Перевірте роботу сценарію для методів *GET* і *POST*.

3. Підготуйте файл з текстом *сценарію 7* і відповідну HTML сторінку з формою, як в аналогічному прикладі мовою Perl.

4. Підготуйте файл з текстом *сценарію 8*. Для першого аргументу функції *fopen* вкажіть ім'я цього файлу.

Перевірте роботу сценарію.

5. Підготуйте файл з текстом *сценарію 9*. Для виконання сценарію необхідно правильно налаштувати права доступу для веб-сервера до директорії, в якій буде зберігатися файл (має бути дозволений доступ на запис). З міркувань безпеки рекомендується для запису створювати окрему директорію.

Запустіть сценарій. Переконайтеся, що був створений файл *rand.dat* і перегляньте його вміст.

Контрольні завдання

1. Витяг списку слів з тексту (сценарій на мові Perl).

1) Підготуйте веб-сторінку з формою, що містить поле для введення тексту і кнопку типу *submit*. Для атрибута *Action* у формі вкажіть як значення *URL* perl-сценарію, наприклад *http://localhost/Scripts/wcount.pl.*

2) Підготуйте сценарій на мові Perl, який витягує з тексту, отриманого від клієнтського додатка, список слів за допомогою функції *split*. Як перший аргумент при виклику функції необхідно вказати регулярний вираз, що задає список роздільників слів. Також сценарій повинен показати загальне число знайдених слів.

2. Витяг списку слів з тексту (сценарій на мові PHP).

1) Змініть веб-сторінку з формою, розроблену в попередньому завданні: для атрибута *Action* у формі вкажіть як значення *URL* PHP-сценарію, наприклад *http://localhost/wcount.php.*

2) Підготуйте сценарій на мові PHP, який витягує з тексту, отриманого від клієнтського додатка, список слів за допомогою функції *preg_split*, який має наступний синтаксис:

array **preg_split**(string *pattern*, string *subject*)

тобто повертає масив, що складається з підрядків заданого рядка *subject*, який розбитий на межі, відповідним шаблоном *pattern*. Шаблон описується за допомогою відповідного регулярного виразу. Також сценарій повинен показати загальне число знайдених слів.

Лабораторна робота №6.

Тема «Знайомство з середовищем розробки Microsoft Visual Studio.NET. Структура програми на C #. Основи мови C #.»

Мета роботи:

1. Знайомство з середовищем розробки додатків Microsoft Visual Studio.NET.
2. Знайомство зі структурою програми мовою C # (для консольного застосування).

Теоретичний матеріал

1. Загальні відомості.

Microsoft Visual Studio.NET – це інтегроване середовище розробки для створення, документування, запуску та налагодження програм, написаних на мовах .NET.

Це середовище розробки є відкритим мовним середовищем. Поряд з мовами програмування, що з самого спочатку включені в середовище – C++, C #, *J#*, *Visual Basic*, – в неї можуть додаватися будь-які мови програмування, компілятори яких створюються сторонніми розробниками. Необхідною умовою для включення мов в середовище **Visual Studio.NET** є використання єдиного каркаса - платформи *Framework.Net*.

Платформа *Framework.NET* дозволяє:

- легко використовувати компоненти, які були розроблені на різних мовах;
- розробляти єдиний додаток з декількох частин на різних мовах.

Для розробки додатків в Visual Studio.NET використовуються *проекти*.

Проект (Project) – це основна одиниця, з якою має справу розробник. Спочатку він повинен вибрати тип проекту, після чого Visual Studio створює каркас проекту відповідно до обраного типу.

Проект складається з класів, зібраних в одному або декількох *просторах імен*. *Простори імен* (Namespaces) дозволяють структурувати проекти, що містять велику кількість класів, об'єднуючи в одну групу близькі класи.

Кілька проектів можуть об'єднуватися в *рішення* (Solution), яке також може включати ресурси, необхідні цим проектам.

З точки зору розробника кінцевим результатом його роботи, одержуваних після компіляції вихідного програмного коду, є *рішення*, а з точки зору CLR (Common Language Runtime – загальномовного середовища виконання) – *збірка* (assembly), що містить PE файл, тобто модуль у форматі виконуваного файлу PE (Portable Executable) для 32-розрядної ОС Windows або DLL (Dynamic Link Library) файл.

Visual Studio.Net пропонує велику різноманітність можливих типів проектів. Для ознайомлення з основами мови C# в основному будуть використані консольні додатки без залучення звичних в Windows можливостей для користувача інтерфейсу.

У C# немає глобальних функцій, тому за замовчуванням при створенні консольного застосування оголошується клас *Program*, який містить функцію *static Main()*, що служить початковою точкою виконання програми. Функція *Main* може бути оголошена без параметрів або з параметром, що представляє собою масив рядків.

Також програма на C# може містити *коментарі* і *атрибути*.

У C# використовуються однорядкові і багаторядкові *коментарі*. *Однорядковий* коментар починається з пари символів «//», і весь текст до кінця рядка є коментарем. *Багаторядковий* коментар починається з пари символів /* і закінчується */.

Атрибут – засіб додавання декларативною інформації до елементів програмного коду. Призначення атрибутів: організація взаємодії між програмними модулями, додаткова інформація про умови виконання коду, управління серіалізацією (тобто правила збереження інформації), налагодження, та ін. Атрибути можуть бути як стандартними, так і заданими користувачем. Стандартні атрибути використовуються CLR і впливають на те, як виконуватиметься проект. Атрибут розміщується в дужках «*[]*».

2. Типи даних C#.

Стандарт мови C++ включає наступний набір фундаментальних типів.

- Логічний тип (*bool*).
- Символьний тип (*char*).
- Цілі типи. Вони можуть відрізнятися розміром: *short*, *int*, *long*, а також можуть бути знаковими (*signed*) або без знаковими (*unsigned*).
- Типи з плаваючою крапкою. Вони також можуть відрізнятися розмірами: *float*, *double* і *long double*.

Тип *void* вказує на відсутність інформації.

До *конструйованих* типів відносяться наступні:

- Покажчики (наприклад, *char**).
- Посилання (наприклад, *char&*).
- Масиви (наприклад, *char[]*).

Також мова дозволяє розробнику конструювати власні типи:

- Перелічуваних типи (*enum*).
- Структури (*struct*).
- Класи.

У мові C# всі типи можна розглядати і під іншим ракурсом, розділивши їх на чотири категорії:

- Типи-значення (*value*).
- Посилальні (*reference*).
- Покажчики (*pointer*).
- Тип *void*.

Для посилального типу значення задає посилання на область пам'яті в «купі» (heap), де розташований відповідний об'єкт. Для типу-значення значенням є власне дані, а пам'ять для них виділяється в стек.

Логічний, арифметичний, структури, перерахування відносяться *типам-значень*. Масиви, рядки і класи відносяться до *посилальним* типам.

І посилальні, і звичайні типи є похідними від базового класу *object*. У тих випадках, коли звичайний тип повинен вести себе як об'єкт, створюється оболонка (wrapper), яку можна розглядати як контрольний об'єкт, поміщений в купу, і в неї копіюється значення змінної звичайного типу. Оболонка автоматично позначається таким чином, що система знає, яке значення вона містить. Цей процес називається *упаковкою* (boxing), а зворотний процес – *розпакуванням* (unboxing).

Упаковка відбувається автоматично, для цього потрібно лише привласнити значення звичайного типу змінної типу *object*. Упаковка та розпакування дозволяють обробляти будь-який тип як об'єкт. Наприклад, у виразі

```
7.ToString();
```

ціле число 7 упаковується шляхом виклику функції *Int32.ToString()*.

Масиви в C# можуть бути *багатомірними* (multidimensional) або *невирівняними* (jagged). Більш складні структури даних такі, як *стек* і *хеш-таблиця* визначені в просторі імен *System.Collections*.

3. Вирази та оператори C#.

Вираження будуються з операндів – констант, змінних, функцій, – об'єднаних знаками операцій і дужками. При обчисленні виразу визначається його значення і тип.

У таблиці нижче наведений список операцій C#.

Категорія операції	Операції
Арифметичні	+ – * / %
Логічні (*boolean* і побітові)	& | ^ ! ~ && ||
Рядкові	+
Інкремент і декремент	++ --
Здвиг	>> <<
Порівняння	== != < > <= >=
Присвоєння	= += –= *= /= %= &= |= ^= <<= >>=
Звернення до члену класу	.
Індексація	[]
Приведення типа (Cast)	()
Умова	?:
Створення об'єкту	new()
Інформація про тип	is sizeof typeof
Управління винятками	checked unchecked
Непрямі і адресації	* –> [] &

Ім'я та *тип* змінної задаються при її оголошенні і залишаються незмінними протягом усього часу її життя. Особливістю мови C# є вимога *обов'язкової ініціалізації змінної до початку її використання*. Спроба використати

неініціалізовану змінну призводить до помилок, що виявляється ще на етапі компіляції.

По використовуваних виразам і операторам, C# схожий на C++. Так в програмах на C# використовуються такі оператори як:

- Оператор присвоювання (=).
- Складовий оператор (***{}***).
- Оператори вибору: ***if-else*** і ***switch***.
- Оператори циклу: ***for***, ***while***, ***оператор***.
- Оператори ***break*** і ***continue***.
- Оператор ***return***.
- Оператор переходу ***goto***

Крім того, введені кілька нових інструкцій.

Наприклад, оператор ***foreach*** дозволяє отримати доступ до всіх елементів масиву чи колекції по черзі, в порядку зростання індексів. Його синтаксис:

foreach (*тип ідентифікатор* **in** *контейнер*) *оператор*

У *C# процедури* і *функції* існують тільки як *методи* деякого *класу*, вони не визначені поза класом. Роль бібліотек процедур і функцій виконують *бібліотеки класів*. Бібліотека класів *Framework Class Library* (FCL), доступна в мові C#, істотно розширює можливості мови.

4. Класи і методи в C#.

У Visual Studio.Net, і в C# зокрема, будь-яка програмна система розглядається як *сукупність класів*, об'єднаних в проекти, простору імен, рішення.

Опис класу має наступний синтаксис:

[Атрибути] [модифікатори] ***class*** *ім'я_класу [:список_батьків] {тіло_класа}*

У тілі класу можуть бути оголошені:

- константи;
- поля;
- конструктори і деструктори;
- методи;
- події;
- делегати;
- класи (структури, інтерфейси, перерахування).

Поля класу синтаксично є звичайними змінними (об'єктами) мови. Їх опис задовольняє звичайним правилам оголошення змінних. Поля характеризують *властивості об'єктів класу*.

Методи класу синтаксично є звичайними процедурами і функціями мови. Методи містять описи операцій, доступних над об'єктами класу. Методи, звані *властивостями* являються спеціальною синтаксичною конструкцією, призначеної для забезпечення ефективної роботи з класами.

Конструктор являє собою спеціальний метод класу, що дозволяє створювати об'єкти класу. Його ім'я повинне збігатися з ім'ям класу. Якщо розробник не визначає конструктор класу, то до класу автоматично додається конструктор за замовчуванням – конструктор без аргументів.

Делегат в C# є опис спеціального випадку класу і задає визначення функціонального типу (класу) даних. Екземплярами класу є функції. Кожен *делегат* описує безліч функцій із заданою сигнатурою. Кожна функція (метод), сигнатура якого збігається з сигнатурою *делегата*, може розглядатися як екземпляр класу, заданого *делегатом*. Синтаксис оголошення *делегата* має наступний вигляд:

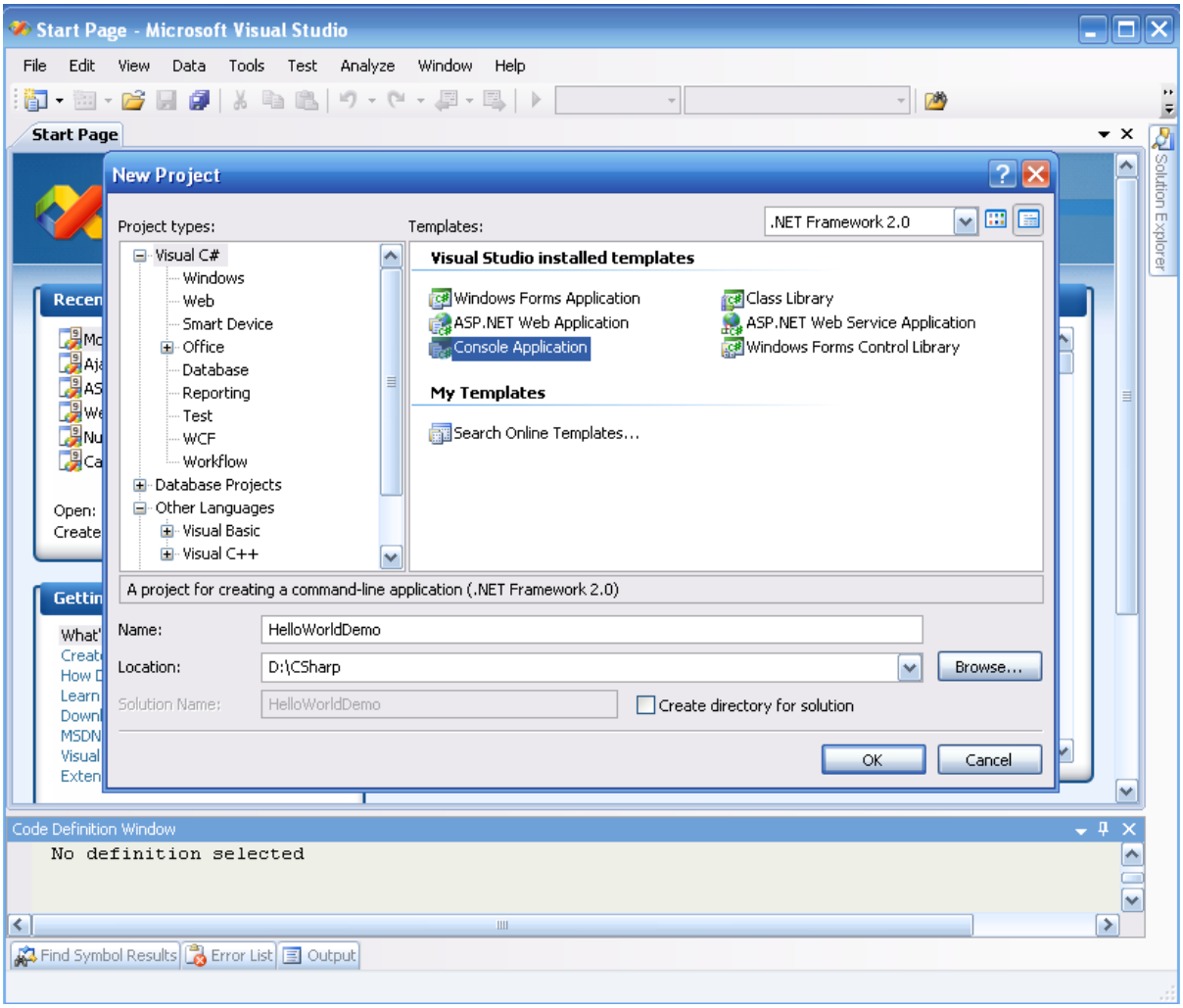
[<Специфікатор доступу>] ***delegate*** *<тип результату> <ім'я класу> (<список аргументів>);*

Порядок виконання роботи

Для практичного вивчення прикладів в даній лабораторній роботі необхідно наявність встановленого пакета *Microsoft Visual Studio* (версії не раніше 2003). Залежно від встановленої версії можливі несуттєві відмінності в інтерфейсі середовища при створенні і роботі з проектом консольного застосування.

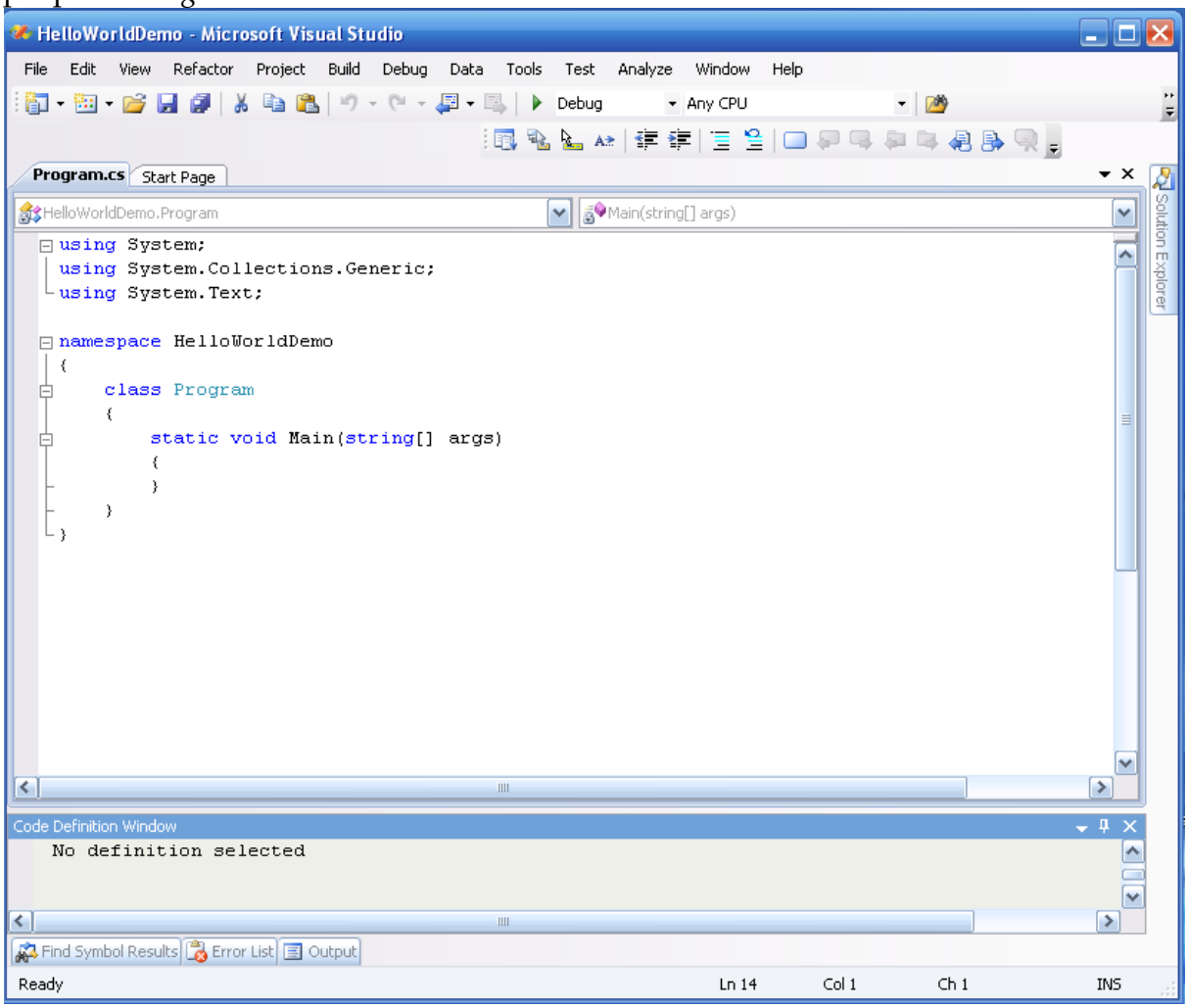
1. Створення проекту консольного застосування.

1) Перший крок при розробці будь-якої програми - створення проекту. Для цього через розділи меню: File> New> Project запускається панель створення нового проекту такого вигляду:



2) Зі списку пропонованих шаблонів необхідно вибрати проект *консольного застосування* (*Console Application*).

3) Після створення проекту має з'явитися таке вікно, що містить текст програми *Program.cs* мовою C#:



У даній програмі оголошення *using System* дає можливість посилатися на класи, які знаходяться в просторі імен *System*, так що їх можна використовувати, не додаючи «*System.*» Перед ім'ям типу.

Простір імен *System* містить багато корисних класів. Одним з них є і клас *Console*, який використовується при створенні консольних додатків.

Клас проекту – *Program* розміщений в просторі імен, що має за замовчуванням те ж саме ім'я (*HelloWorldDemo*), що і *рішення*, що містить в даному прикладі єдиний проект.

Оскільки в C# немає глобальних функцій, тому в даному прикладі оголошується клас *Program*, який містить функцію *static Main()*, що служить початковою точкою виконання програми. Функція *Main* може бути оголошена без параметрів або з параметром, що представляє собою масив рядків. Так як функція *Main* є точкою входу, вона повинна бути *статичною* (static) функцією, тобто вона не пов'язана з конкретним об'єктом класу, в якому оголошена.

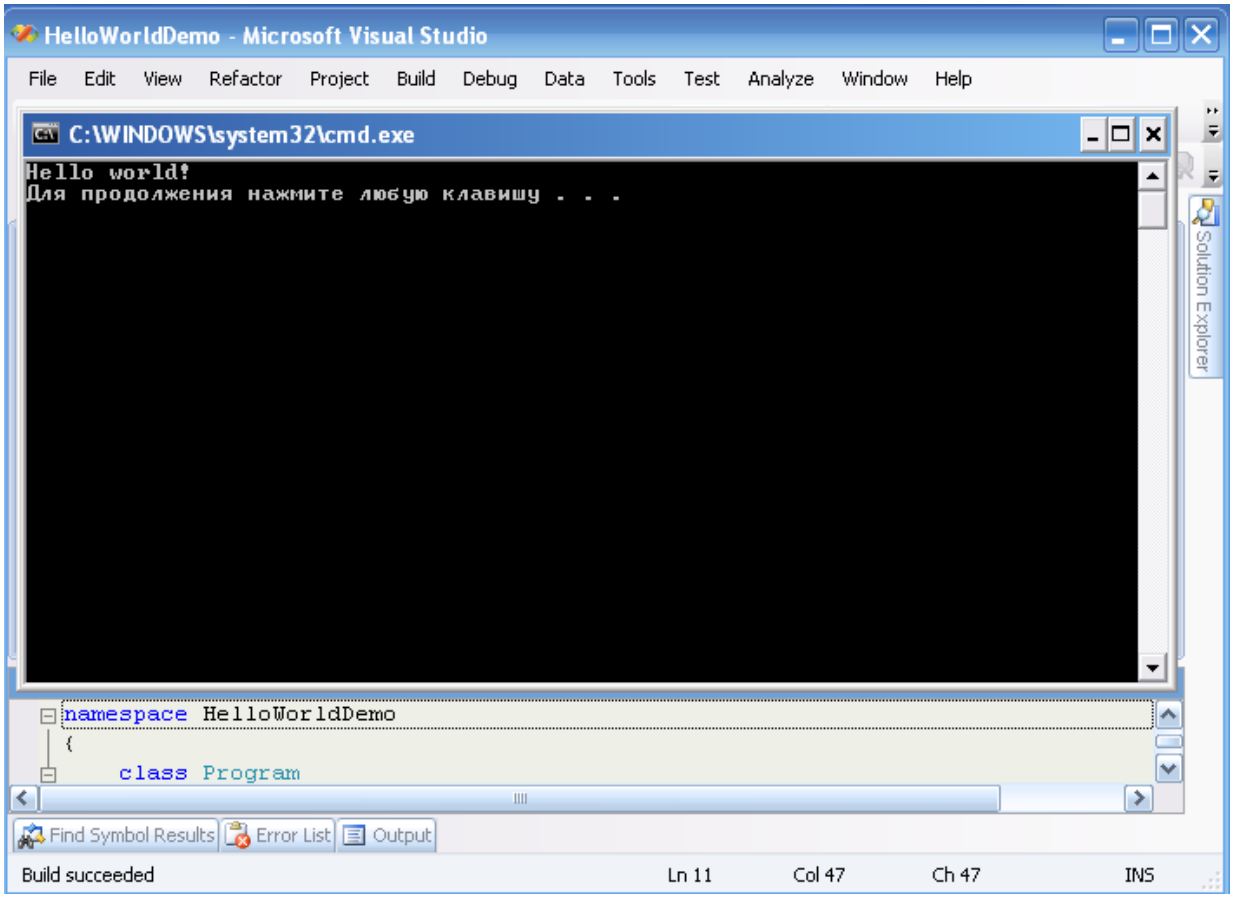
2. Додамо всередині функції *Main* наступний рядок:

```
Console.WriteLine("Hello world!");
```

який використовує метод *WriteLine* класу *Console* для виведення рядка «*Hello world!*».

3. Компіляція проекту. Можна використовувати або меню (Build> Build *Ім'я проекту*) або поєднання клавіш <Shift + F6>. При наявності повідомлень про помилки необхідно їх виправити і скомпілювати проект знову.

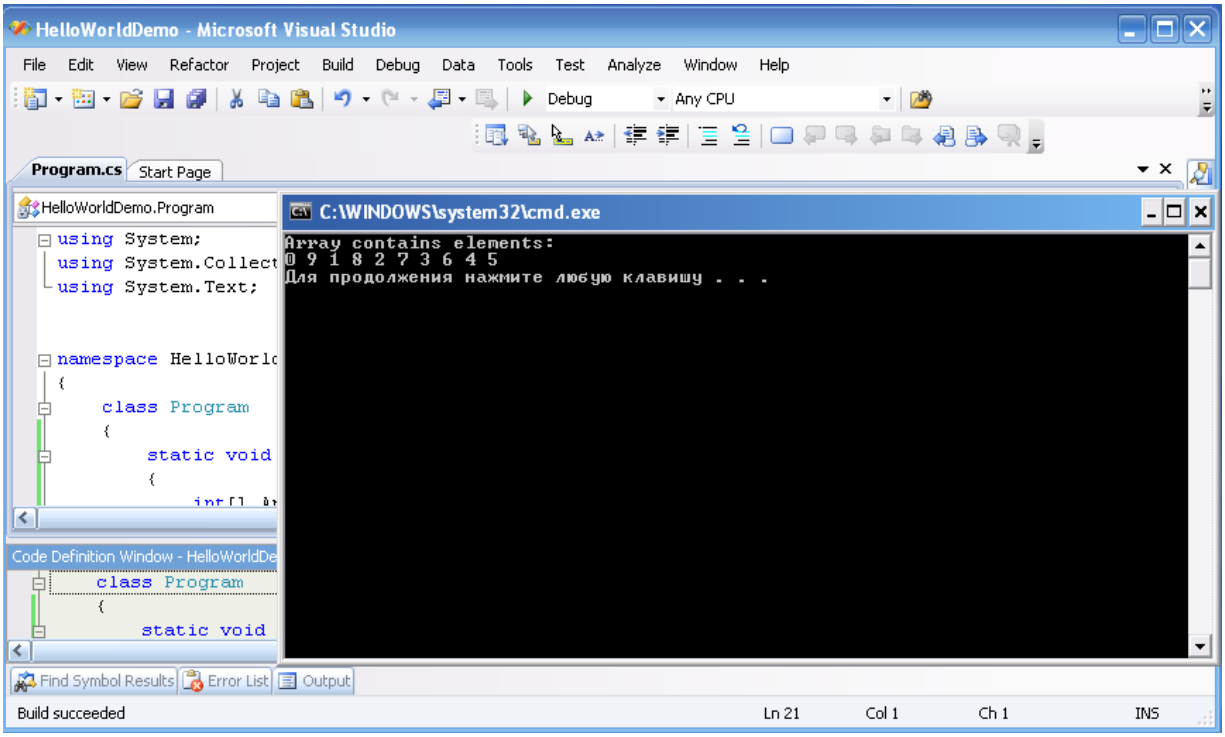
4. Запуск програми. Можна використовувати або меню (Debug> Start Without Debugging) або поєднання клавіш <Ctrl + F5>. В результаті отримаємо наступне консольне вікно:



5. Ускладнимо додаток. Для цього додамо у функцію *Main* наступний код для виведення в консольне вікно змісту масиву цілих чисел:

```
static void Main(string[] args)
{
        int[] ArInt = { 0, 9, 1, 8, 2, 7, 3, 6, 4, 5 };

        Console.WriteLine("Array contains elements:");
        foreach (int item in ArInt)
        {
                Console.Write("{0} ", item);
        }
        Console.WriteLine("");
 }
```

Після повторення кроків 3-4 отримаємо наступне вікно:



Контрольне завдання

Розробка консольного застосування для обчислення коренів квадратного рівняння.

Потрібно розробити програму, що за заданим значенням коефіцієнтів *a, b* і *c* квадратного рівняння (значення вводяться з клавіатури користувачем) обчислює і відображає на екрані корені рівняння.

Для цього додатка потрібно наступні методи:

- string *Console.ReadLine()* – читання рядка символів з вхідного потоку.
- double *Convert.ToDouble*(string) – перетворення рядка символів в число з плаваючою комою подвійної точності.
- double *Math.Sqrt*(double) – витяг квадратного кореня числа.

Лабораторна робота №7.

Тема «Основи мови C#. Робота з масивами і рядками. Інтерфейси і колекції.»

Мета роботи:

1. Продовження знайомства із середовищем розробки додатків Microsoft Visual Studio.NET.
2. Вивчення особливостей роботи з масивами і рядками в C#.
3. Ознайомлення з реалізацією інтерфейсами в C#.

Теоретичний матеріал

1. Масиви

Мова C# надає засоби для роботи з одновимірними *масивами*, *масивами масивів* і *багатовимірними масивами*.

Нижче представлені способи оголошення посилань на масиви різної розмірності і конфігурації:

```
// Оголошення масиву розмірності 3 з елементів типу int.
// Фактично це масив складових, що представляють собою масиви
// розмірності 2 одновимірних масивів елементів типу int.
// Розміри всіх складових масивів одного рівня рівні
// між собою ("прямокутний" масив).

int[,,] ar1;

// Оголошення одновимірного масиву з
// одновимірних масивів елементів, кожен з яких є
// одновимірним масивом елементів типу int.

int[][][] ar2;

// Оголошення одновимірного масиву складових, кожен з яких є
// двовимірним масивом масивів елементів типу int.
// При цьому у всіх складових можуть бути різні розміри

int[][,] ar3;

// Оголошення двовимірного масиву складових, кожна з яких є
// одновимірним масивом елементів типу int.
// При цьому у всіх складових можуть бути різні розміри.

int[,][] ar4;
```

Розглянуті приклади демонструють визначення двох різних категорії масивів:

- прямокутні масиви;
- ламані (jagged) масиви.

Оголошення масиву може бути поєднане з ініціалізацією. При оголошенні з відкладеним ініціалізацією сам *масив* не формується, а створюється тільки посилання на *масив*, що має невизначене значення *Null*. Тому поки *масив* не буде реально створений і його елементи ініціалізовані, використовувати його не можна.

Можливі два варіанти оголошення *масиву* поєднаного з ініціалізацією.

У першому випадку ініціалізація є явною і задається *списком констант*, наприклад:

int[] odd = {1, 3, 5, 7, 9};

У другому випадку створення і ініціалізація масиву реалізується разом з викликом конструктора *масиву*. Наприклад:

int[] even = new int[5];

Всі масиви є нащадками класу *System.Array*, який у свою чергу успадковує ряд інтерфейсів: *ICloneable*, *IList*, *ICollection*, *IEnumerable*, а, отже, реалізує всі їхні методи і властивості. Крім того, клас *Array* має велике число власних методів і властивостей.

Завдяки спадкоємства від класу *System.Array*, для масивів визначені такі вбудовані методи копіювання, пошуку, звернення, сортування. Масиви можна розглядати також як колекції і використовувати цикл *foreach* для перебору його елементів.

Нижче наведені основні властивості і методи класу *System.Array*:

Властивість	**Опис**
IsFixedSize	Повертає *true*, якщо масив є статичним
IsReadOnly	Для всіх масивів повертає значення *false*
IsSynchronized	Повертає *true* або *false*, залежно від того, чи встановлена синхронізація доступу для масиву
Length	Число елементів у масиві
Rank	Розмірність масиву

Метод	**Опис**
BinarySearch	Бінарний пошук елементів
Clear	Виконує ініціалізацію елементів. Числові елементи стають нульовими, а рядкові приймають значення *Null*
Clone	Клонування масиву
Copy	Копіювання частини або всього масиву в інший масив
CopyTo	Копіюються всі елементи одновимірного масиву в інший одновимірний масив, починаючи з заданого індексу
CreateInstance	Створення примірника масиву

GetEnumerator	Повертає інтерфейс *IEnumerator*. Необхідний для використання в циклі **foreach**
GetLength	Повертає число елементів масиву за вказаною вимірюванню
GetLowerBound, GetUpperBound	Повертає нижню і верхню межу за вказаною виміру. Для масивів нижня межа завжди дорівнює нулю.
GetValue etValue	Повертає або встановлює значення елемента масиву із зазначеними індексами.
IndexOf	Індекс першого входження зразка в масив
LastIndexOf	Індекс останнього входження зразка в масив
Reverse	Сортування одновимірного масиву в зворотному порядку
Sort	Сортування масиву

2. Рядки

У мові C # визначено клас *char[]*, і його можна використовувати для представлення рядків постійної довжини. Однак масив *char[]* – це звичайний масив, тому його не можна ініціалізувати рядком символів. У C# не визначене перетворення з класу *char[]* в клас *String*. У *String* є динамічний метод *ToCharArray*, що задає подібне перетворення в *char[]*.

Об'єкти класу String оголошуються з явною або відкладеною ініціалізацією, з явним або неявним викликом конструктора класу. Найчастіше, при оголошенні строкової змінної конструктор явно не викликається, а ініціалізація задається строковою константою. У класі String досить багато конструкторів, які дозволяють сконструювати рядок з:

- символу, повторюваного вказане число разів;
- масиву символів *char[]*;
- частини масиву символів.

Методи класу String дозволяють виконувати вставку, видалення, заміну, пошук входження підрядка в рядок. Клас String успадковує методи класу *Object*, а також методи чотирьох інтерфейсів: *IComparable*, *ICloneable*, *IConvertible*, *IEnumerable*.

Деякі з членів класу наведені в таблиці:

Метод	Опис
Length	Ця властивість повертає довжину зазначеного рядка
Insert	Вставляє підрядок в задану позицію
Remove	Видаляє подстроку в заданій позиції
Replace	Замінює підрядок в заданій позиції на нову підрядок
Substring	Виділяє подстроку в заданій позиції
IndexOf, IndexOfAny, LastIndexOf, LastIndexOfAny	Визначаються індекси першого і останнього входження заданого підрядка або будь-якого символу із заданого набору
StartsWith, EndsWith	Повертається *true* або *false*, залежно від того, починається або закінчується рядок заданої підрядком

PadLeft, PadRight	Виконує заповнення потрібним числом прогалин на початку і в кінці рядка
Trim, TrimStart, TrimEnd	Видаляються пробіли на початку і в кінці рядка, або тільки з одного кінця
ToCharArray	Перетворення рядка в масив символів

Клас String не дозволяє змінювати існуючі об'єкти. Для цієї мети є інший клас – клас *StringBuilder*, який дозволяє виправити даний недолік. Цей клас належить до змінюваних класів і його можна знайти в просторі імен *System.Text*.

Об'єкти цього класу оголошуються з явним викликом конструктора класу. Конструктор класу перевантажений, і поряд з конструктором без параметрів, що створює порожній рядок, є набір конструкторів, яким можна передати дві групи параметрів:

- Перша група дозволяє задати *рядок* або підрядок, значенням якого буде ініціалізування створювання об'єкту класу *StringBuilder*.
- Друга група параметрів дозволяє задати розмір об'єкту, тобто обсяг пам'яті, що відводиться для екземпляру класу *StringBuilder*.

Приклади конструкторів:

- *public StringBuilder (string str, int size)*. Параметр *str* задає рядок для ініціалізації, *size* - розмір об'єкту;
- *public StringBuilder (int cursize, int maxsize)*. Параметри *cursize* і *maxsize* задають початковий і максимальний розмір об'єкту;
- *public StringBuilder (string str, int start, int len, int size)*. Параметри *str*, *start*, *len* задають рядок ініціалізації, *size* – розмір об'єкту.

Над рядками цього класу визначені (також як і для рядків класу *String*) операції:

- присвоювання (**=**);
- дві операції перевірки еквівалентності (**==**) і (**! =**);
- взяття індексу (**[]**).

Операція конкатенації (**+**) не визначена над рядками класу *StringBuilder*, її роль грає метод *Append*, який додає новий рядок в кінець існуючого.

3. Інтерфейси

Інтерфейс являє собою повністю абстрактний клас, всі методи якого абстрактні. Проте методи *інтерфейсу* оголошуються без вказівки модифікатора доступу, і клас, що успадковує *інтерфейс*, зобов'язаний повністю реалізувати всі методи *інтерфейсу*. У цьому – відмінність від класу, що посяде абстрактний клас, де нащадок може реалізувати лише деякі методи батьківського абстрактного класу, залишаючись абстрактним класом.

Інтерфейс дозволяє описувати деякі бажані властивості, якими можуть володіти об'єкти різних класів.

Серед інтерфейсів, вбудованих в бібліотеку базових класів .NET, можна особливо виділити такі як:

- *IEnumerable* (для роботи з наборами об'єктів, в т. ч. з використанням оператора *foreach*).
- *IClonable* (копіювання об'єктів).

- *IComparable* (для порівняння і сортування об'єктів).

Простір імен System.Collections, призначений для роботи з наборами об'єктів, підтримує інтерфейси:

- *ICollection* (визначає загальні характеристики класу набору елементів).
- *IComparer, IDictionary* (дозволяє представляти зміст обєкту у вигляді пар ім'я-значення).
- *IDictionaryEnumerator* (нумерація змісту об'єкту, що підтримує IDictionary).
- *IEnumerable, IEnumerator*.
- *IHashCodeProvider* (повертає хеш-код за допомогою обраного алгоритму хешування).
- *IList* (забезпечує методи додавання, видалення та індексування елементів у списку об'єктів).

Порядок виконання лабораторної роботи

Для вивчення приводяться в лабораторній роботі приклади, які необхідно створити в *Microsoft Visual Studio* проект консольного застосування і описати клас *Program*, використовуючи код, що приводиться в прикладах.

1. Робота з масивами.

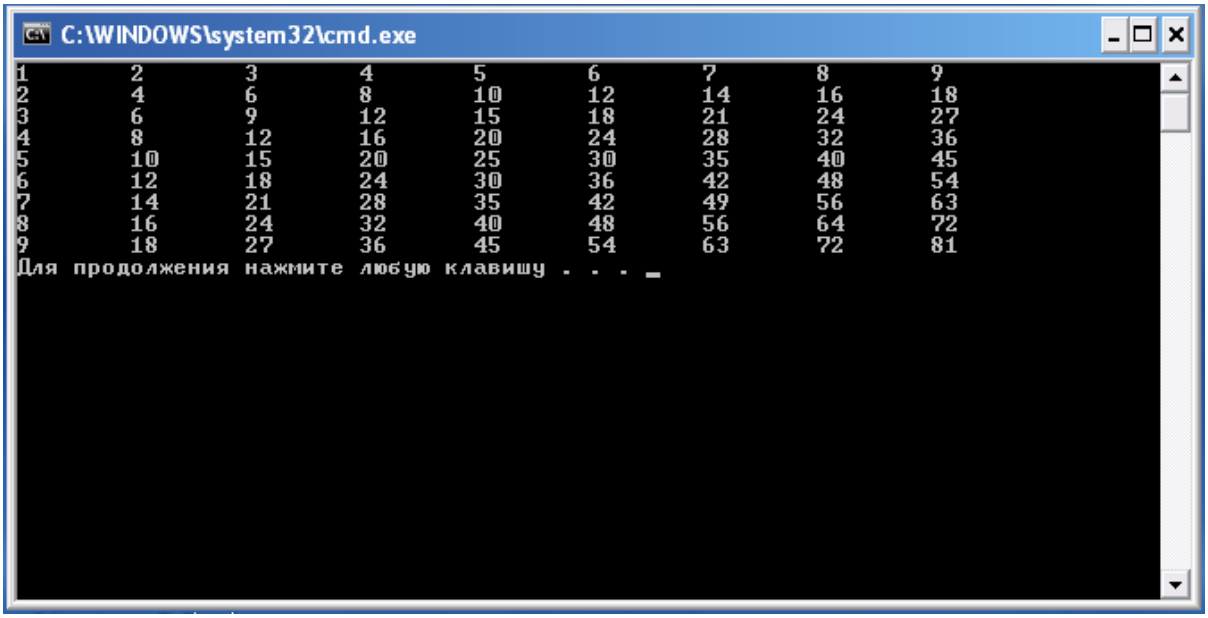
Наступні дві програми демонструють, яким чином виконується ініціалізація і робота з прямокутними і ламаними масивами в C#.

1) Прямокутний масив.

```
using System;
using System.Collections.Generic;
using System.Text;

namespace HelloWorldDemo
{
class Program
{
        static void Main(string[] args)
        {
                // Прямокутний багатовимірний масив
                int[,]multMatr;
                multMatr = new int[10,10];

                // заповнення масиву 9 на 9:
                for (int i = 1; i <= 9; i++)
                        for (int j = 1; j < 9; j++)
                                multMatr[i, j] = i * j;

                // Виведення елементів багатовимірного масиву
                for (int i = 1; i <= 9; i++)
                {
                        for (int j = 1; j <= 9; j++)
```

```
                {
                        Console.Write(multMatr [i, j] + "\t");
                }
                Console.WriteLine();
            }
        }
}
```

В результаті виконання програми отримаємо таблицю множення такого вигляду:

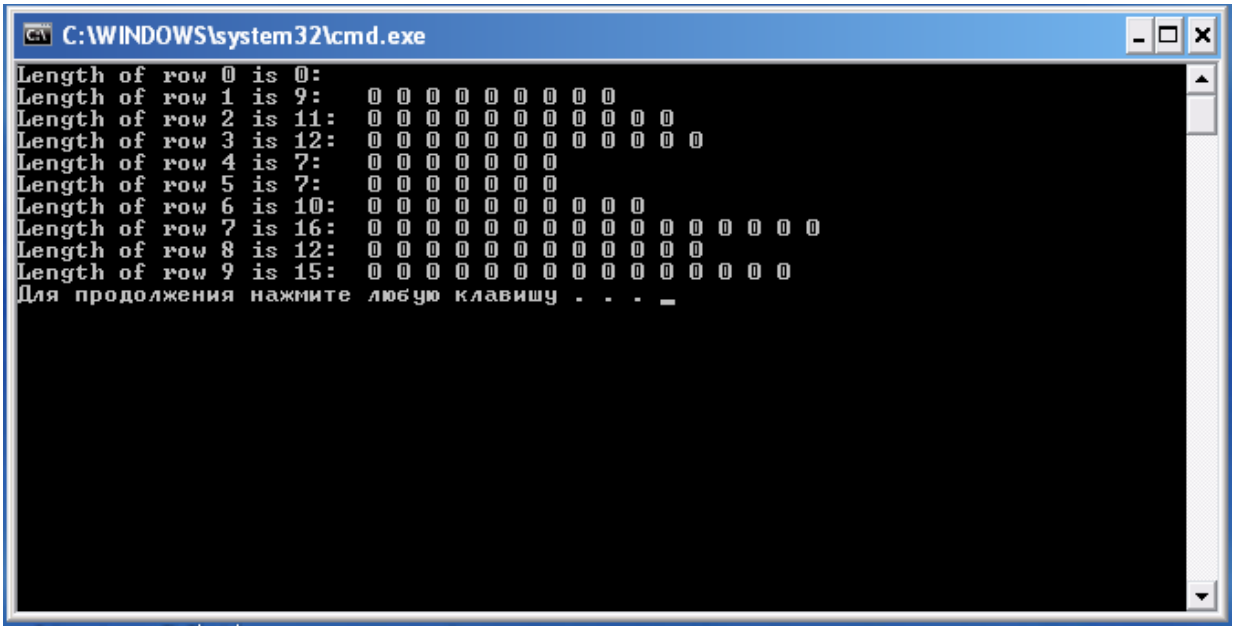


2) Ламаний масив.

```
using System;
using System.Collections.Generic;
using System.Text;

namespace HelloWorldDemo
{
        class Program
        {
          static void Main(string[] args)
          {
                // Ламаний багатовимірний масив з п'яти внутрішніх масивів
                різного розміру
                int[][] JagArr = new int[10][];

                // Ініціалізація генератора випадкових чисел
                Random rand = new Random();

                // Динамічне створення ламаного масив
```

```
            for (int i = 0; i < JagArr.Length; i++)
            {
                JagArr[i] = new int[i + rand.Next(10)];
            }

            // Виведення рядків на консоль
            // Кожен елемент за замовчуванням має значення, рівне
            for (int i = 0; i < 10; i++)
            {
                // Властивість Length масиву повертає його розмір
                Console.Write("Length of row {0} is {1}:\t", i, JagArr[i].Length);
                for (int j = 0; j < JagArr[i].Length; j++)
                {
                    Console.Write(JagArr[i][j] + " ");
                }
                Console.WriteLine();
            }
        }
    }

}
```

В результаті виконання програми отримаємо наступну таблицю:



2. Властивості і методи класу *Array*.

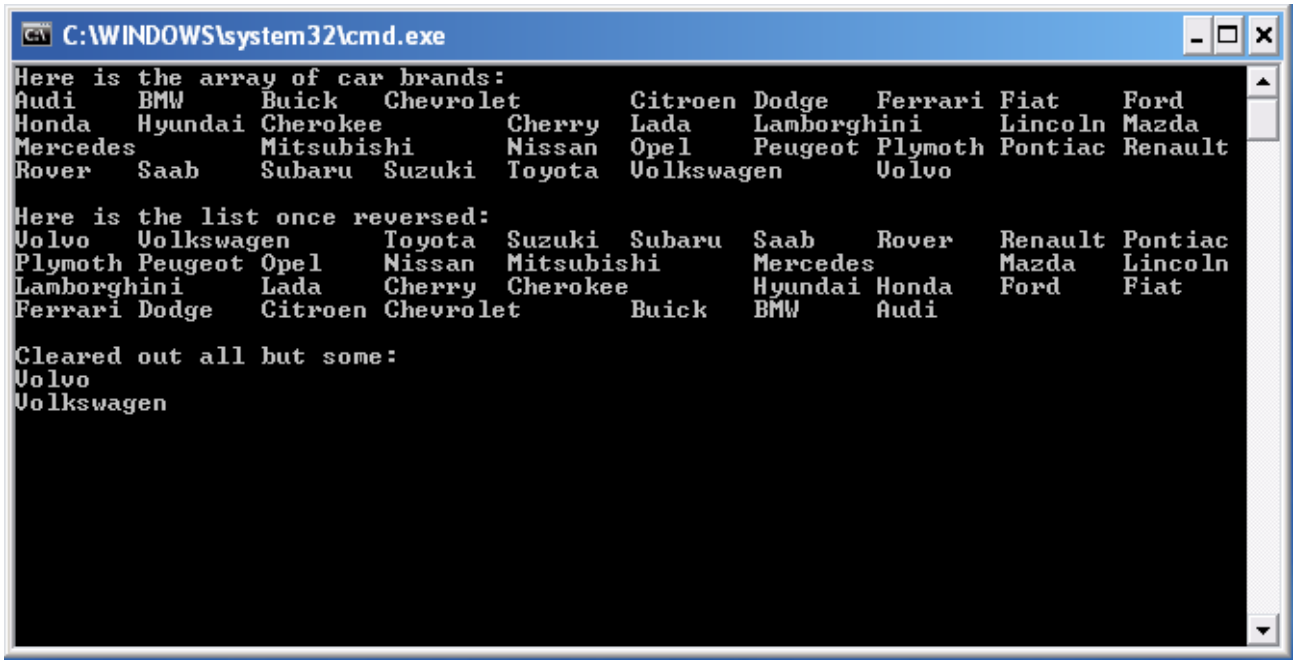
Наступний приклад демонструє використання деяких з властивостей і методів класу *Array*:

```
using System;
using System.Collections.Generic;
using System.Text;
```

```
namespace HelloWorldDemo
{
   class Program
   {
      static void Main(string[] args)
      {
    // Масив символьних рядків
         string[] Brands = new string[]
                                 {"Audi",
            "BMW",
            "Buick",
            "Chevrolet",
            "Citroen",
            "Dodge",
            "Ferrari",
            "Fiat",
            "Ford",
            "Honda",
            "Hyundai",
            "Cherokee",
            "Cherry",
            "Lada",
            "Lamborghini",
            "Lincoln",
            "Mazda",
            "Mercedes",
            "Mitsubishi",
            "Nissan",
            "Opel",
            "Peugeot",
            "Plymoth",
            "Pontiac",
            "Renault",
            "Rover",
            "Saab",
            "Subaru",
            "Suzuki",
            "Toyota",
            "Volkswagen",
            "Volvo"};

        // Виведення марок автомобілів відповідно до порядку елементів у масиві

                        Console.WriteLine("Here is the array of car brands:");

                        for (int i = 0; i < Brands.Length; i++)
```

```
                    Console.Write(Brands[i] + "\t");

            Console.Write("\n\n");

        // Сортування елементів у зворотному порядку

                    Array.Reverse(Brands);

        // Виведення автомарок
                    Console.WriteLine("Here is the list once reversed:");
                    for (int i = 0; i < Brands.Length; i++)
                        Console.Write(Brands[i] + "\t");

            Console.Write("\n\n");

        // Залишаються тільки перший і останній

                    Console.WriteLine("Only two remain: ");
                    Array.Clear(Brands, 2, Brands.Length-2);

                    for (int i = 0; i < Brands.Length; i++)
                    {
                        Console.Write(Brands[i] + "\t\n");
                    }
            }

    }
}
```

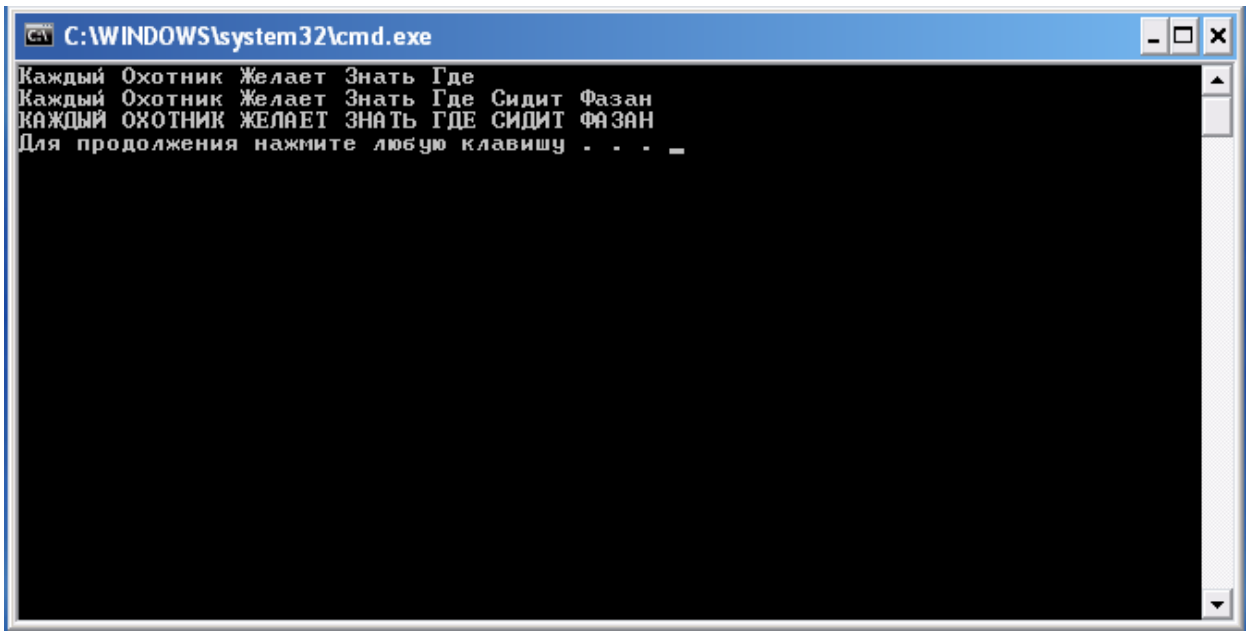
Результат роботи програми:



3. Використання класу *StringBuilder*.

```
using System;
using System.Collections.Generic;
using System.Text;

namespace HelloWorldDemo
{
        class Program
        {
                static void Main(string[] args)
                {
                        StringBuilder WordList = new StringBuilder("Каждый Охотник ");

                        WordList.Append("Желает Знать Где ");
                        Console.WriteLine(WordList);

                        WordList.Append("Сидит Фазан");
                        Console.WriteLine(WordList);

                        // Зробити всі букви прописними
                        string Spectrum = WordList.ToString().ToUpper();
                        Console.WriteLine(Spectrum);
                }
        }
}
```

Результат роботи програми:



4. Використання класу *ArrayList*.

Використання класу *ArrayList* з простору імен *System.Collections* дозволяє ефективно реалізувати роботу з масивами об'єктів, оскільки багато можливостей,

які необхідні для цього, реалізовані спочатку, зокрема методи вставки, видалення і нумерації елементів.

Для використання можливостей *ArrayList* використовується не звичайне спадкування, а модель включення у вигляді делегування викликів на виконання різних дій класу похідному від *ArrayList*:

```
public class NBooks : IEnumerable
{
// nbList — внутрішній клас, який буде робити всю роботу
        private ArrayList nbList;

// Створюємо об'єкт класу nbList за допомогою конструктора NBooks
        public NBooks() {nbList = new ArrayList();}

// Реалізуємо потрібні нам методи для прийому викликів ззовні і передачі їх
nbList

        // Метод для вставки об'єкта NBook
        public void AddNBook(NBook nb)
        { nbList.Add(nb); }

// Метод для видалення об'єкту NBook
        public void RemoveNBook(int nbToRemove)
        { nbList.RemoveAt(nbToRemove); }

// Властивість, що повертає кількість об'єктів NBook
        public int NBookCount
        { get { return nbList.Count; } }

// Метод для очищення об'єкту - видалення всіх об'єктів NBook
        public void ClearAllNBooks()
        { nbList.Clear(); }

// Метод, який відповідає на питання - чи є вже в наборі такий об'єкт NBook
        public bool NBookIsPresent(NBook c)
        { return nbList.Contains(c); }

// Все, що пов'язано з реалізацією IEnumerator, перенаправляється в nbList
        public IEnumerator GetEnumerator()
        { return nbList.GetEnumerator(); }
}
```

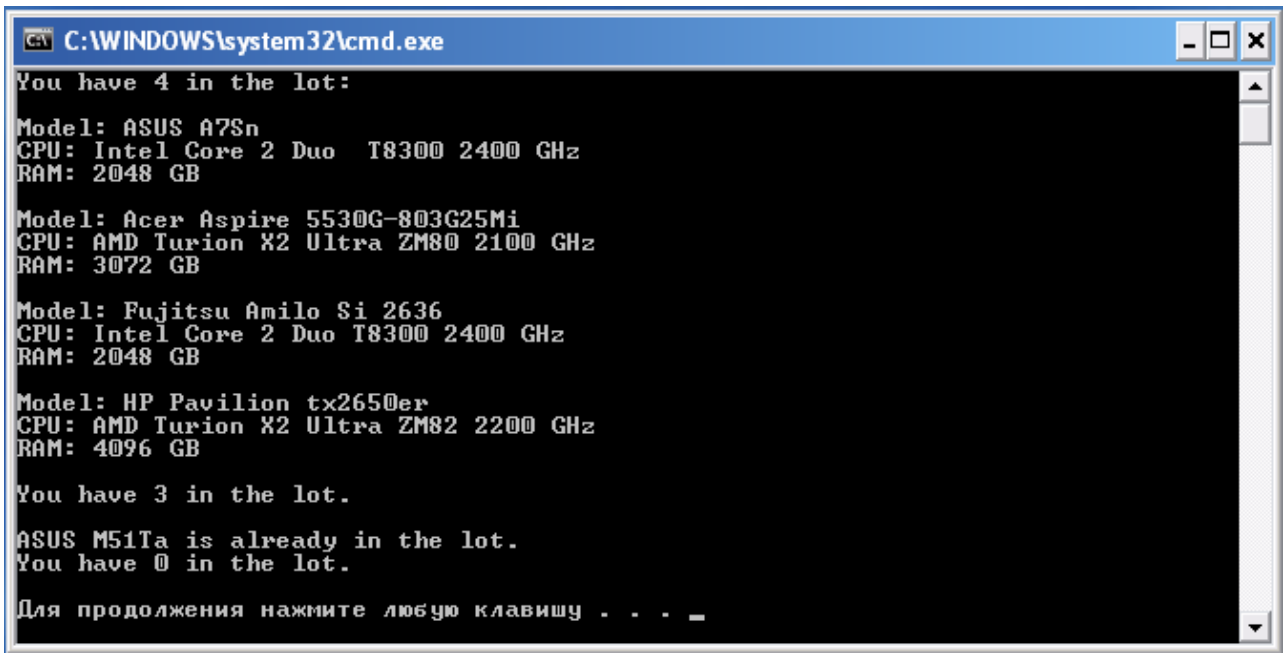
Реалізація класу *NBook* і код програми, що використовує клас *NBooks* представлені нижче:

```
using System;
using System.Collections.Generic;
```

```
using System.Collections;
using System.Text;

namespace NBooks
{
  public class NBook
  {

// конструктор класу NBook

  public string Model;
  public string CPU_model;
  public int CPU_clock;
  public int RAM_size;

  public NBook(string mname, string CPU, int Clock, int RAM)
  {
    Model = mname;
    CPU_model = CPU;
    CPU_clock = Clock;
    RAM_size = RAM;
  }
}

  class Program
  {
    static void Main(string[] args)
    {
      NBooks nbLot = new NBooks();

// Створення списку об'єктів NBook

          nbLot.AddNBook( new NBook("ASUS A7Sn","Intel Core 2 Duo  T8300", 2400,
2048));
          nbLot.AddNBook( new NBook("Acer Aspire 5530G-803G25Mi","AMD Turion
X2 Ultra ZM80", 2100, 3072));
          nbLot.AddNBook( new NBook("Fujitsu Amilo Si 2636", "Intel Core 2 Duo
T8300", 2400, 2048));
          nbLot.AddNBook( new NBook("HP Pavilion tx2650er", "AMD Turion X2 Ultra
ZM82",  2200, 4096));

// Виводимо інформацію про кожен об'єкт за допомогою конструкції foreach

          Console.WriteLine("You have {0} in the lot: \n", nbLot.NBookCount);
          foreach (NBook nb in nbLot)
          {
            Console.WriteLine("Model: {0}", nb.Model);
```

```
            Console.Write("CPU: {0}", nb.CPU_model);
            Console.WriteLine(" {0} GHz", nb.CPU_clock);
            Console.WriteLine("RAM: {0} GB\n", nb.RAM_size);
          }

// Видаляємо один з ноутбуків

          nbLot.RemoveNBook(3);
          Console.WriteLine("You have {0} in the lot.\n", nbLot.NBookCount);

// Додаємо ще один ноутбук і перевіряємо його наявність в наборі

          NBook temp = new NBook("ASUS M51Ta", "AMD Turion™ X2 Ultra ZM84",
2300, 4096);
          nbLot.AddNBook(temp);

          if(nbLot.NBookIsPresent(temp))
                Console.WriteLine(temp.Model + " is already in the lot.");

// Видалити все

          nbLot.ClearAllNBooks();
          Console.WriteLine("You have {0} in the lot.\n", nbLot.NBookCount);
    }
  }
}
```

Результат виконання програми:



Контрольне завдання

Необхідно розробити консольний додаток для введення з клавіатури масиву рядків і пошуку серед них рядків, що містять заданий строковий фрагмент.

Для пошуку потрібно використання методу IndexOf (string *findThisString*) для строкових елементів масиву. Метод повертає позицію початку пошукового підрядка від початку рядка, або значення *– 1* при відсутності відповідності.

Лабораторна робота №8.

Тема «Основи розробки веб-додатків за допомогою ASP.NET.»

Мета роботи:

1. Ознайомлення з основними етапами розробки веб-додатків на основі ASP.NET в середовищі *Microsoft Visual Studio.NET*.
2. Вивчення структури проекту *ASP.NET Web Application*.

Теоретичний матеріал

ASP.NET файл є текстовим файлом і може містити коди HTML, XML і мов сценаріїв. Коди останніх виконуються на веб-сервері. Файл ASP.NET має спеціальне розширення «*.aspx*».

Порядок роботи ASP.NET виглядає наступним чином:

- Коли веб-браузер запрошує файл ASP.NET, веб-сервер IIS перенаправляє запит модулю ASP.NET на сервері.
- Модуль ASP.NET читає файл построчно і виконує, коди сценаріїв, що містяться у файлі.
- Веб-браузеру повертається назад файл ASP.NET, але вже у вигляді звичайного HTML документа.

Будь-яка сторінка ASP.NET представлена класом, похідним від класу *System.Web.UI*, який визначає властивості, методи і події, загальні для всіх сторінок, призначених для обробки середовищем ASP.NET.

Найбільш важливі властивості цього об'єкту наведені в таблиці нижче:

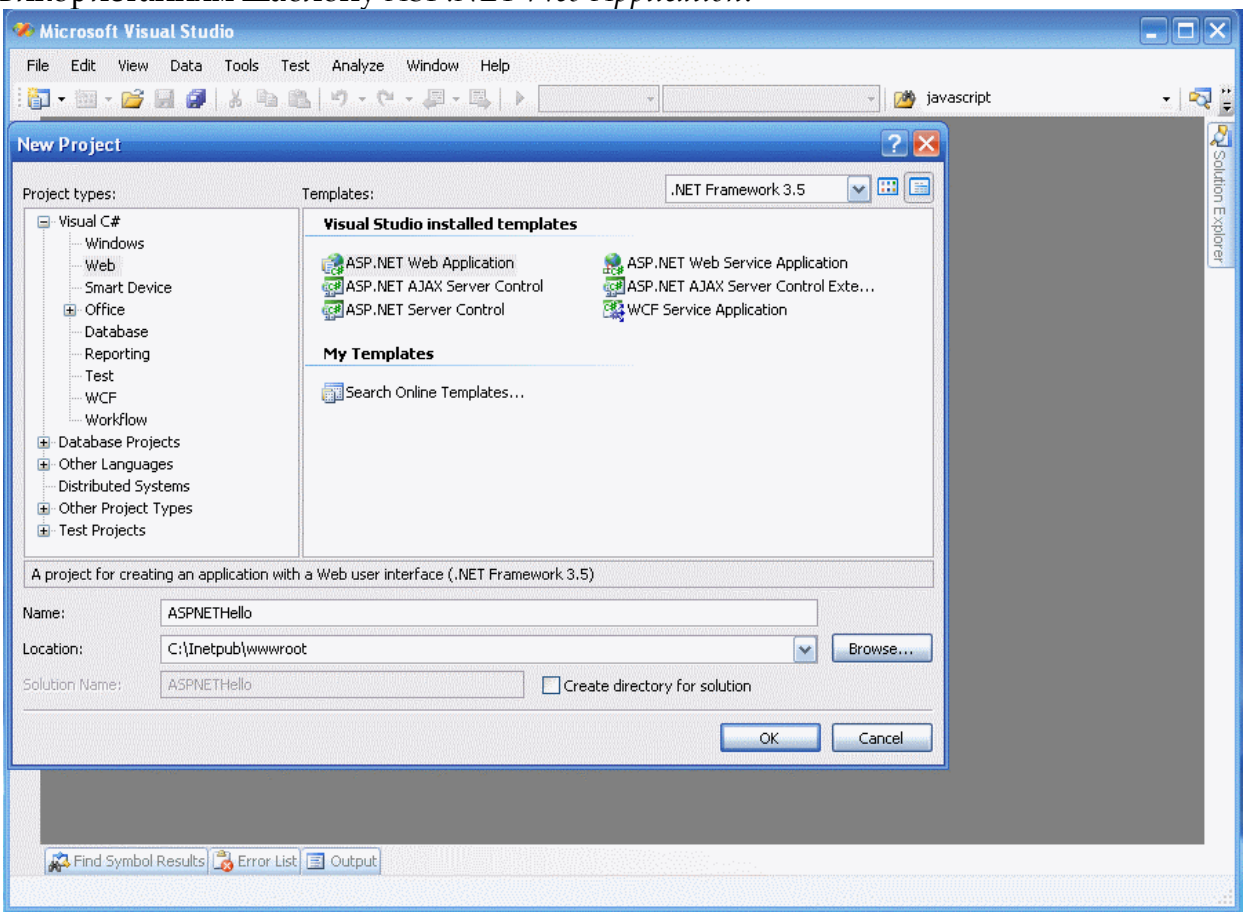
Властивість	Опис
Application	Повертає об'єкт *HttpApplicationState*
Cache	Повертає об'єкт *Cache*, в якому зберігаються дані програми, в т. ч. і самої сторінки
IsPostBack	Повертає значення, що визначає, чи була сторінка завантажена клієнтом вперше раз, або завантажена повторно у відповідь на запит клієнта
Request	Повертає об'єкт *HttpRequest,* використовуваний для отримання інформації про вхідному запиті HTTP
Response	Повертає об'єкт *HttpResponse*, використовувані для формування відповіді сервера клієнту
Server	Повертає об'єкт *HttpServerUtility*
Session	Повертає об'єкт *System.Web.SessionState.HttpSessionState*, за допомогою якого виходить інформація про поточний час HTTP

Така побудова проекту дозволяє зберігати окремо *код подання* для генерації HTML коду (у файлі * *.aspx*) від *програмної логіки* (у файлі * *.aspx.cs*), що в багатьох випадках істотно спрощує розробку складних веб-додатків.

Порядок виконання лабораторної роботи

Для роботи з прикладами, що приводяться в даній лабораторній роботі, потрібно установка середовища розробки Microsoft Visual Studio 2005+ і веб-сервера IIS 5+ (Internet Information Server).

1. Створення нового проекту в середовищі Microsoft Visual Studio з використанням шаблону *ASP.NET Web Application*:

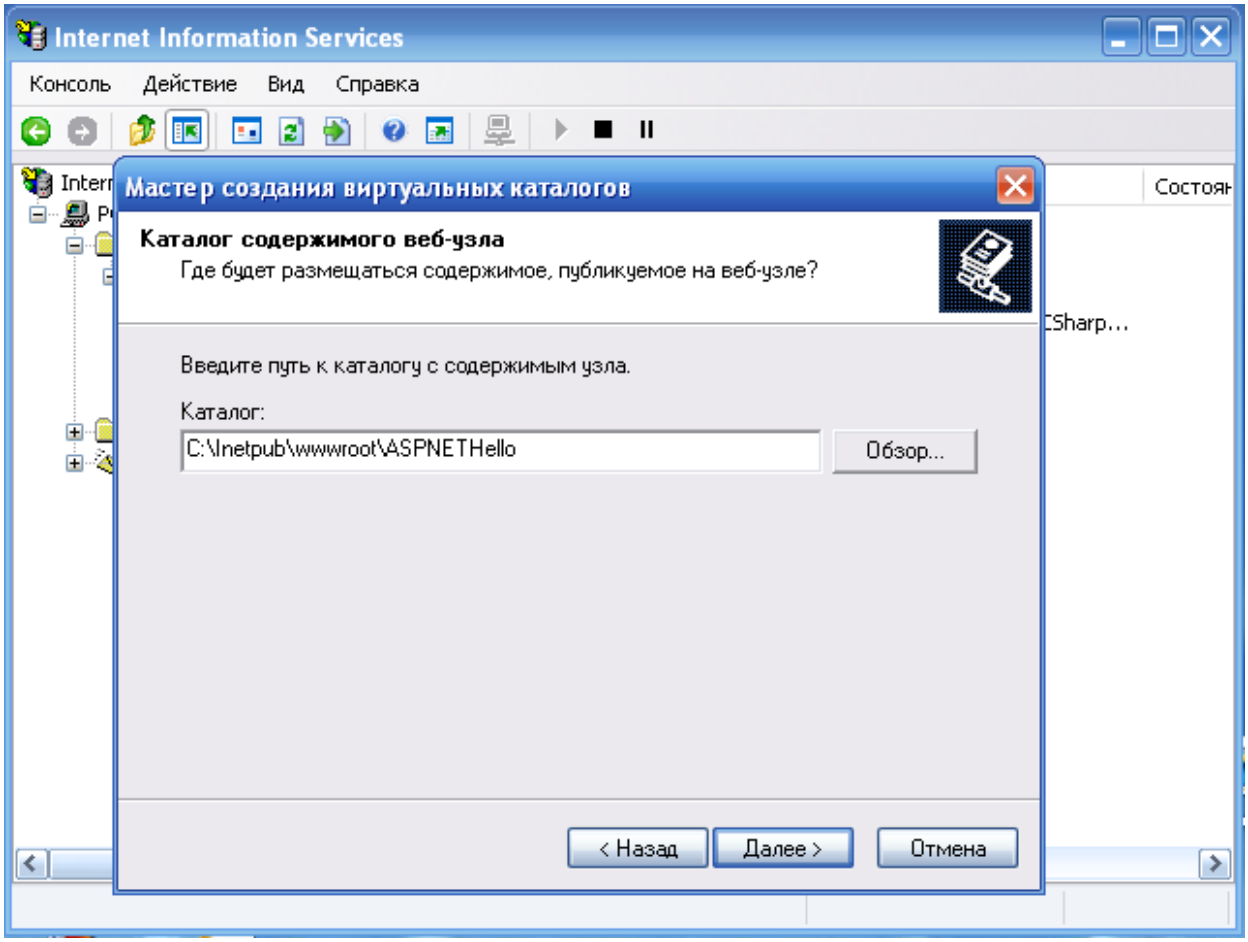


2. Кожне з веб-додатків для IIS повинна розміщуватись у своєму віртуальному каталозі, якому відповідає фізичний каталог на диску.

Для створення віртуального каталогу необхідно:

1) відкрити оснастку Microsoft Internet Information Services (Пуск> Панель управління> Адміністрування> Internet Information Services);

2) розкрити гілку «Веб-сайти» і, перейшовши в «Веб-вузол за замовчуванням», створити новий віртуальний каталог:



3) у властивостях створеного віртуального каталогу вибрати закладку «Документи» і додати в список документів, що використовуються за замовчуванням, документ *Default.aspx*.

Після завершення створення проекту, він міститиме файли *Default.aspx*, *Default.aspx.cs* і *Default.asp.designer.cs*.

Перший з них буде містити приблизно наступний код:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="ASPNETHello._Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
   <title>Untitled Page</title>
</head>
<body>
   <form id="form1" runat="server">
   <div>
   </div>
   </form>
```

```
</body>
</html>
```

З цього коду видно, що, по-перше, для створення коду HTML, що повертається браузеру, буде використаний мову C# (див. Атрибут *Language*). По-друге, код C# міститься в окремому файлі, який буде виконуватися на веб-сервері (див. Атрибут *CodeBehind*). І, нарешті, атрибут *Inherits* вказує на ім'я класу, визначеного в *CodeBehind*.

Важливим нововведенням в ASP.NET є атрибут *runat*="*server*", розміщений в тегі <form>. Він означає, що даний елемент повинен бути оброблений середовищем виконання ASP.NET.

3. Тепер між тегами і </ div> у файлі *Default.aspx* можна вставити код:

```
<h1>Hello!</h1>
<h1>I am</h1>
<%=Request.ServerVariables["HTTP_USER_AGENT"] %>
```

У даному прикладі було використано властивість *Request* об'єкту, похідного від *Page*, для отримання значення змінної оточення *HTTP_USER_AGENT*.

Код C#, який міститься у файлі *Default.aspx.cs* може виглядати приблизно так:

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;

namespace ASPNETHello
{
   public partial class _Default : System.Web.UI.Page
   {
      protected void Page_Load(object sender, EventArgs e)
      {

      }
   }
```

```
}
```

Тут міститься опис методу *Page_Load*, який викликається при завантаженні веб-сторінки Default.aspx з віртуального каталогу для даного проекту. У даному прикладі метод містить пусте тіло.

4. Після компіляції проекту (опція меню «*Build*» або <*Shift* + *F6*>) і його виконання (<*Ctrl* + *F5*>) можна буде побачити в браузері сторінку приблизно наступного вигляду:

```
                                                                    Hello!
                                                                    I am
Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322; .NET CLR
2.0.50727; .NET CLR 3.0.04506.30; InfoPath.2; .NET CLR 3.0.04506.648; .NET CLR
3.5.21022)>
```

Контрольне завдання

Створіть ASP.NET додаток, що виводить список всіх *змінних оточення* веб-серверу.

Для цього буде потрібно вставка наступного коду:

```
<%
 // метод AllKeys формує список всіх ключів масиву
   string[] rkeys = Request.ServerVariables.AllKeys;

// створення змінної типу StringBuilder (тобто змінною рядка)
   StringBuilder output = new StringBuilder();

// перебір всіх елементів масиву rkeys і формування списку значень
   foreach (string rkey in rkeys)
   {
      output.Append(rkey + "=" + Request.ServerVariables[rkey] + "<br>");
   }
// формування відповіді сервера
   Response.Write(output);
%>
```

Лабораторна робота №9.

Тема «Серверні елементи управління ASP.NET.»

Мета роботи:

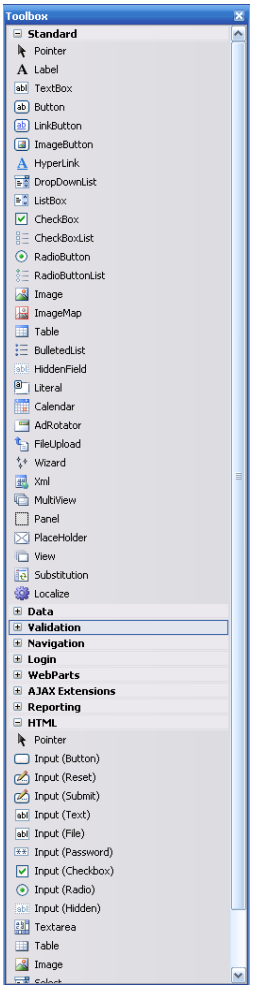
1. Знайомство із засобами автоматизації розробки веб-додатків в *ASP.NET* у вигляді серверних елементів управління *WebForm.*

Теоретичний матеріал

Важливою особливістю ASP.NET є використання *серверних елементів управління* на веб-сторінці (елементи *WebForm*), які фактично є тегами, які є зрозумілими веб-серверу. Ці елементи визначені в просторі імен *System.Web.UI.WebControls*.

Прийнято виділяти три типи серверних елементів управління:

- Серверні *елементи управління HTML* – звичайні HTML теги.
- Елементи *управління веб-сервера* – нові теги ASP.NET.
- Серверні елементи управління для *перевірки даних (валідації)* – застосовуються для валідації вхідних даних від клієнтського додатку (зазвичай веб-браузера).



Переваги від використання таких елементів при розробці веб-додатків:

- Скорочується кількість коду, написаного вручну (що особливо помітно для складних елементів документа). Елемент просто «перетягується» з панелі інструментів, після чого виконується налаштування його параметрів у спеціальному вікні. При цьому всі зміни автоматично заносяться безпосередньо в *.aspx файл.
- З програмної точки зору кожного з цих елементів управління відповідає певний клас в бібліотеці базових класів .NET, що дозволяє писати для них такий же код як і для будь-яких інших класів.
- Для будь-якого елементу управління WebForm визначений набір подій, оброблюваних на веб-сервері.
- Для будь-якого елементу управління WebForm надається можливість для перевірки введення даних користувачем.

1. Серверні елементи управління HTML

За замовчуванням такі елементи управління в ASP.NET файлах розглядаються як текст. Для їх програмування потрібно додавання атрибуту

runat="*server*" у відповідний HTML елемент. Крім того, всі серверні елементи управління HTML повинні бути розміщені всередині області дії тегу <*form*>, який також має атрибут *runat*="*server*".

Серверний елемент управління HTML	Опис
HtmlAnchor	Управління HTML елементом <a>
HtmlButton	Управління HTML елементом <button>
HtmlForm	Управління HTML елементом <form>
HtmlGeneric	Управляє HTML елементами, що не описуються як елементи управління HTML, наприклад, <body>, , та ін.
HtmlImage	Управління HTML елементом <image>
HtmlInputButton	Управління HTML елементами <input type="button">, <input type="submit"> та <input type="reset">
HtmlInputCheckBox	Управління HTML елементом <input type="checkbox">
HtmlInputFile	Управління HTML елементом <input type="file">
HtmlInputHidden	Управління HTML елементом <input type="hidden">
HtmlInputImage	Управління HTML елементом <input type="image">
HtmlInputRadioButton	Управління HTML елементом <input type="radio">
HtmlInputText	Управління HTML елементами <input type="text"> та <input type="password">
HtmlSelect	Управління HTML елементом <select>
HtmlTable	Управління HTML елементом <table>
HtmlTableCell	Управління HTML елементами <td> и <th>
HtmlTableRow	Управління HTML елементом <tr>
HtmlTextArea	Управління HTML елементом <textarea>

2. Елементи керування веб-серверу

Подібно серверним елементам керування, HTML елементи даного типу також створюються на веб-сервері і передбачають додавання атрибуту *runat*="*server*". Однак, вони можуть не відповідати конкретним елементам HTML, але представляти більш складні елементи.

Загальний синтаксис для опису таких елементів:

```
<asp:тип_елемента id="ідентифікатор" runat="server"/>
```

Перелік доступних елементів такого типу наведено в таблиці.

Елемент керування веб-серверу	Опис
AdRotator	Банерна рулетка
Button	Відображення кнопки
Calendar	Відображення календаря
CalendarDay	Елемент вибору дня календаря
CheckBox	Відображення прапорця
CheckBoxList	Група прапорців
DataGrid	Відображення полів джерела даних
DataList	Відображення елементів з джерела даних за допомогою шаблонів
DropDownList	Випадаючий список
HyperLink	Гіперпосилання
Image	Зображення
ImageButton	Кнопка у вигляді зображення
Label	Відображення статичного змісту, доступного для програмування, і з можливістю задання стилю
LinkButton	Кнопка с гіперпосиланням
ListBox	Випадаючий список з одиничним або множинним виділенням
ListItem	Елемент списку
Literal	Відображення статичного вмісту, доступного для програмування, але без можливості задання стилю
Panel	Контейнер для інших елементів керування
PlaceHolder	Місце для додавання елементів керування програмним способом
RadioButton	Радіо-кнопка
RadioButtonList	Група радіо-кнопок
BulletedList	Маркерний список
Repeater	Повторюючий список елементів
Style	Задання стилю елемента керування
Table	Таблиця
TableCell	Комірка таблиці
TableRow	Рядок таблиці
TextBox	Поле для введення тексту
Xml	Відображення XML файлу або результату XSLT перетворення

3. Серверні елементи управління для перевірки даних (валідації)

Елементи управління даного типу застосовуються для перевірки введених даних.

Мають наступний синтаксис:

<asp:*тип_элемента* id="*идентификатор*" runat="server" />

Найбільш важливі елементи наводяться в наступній таблиці.

Елемент керування для перевірки даних	**Опис**
CompareValidator	Порівнює значення, введене в один елемент керування зі значенням, введеним в інший елемент, або з фіксованим значенням
CustomValidator	Дозволяє задавати користувацький метод перевірки введених значень
RangeValidator	Перевіряє, що значення, введене користувачем, знаходиться між двома величинами
RegularExpressionValidator	Перевіряє введене значення на відповідність вказаному шаблону
RequiredFieldValidator	Перевіряє обов'язкову наявність введеного значення
ValidationSummary	Відображає звіт по всім помилкам перевірки значень, які виникли на веб-сторінці

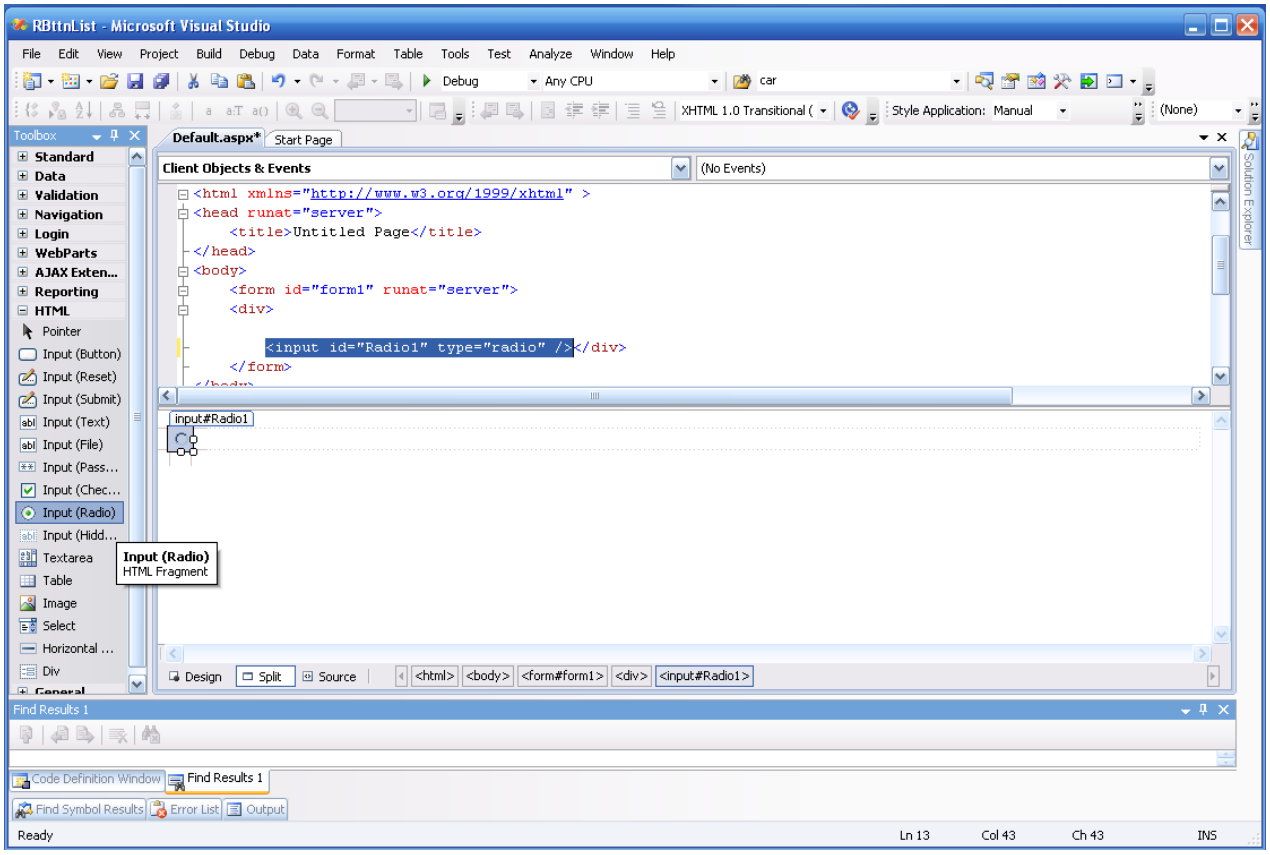
Порядок виконання лабораторної роботи

Для роботи з прикладами, що приводяться в даній лабораторній роботі, потрібно установка середовища розробки Microsoft Visual Studio 2005+ і веб-сервера IIS 5+ (Internet Information Server).

1. Серверні елементи керування HTML.

1) Створіть новий проект по шаблону *ASP.NET Web Application*.

2) На сторінці *Default.aspx* додайте в форму стандартні HTML елементи введення даних. Це можна зробити, використовуючи панель елементів управління, як це показано на скріншоті нижче.



Код форми повинен виглядати наступним чином:

```
<form id="form1" runat="server">
 <div>
  Ваша оцінка:<br><br>
 <input id="r1" type="radio" title="Отлично" value="Отлично"
runat="server"/>Відмінно <br />
 <input id="r2" type="radio" value="Хорошо" runat="server" />Добре<br />
 <input id="r3" type="radio" value="Удовлетворительно" runat="server"
/>Задовільно<br />
 <input id="r4" type="radio" value="Неудовлетворительно" runat="server"
/>Незадовільно<br />
 <p id="ans" runat="server"></p>
 <br>
 <input id="Submit1" type="submit" value="Отправить" OnServerClick="submit"
runat="server" /><br />
 </div>
</form>
```

Обробник події *OnServerClick* необхідно описати в файлі *Default.aspx.cs*. Код методу *submit*:

```
protected void submit(object sender, System.EventArgs e)
{
```

```
  if (r1.Checked == true)
    ans.InnerHtml = "Вы оценили на отлично";
  else if (r2.Checked == true)
    ans.InnerHtml = "Вы оценили на хорошо";
  else if (r3.Checked == true)
    ans.InnerHtml = "Вы оценили на удовлетворительно";
  else if (r4.Checked == true)
    ans.InnerHtml = "Вы оценили на неудовлетворительно";
  else
    ans.InnerHtml = "Вы не выбрали оценку";
}
```

3) Виконайте компіляцію проекту і запустіть його на виконання (або вручну завантажте документ в браузері, вказавши URL відповідного віртуального каталогу).

2. Елементи управління веб-серверу.

Якщо для новоствореного проекту типу *ASP.NET Web Application* у файлі *Default.aspx* всередині форми (між тегами <form> і </ form>) вставити наступний код (або зробити це перетягуванням потрібних елементів з панелі інструментів, розділ «Standard»):

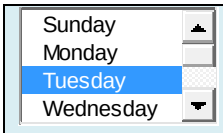
```
<asp:ListBox id="LstBx" runat="server" width="100" height="80"
AutoPostBack="True" OnSelectedIndexChanged="LBSelChanged" >
        <asp:ListItem value="Sunday">Sunday</asp:ListItem>
        <asp:ListItem value="Monday">Monday</asp:ListItem>
        <asp:ListItem value="Tuesday">Tuesday</asp:ListItem>
        <asp:ListItem value="Wednesday">Wednesday</asp:ListItem>
        <asp:ListItem value="Thursday">Thursday</asp:ListItem>
        <asp:ListItem value="Friday">Friday</asp:ListItem>
        <asp:ListItem value="Saturday">Saturday</asp:ListItem>
</asp:ListBox>
<br/><br/>
<asp:Label id="Label1" runat="server"/>
```

і описати обробку подій *Page_Load* і *OnSelectedIndexChanged* у відповідному **.aspx.cs* файлі:

```
protected void Page_Load(object sender, EventArgs e)
{
  Label1.Text = "You selected: " + Label1.Text;
}

protected void LBSelChanged(object sender, System.EventArgs e)
{
  if (LstBx.SelectedItem != null)
   Label1.Text = "You selected: " + LstBx.SelectedItem.Value;
```

```
  else
   Label1.Text = "You selected: ";
}
```

то при запуску проекту після вибору користувачем елемента зі списку можна буде побачити у вікні браузера:



You selected: Wednesday

У даному прикладі атрибут *AutoPostBack="True"* вказує на необхідність негайної обробки події на сервері.

3. Серверні елементи управління для перевірки даних (валідації)

Наступний код показує, яким чином виконується перевірка змісту поля введення форми:

```
<form id="form1" runat="server">
 <asp:TextBox ID="TextBox1" runat="server" Width="123px"></asp:TextBox>
 <br />
 <asp:RegularExpressionValidator
       ID="RegularExpressionValidator1"
       runat="server"
       ControlToValidate="TextBox1"
       ErrorMessage="Not valid e-mail"
       ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*" >
 </asp:RegularExpressionValidator>
</form>
```

В даному випадку атрибут *ControlToValidate* вказує на те, що контролюється зміст елемента з ідентифікатором *TextBox1* (поля введення тексту), у разі невідповідності змісту контрольованого поля з шаблоном, описаному у вигляді регулярного виразу в атрибуті *ValidationExpression*, видається повідомлення, вказане в атрибуті *ErrorMessage*.

Шаблон описує допустимий формат адреси електронної пошти. Додайте код в проект і перевірте роботу програми.

Контрольне завдання

Створіть веб-додаток *ASP.NET*, що підтримує введення на сторінці міжнародного телефонного номера користувачем і використовує серверний елемент управління <*asp:RegularExpressionValidator*>.

Лабораторна робота №10.

Тема «Робота з джерелами даних в ASP.NET.»

Мета роботи:

1. Вивчення елементів *WebForm*, призначених для відображення на веб-сторінці даних, одержуваних з джерел даних.

Теоретичний матеріал

У ASP.NET використовуються два елементи управління WebForm для управління відображенням даних, одержуваних з джерела даних:

- *DataGrid* – Елемент керування, що відображає зміст об'єкту ADO.NET *DataSet* у вигляді таблиці.
- *DataList* – Елемент керування для вибору значень, що заповнюються з джерела даних.

Якщо необхідно відобразити дані, отримані за запитом користувача з джерела даних, у вигляді таблиці на веб-сторінці, то ASP.NET надає в розпорядження веб-програміста зручний елемент керування *DataGrid*, який був введений в *ASP.NET 1.x*. але тепер його функції перекриваються *GridView*. Елемент управління *GridView* може не тільки показувати дані, але і сортувати, вибирати, редагувати їх. Якщо цієї функціональності недостатньо, її можна розширити, написавши власні обробники подій.

Елементи, які можуть бути пов'язані з джерелами даних, різноманітні, наприклад, *DropDownList, ListBox, CheckBoxList, RadioButtonList, BulletedList*.

1. Елемент GridView (DataGrid)

Елемент управління *GridView* – головний елемент керування для подання інформації з баз даних в ASP.NET. Він є «спадкоємцем» елемента управління *DataGrid*, який використовувався в попередніх версіях ASP.NET. Основне призначення цього елементу керування – представлення користувачу інформації в табличному вигляді з джерела даних з можливістю фільтрації, сортування та редагування.

Найпростіше налаштувати цей елемент керування за допомогою вбудованого майстра. При використанні майстра необхідно:

1. Вибрати тип джерела даних.
2. У властивостях джерела даних (елементу управління *DataSource*) налаштувати параметри підключення до бази даних.
3. Згенерувати або ввести код команд SELECT, INSERT, UPDATE і DELETE. При цьому необов'язково заповнювати дані для всіх команд.
4. Налаштувати додаткові параметри об'єкту GridView, наприклад, дозволити розбиття на сторінки, фільтрацію, сортування, вибрати оформлення і т. п.

2. DataList

У порівнянні з елементом керування *GridView* елемент управління *DataList* надає більше можливостей по налаштуванню зовнішнього вигляду даних, які беруться з джерела, однак вимагає значно більше часу і зусиль по налаштуванню. Додавати нові записи, також як і при використанні елемента керування *GridView*, засобами *DataList* неможливо.

Робота з цим елементом керування *DataList* (на закладці *Default.aspx* в режимі *Design*) виглядає наступним чином:

1. Після перетягування елементу керування *DataList* в форму, необхідно налаштувати для нього джерело даних (або використовуючи візуальний інтерфейс – за допомогою об'єкту *DataSource*, або програмно через властивість DataSource - DataSet);

2. Далі потрібно перейти в режим редагування шаблонів для записів. Для цього в контекстному меню вибрати *EditItemTemplate*. Для *Item Templates* з'являться чотири області:

- *ItemTemplate* – шаблон для відображення звичайних елементів на формі;
- *AlternatingItemTemplate* – необов'язковий шаблон для відображення кожного другого запису. Зазвичай для нього тільки трохи змінюється фон у порівнянні з *ItemTemplate* – щоб зручніше було сприймати довгі сторінки;
- *SelectedItemTemplate* – шаблон для відображення обраного в даний момент запису;
- *EditItemTemplate* – шаблон для відображення редагованого в даний момент запису.

Ці властивості можна також визначити і через розділ *Properties* в контекстному меню елемента керування *DataList*. Можна обмежитися також простим вибором одного з шаблонів автоформатування.

3. Потім потрібно заповнити необхідними елементами управління (і просто кодом HTML) кожен з шаблонів. Для кожного поля, інформацію про яке потрібно надавати користувачам, необхідно створити свій елемент управління. Для всіх шаблонів, крім *EditItemTemplate*, для більшості полів будуть використовуватися елементи управління *Label* (хоча можливі й інші варіанти). Для *EditItemTemplate* найчастіше використовуються елементи управління *TextBox*.

Щоб зв'язати елементи управління з полями в джерелі даних, найпростіше використати посилання *EditDataBindings* в майстрові, який з'являється для кожного з доданих елементів управління.

Порядок виконання лабораторної роботи

1. Використання елементу *GridView (DataGrid)*.

1) У наступному прикладі в якості джерела даних використовується база даних в форматі MS ACCESS, що містить таблицю *lesson* з полями:

- id (тип – лічильник);
- Title (тип – текст);
- Teacher (тип – текст);

– Type (тип – текст).

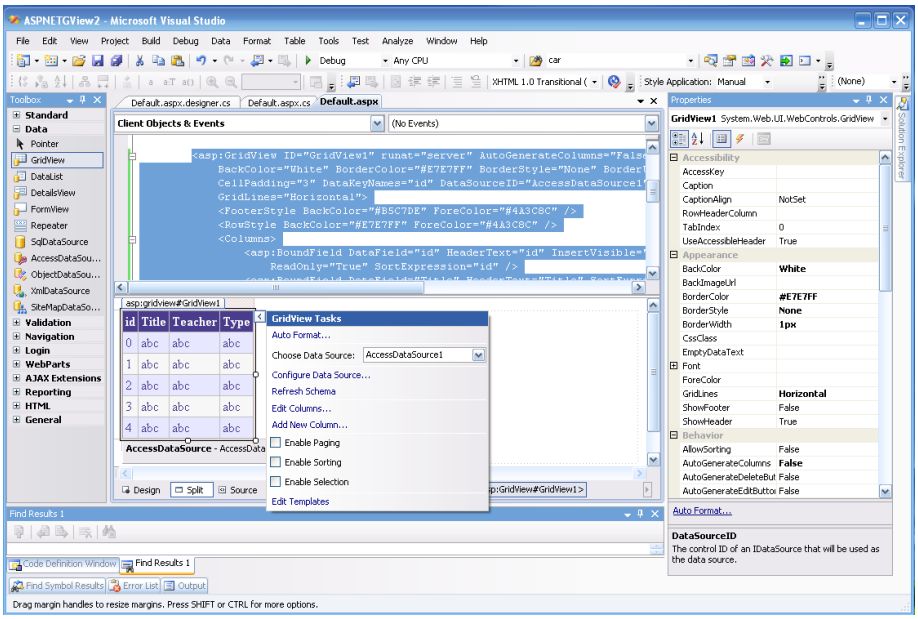
Створіть файл у форматі MS ACCESS, що містить цю таблицю.

2) Створіть новий проект по шаблону *ASP.NET Web Application*.

3) Для сторінки *Default.aspx* виберіть режим відображення *Split*

4) З панелі елементів управління *Toolbox* (відображається за допомогою меню *View*> *Toolbox*) перетягніть у форму елемент *GridView* (з групи *Data*).

5) Для елементу *GridView* налаштуйте джерело даних (*Choose Data Source, Configure Data Source*) і форматування (можна вибрати один із шаблонів автоформатування – *Auto Format*).



6) Автоматично згенерований код подання (файл *Default.aspx*) може виглядати наступним чином:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="ASPNETGView2._Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
   <title>Untitled Page</title>
</head>
<body>
   <form id="form1" runat="server">
   <div>

      <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
         BackColor="White" BorderColor="#E7E7FF" BorderStyle="None"
BorderWidth="1px"
         CellPadding="3" DataKeyNames="id" DataSourceID="AccessDataSource1"
```

```
        GridLines="Horizontal">
        <FooterStyle BackColor="#B5C7DE" ForeColor="#4A3C8C" />
        <RowStyle BackColor="#E7E7FF" ForeColor="#4A3C8C" />
        <Columns>
          <asp:BoundField DataField="id" HeaderText="id" InsertVisible="False"
            ReadOnly="True" SortExpression="id" />
          <asp:BoundField DataField="Title" HeaderText="Title"
SortExpression="Title" />
          <asp:BoundField DataField="Teacher" HeaderText="Teacher"
            SortExpression="Teacher" />
          <asp:BoundField DataField="Type" HeaderText="Type"
SortExpression="Type" />
        </Columns>
        <PagerStyle BackColor="#E7E7FF" ForeColor="#4A3C8C"
HorizontalAlign="Right" />
        <SelectedRowStyle BackColor="#738A9C" Font-Bold="True"
ForeColor="#F7F7F7" />
        <HeaderStyle BackColor="#4A3C8C" Font-Bold="True" ForeColor="#F7F7F7"
/>
        <AlternatingRowStyle BackColor="#F7F7F7" />
      </asp:GridView>
      <asp:AccessDataSource ID="AccessDataSource1" runat="server"
        DataFile="D:\CSharp\ASPNETDataGrid\lessons.mdb"
        SelectCommand="SELECT * FROM [lesson]"></asp:AccessDataSource>

    </div>
    </form>
</body>
</html>
```

У даному прикладі використаний елемент управління <*asp: GridView*>, що має ідентифікатор «*GridView1*».

7) Якщо джерело даних не налаштований автоматично (при описі коду подання):

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

      <asp:DataGrid ID="GridView1" runat="server" BackColor="White"
```

```
    BorderColor="White" BorderStyle="Ridge" BorderWidth="2px" CellPadding="3"
    CellSpacing="1" GridLines="None" >

    <FooterStyle BackColor="#C6C3C6" ForeColor="Black" />
    <SelectedItemStyle BackColor="#9471DE" Font-Bold="True" ForeColor="White"
/>
    <PagerStyle BackColor="#C6C3C6" ForeColor="Black" HorizontalAlign="Right"
/>
    <ItemStyle BackColor="#DEDFDE" ForeColor="Black" />
    <HeaderStyle BackColor="#4A3C8C" Font-Bold="True" ForeColor="#E7E7FF" />

    </asp:DataGrid>

  </div>
  </form>
</body>
</html>
```

то це можна зробити програмно у відповідному програмному коді, наприклад:

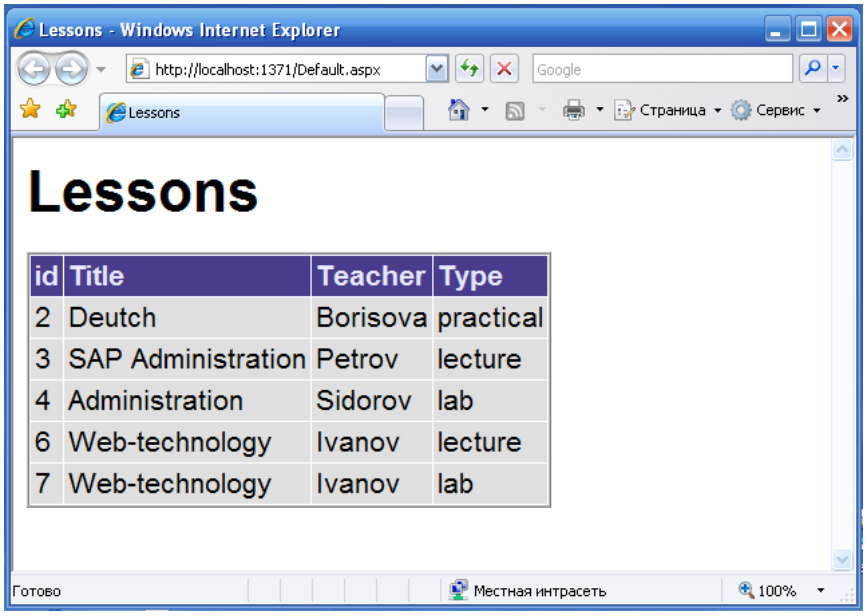
```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.Data.OleDb;

namespace ASPNETDataGrid
{
  public partial class _Default : System.Web.UI.Page
  {
    protected void Page_Load(object sender, EventArgs e)
    {
      if (!IsPostBack)
      {

// Підключення до джерела даних

        OleDbConnection cn = new OleDbConnection();
        cn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;" + @"data source
```

```
= D:\CSharp\ASPNETDataGrid\lessons.mdb";
            cn.Open();

// Формується рядок SQL запиту даних з джерела

            string str = "SELECT * from lesson";

// Відбувається з'єднання з базою даних
// За допомогою керованого провайдера OLE DB

            OleDbDataAdapter dAdapt = new OleDbDataAdapter(str, cn);

// Отримання даних з джерела

            DataSet myDS = new DataSet("lessons");

            dAdapt.Fill(myDS, "lesson");

// Заповнення таблиці даними

            GridView1.DataSource = myDS.Tables["lesson"].DefaultView;
            GridView1.DataBind();

            cn.Close();

        }
    }
  }
}
```

Слід звернути увагу на те, що в програмі додано простір імен:

- using System.Data.OleDb. Властивість об'єкту **Page** – IsPostBack показує, чи була сторінка клієнта завантажена в перший раз або повторно у відповідь на передані клієнтом дані.

8) Результат виконання виглядає наступним чином:



6. Використання елемента *DataList*.

1) У наступному нижче прикладі в якості джерела даних використовується база даних в форматі MS ACCESS, що містить таблицю *lesson* (як у попередньому прикладі).

2) Створіть новий проект по шаблону *ASP.NET Web Application*.

3) Для сторінки *Default.aspx* виберіть режим відображення *Split*.

4) З панелі елементів управління *Toolbox* (відображається за допомогою меню *View*> *Toolbox*) перетягніть у форму елемент *DataList* (з групи *Data*).

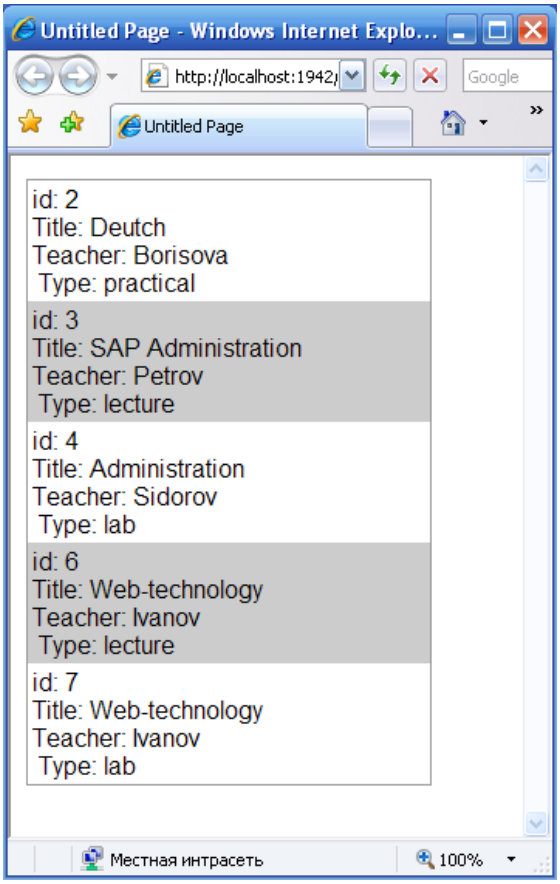
5) Для елементу *DataList* налаштуйте джерело даних (*Choose Data Source, Configure Data Source*) і форматування (можна вибрати один із шаблонів автоформатування – *Auto Format*).

6) Вийде наступний код подання:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="Default.aspx.cs" Inherits="ASPNETGView2._Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
   <title>Untitled Page</title>
</head>
<body>
   <form id="form1" runat="server">
   <div>

      <asp:DataList ID="DataList1" runat="server" BackColor="White"
         BorderColor="#999999" BorderStyle="Solid" BorderWidth="1px"
CellPadding="3"
         DataKeyField="id" DataMember="DefaultView"
```

```
DataSourceID="AccessDataSource1"
        ForeColor="Black" GridLines="Vertical" style="margin-right: 0px"
Width="262px" >
        <FooterStyle BackColor="#CCCCCC" />
        <AlternatingItemStyle BackColor="#CCCCCC" />
        <SelectedItemStyle BackColor="#000099" Font-Bold="True"
ForeColor="White" />
        <HeaderStyle BackColor="Black" Font-Bold="True" ForeColor="White" />
        <ItemTemplate>
          id:
          <asp:Label ID="idLabel" runat="server" Text='<%# Eval("id") %>' />
          <br />
          Title:
          <asp:Label ID="TitleLabel" runat="server" Text='<%# Eval("Title") %>' />
          <br />
          Teacher:
          <asp:Label ID="TeacherLabel" runat="server" Text='<%# Eval("Teacher")
%>' />
           <br />
           Type:
          <asp:Label ID="TypeLabel" runat="server" Text='<%# Eval("Type") %>'
/>
          <br />
        </ItemTemplate>
      </asp:DataList>

      <asp:AccessDataSource ID="AccessDataSource1" runat="server"
        DataFile="D:\CSharp\ASPNETDataGrid\lessons.mdb"
        SelectCommand="SELECT * FROM [lesson]">
        </asp:AccessDataSource>


   </div>
   </form>
</body>
</html>
```

Результат роботи в браузері:



7) Альтернативний варіант – використання методу *GetDataItem* обєкту *DataBinder* для заповнення елементів замість <*asp: Label*>.

Нижче представлений відповідний код подання:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="ASPNETDataGrid._Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head2" runat="server">
   <title>Lessons</title>
</head>
<body>
   <form id="form2" runat="server">
   <div>
      <h1>Lessons</h1>
   </div>
   <asp:DataList ID="DataList1" runat="server" BackColor="White"
      BorderColor="White" BorderStyle="Ridge" BorderWidth="2px" CellPadding="3"
      CellSpacing="1">
         <FooterStyle BackColor="#C6C3C6" ForeColor="Black" />
         <ItemStyle BackColor="#DEDFDE" ForeColor="Black" />
         <SelectedItemStyle BackColor="#9471DE" Font-Bold="True"
```

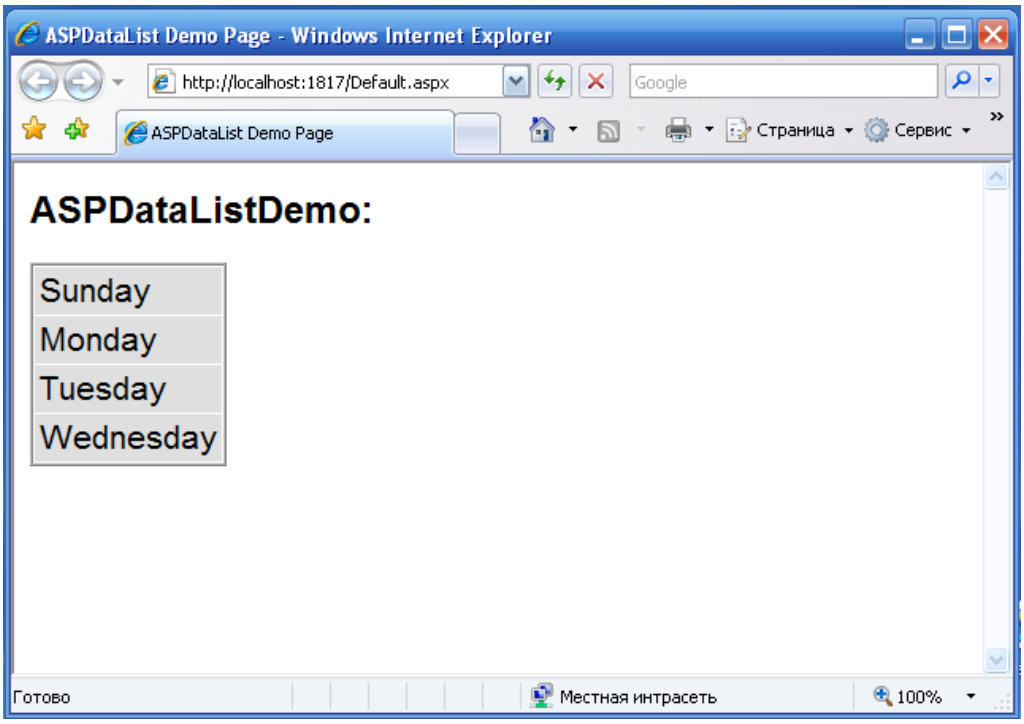
```
ForeColor="White" />
        <HeaderStyle BackColor="#4A3C8C" Font-Bold="True" ForeColor="#E7E7FF"
/>
        <ItemTemplate>
          <%# DataBinder.GetDataItem(Container)%>
  </ItemTemplate>
   </asp:DataList>

   </form>
</body>
</html>
```

Програмний код:

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;

namespace ASPNETDataGrid
{
   public partial class _Default : System.Web.UI.Page
   {
      protected void Page_Load(object sender, EventArgs e)
      {
         if (!IsPostBack)
         {
            ArrayList DWeek = new ArrayList();
            DWeek.Add("Sunday");
            DWeek.Add("Monday");
            DWeek.Add("Tuesday");
            DWeek.Add("Wednesday");
            DWeek.Add("Thursday");
            DWeek.Add("Friday");
            DWeek.Add("Saturday");

            DataList1.DataSource = DWeek;
            DataList1.DataBind();
         }
```

```
        }
    }
}
```

Результат, який відображається в веб-браузері:



3. Використання елемента *ListBox* (із заповненням даними із звичайного масиву).

1) Створіть новий проект по шаблону *ASP.NET Web Application*.

2) Для сторінки *Default.aspx* виберіть режим відображення *Split*

3) З панелі елементів управління *Toolbox* (відображається за допомогою меню *View*> *Toolbox*) перетягніть у форму елемент *ListBox* (з групи Data).

4) Вийде наступний код подання:

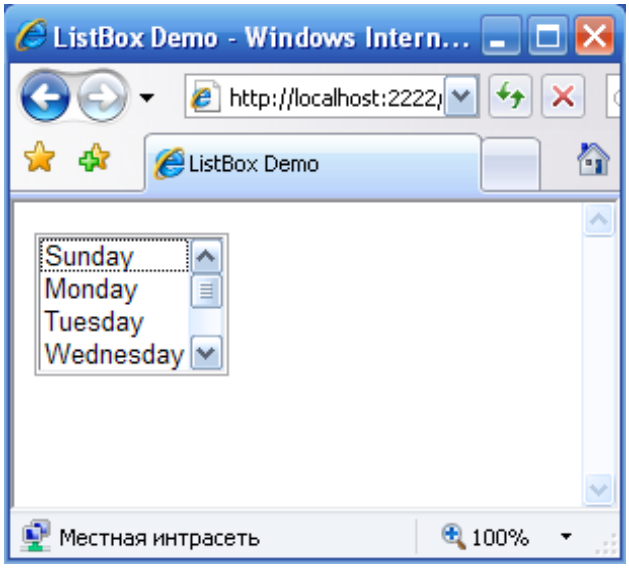
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="ASPNETListBox._Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
   <title>Untitled Page</title>
</head>
<body>

   <form id="Form1" runat="server">
```

```
    <asp:ListBox ID="ListBox1" runat="server"></asp:ListBox>

    </form>
</body>
</html>
```

і відповідний програмний код:

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;

namespace ASPNETListBox
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

// Масив рядків, який потрібно вставити

            ArrayList DWeek = new ArrayList();
            DWeek.Add("Sunday");
            DWeek.Add("Monday");
            DWeek.Add("Tuesday");
            DWeek.Add("Wednesday");
            DWeek.Add("Thursday");
            DWeek.Add("Friday");
            DWeek.Add("Saturday");

// Зв'язування елемента управління з об'єктом DWeek

            ListBox1.DataSource = DWeek;
            ListBox1.DataBind();
        }

    }
}
```

Результат виконання:



Контрольне завдання.

Створіть веб-додаток ASP.NET, що відображає на веб-сторінці зміст таблиці *Exam*, що містить поля: *ідентифікатор запису, прізвище студента, назва дисципліни, оцінка*. При розробці використовуйте елемент <*asp: GridView*> або <*asp: DataList*>.

Лабораторна робота №11.

Тема «Структура XML документа. XML-схеми.»

Мета роботи:

1. Ознайомлення зі структурою XML-документа, основами синтаксису мови та методами контролю вмісту документа за допомогою схем.

Теоретичний матеріал

XML-документ можна представити у вигляді дерева, яке починається з «кореня» і закінчується «листям».

Нижче наведено приклад XML-документа:

```
<?xml version="1.0" encoding="Windows-1251"?>
<mail>
  <to>user1@domain.ru</to>
  <from>user2@domain.ru </from>
  <subject>Зустріч</heading>
  <body>Зателефонуй мені завтра вранці</body>
</mail>
```

У першому рядку поміщена XML декларація, що описує версію XML і використане кодування.

У наступному рядку описаний кореневий елемент документа.

У наступних чотирьох рядках описано 4 *дочірніх* елемента відносно кореневого (*to, from*, *subject* і *body*).

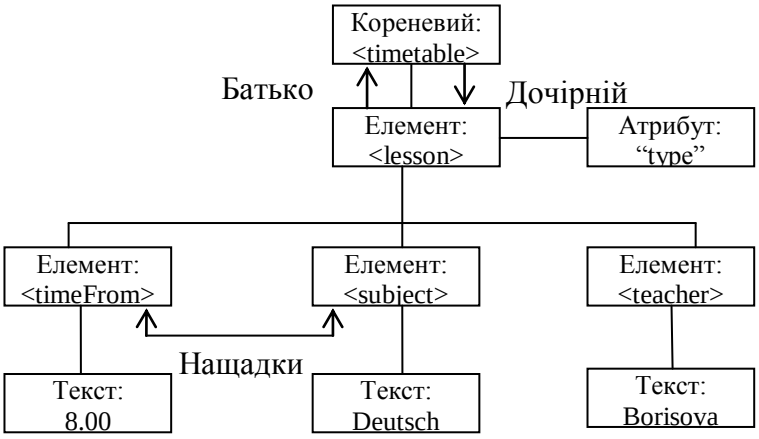
І в останньому рядку вказується *кінець кореневого* елемента.

XML-документ повинен містити *кореневий* елемент, який є *батьківським* для всіх інших елементів. Будь-який елемент (крім тих, що знаходяться на самому нижньому рівні дерева) може мати вкладені елементи (дочірні елементи).

За допомогою термінів *батько*, *дочірній* і *нащадок* описуються відносини між елементами в дереві XML-документа. *Батьки* містять *дочірні елементи*, а дочірні елементи одного рівня називаються *нащадками* (братами або сестрами).

Елементи можуть мати *текстовий вміст* і *атрибути*.

Нижче наведена схема, що показує відношення між елементами і атрибутами цього документа:



Крім того в XML-документі слід виконувати ще кілька нескладних правил:

- Кожен XML-елемент повинен мати закриваючий тег.
- XML-теги чутливі до регістру.
- Повинні дотримуватися правила вкладеності елементів.
- XML-документи повинні мати кореневий елемент, який є батьківським для всіх інших елементів.
- Значення XML-атрибутів повинні братися в подвійні лапки.

Деякі із символів (їх всього 5) мають спеціальне значення в форматі XML. Для їх позначення використовуються альтернативні поєднання:

заміна	символ
&lt;	<
&gt;	>
&amp;	&
&apos;	'
&quot;	"

Коментарі в XML-виділяються також як і в HTML:

<!-- Це коментар -->

На відміну від HTML, який ігнорує повтори пробілів, XML зберігає повну довжину рядка з прогалин.

XML-документи, оформлені відповідно до наведених вище правил, називаються «правильно оформленими». Якщо при цьому вони ще й відповідають правилам DTD (Document Type Definition), то вони є і «дійсними».

Дійсні XML-документи містять посилання на DTD файл, наприклад:

<!DOCTYPE note SYSTEM "mail.dtd">

DTD схема призначена для визначення структури XML-документа шляхом задання списку допустимих елементів, наприклад:

```
<!DOCTYPE mail [
  <!ELEMENT note (to,from,subject,body)>
  <!ELEMENT to      (#PCDATA)>
  <!ELEMENT from    (#PCDATA)>
  <!ELEMENT subject (#PCDATA)>
  <!ELEMENT body    (#PCDATA)>
]>
```

Консорціум W3C підтримує альтернативну до DTD, засновану на XML-схемі, звану як «*XML Schema*»:

```
<xs:element name="mail">
<xs:complexType>
 <xs:sequence>
  <xs:element name="to"      type="xs:string"/>
  <xs:element name="from"    type="xs:string"/>
  <xs:element name="subject" type="xs:string"/>
  <xs:element name="body"    type="xs:string"/>
 </xs:sequence>
</xs:complexType>
</xs:element>
```

Для перевірки дійсності XML-документа можна використовувати спеціальні валідатори, наприклад *W3C валідатор* (http://validator.w3.org/). Для перевірки схем також існують спеціальні валідатори, наприклад *валідатор XML Schema* (http://www.w3.org/2001/03/webdata/xsv).

Згідно специфікації W3C XML програма повинна припинити обробку XML-документа, як тільки буде виявлена помилка в цьому документі.

1. DTD схема

DTD (Document Type Definition) визначає допустимі будівельні блоки XML-документа, шляхом зазначення списку допустимих елементів і атрибутів.

DTD може описуватися як усередині XML-документа, так і за допомогою зовнішнього посилання.

Приклад внутрішнього опису:

```
<?xml version="1.0"?>
<!DOCTYPE note [
 <!ELEMENT mail (to,from,subject,body)>
 <!ELEMENT to      (#PCDATA)>
 <!ELEMENT from    (#PCDATA)>
 <!ELEMENT subject (#PCDATA)>
 <!ELEMENT body    (#PCDATA)>
]>
<note>
 <to>user1@domain.ru</to>
 <from>user2@domain.ru</from>
 <heading>Зустріч</heading>
 <body> Зателефонуй мені завтра вранці </body>
</note>
```

У даному прикладі:

- **!DOCTYPE mail** визначає кореневий елемент документа **mail**.
- **!ELEMENT note** визначає елемент **note**, який містить чотири елементи: "*to, from, subject, body*".
- **!ELEMENT to** визначає елемент **to** типу "#PCDATA".

- **!ELEMENT from** визначає елемент **from** типу "#PCDATA".
- **!ELEMENT subject** визначає елемент **subject** типу "#PCDATA".
- **!ELEMENT body** визначає елемент **body** типу "#PCDATA".

Приклад зовнішнього опису:

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "mail.dtd">
<note>
 <to>user1@domain.ru</to>
 <from>user2@domain.ru</from>
 <heading>Встреча</heading>
 <body> Зателефонуй мені завтра вранці </body>
</note>
```

Для чого необхідне використання DTD?

- За допомогою DTD XML-файли можуть містити опис власного формату.
- Незалежні групи людей можуть обмінюватися даними.
- DTD-схема може бути використана для перевірки дійсності, як документів одержуваних ззовні, так і власних документів.

З точки зору DTD XML-документи все (HTML документи) будуються з наступних будівельних блоків:

- *Елементи* (Elements). Елементи можуть містити текст, інші елементи або бути порожніми.
- Атрибути (Attributes). Атрибути завжди розміщуються всередині відкриваючого тега елемента. Атрибути завжди записуються у вигляді пари *ім'я / значення*.
- *Entities*.
- PCDATA. Означає структуровані символьні дані. Підлягає аналізу за допомогою парсера на наявність спеціальних символів (entities) та елементів розмітки.
- CDATA. Означає текст, зміст якого не розглядається аналізатором.

При описі структури елементів, що містять вкладені елементи можна використовувати спеціальні оператори, що задають множинність їх входження:

- '+': один і більше разів, наприклад, <*!ELEMENT note (message+)*>;
- '*': нуль і більше разів, наприклад, <*!ELEMENT note (message*)*>;
- '?': нуль і один раз, наприклад, <*!ELEMENT note (message?)*>

Оголошення виду <*!ELEMENT note (message|body)*> вказує на можливість входження в елемент *note* іншого елемента: *message* АБО *body*.

Атрибути елементів описуються у формі:

<*!ATTLIST element-name attribute-name attribute-type default-value*>

наприклад:

```
<!ATTLIST payment type CDATA "check">
```

Що відповідає XML-коду:

<payment type="check" />.

Тип атрибута може приймати одне із значень:

Тип	Опис
CDATA	Символьні данні
(*en1*|*en2*|..)	Значення зі списку
ID	Унікальний *id*
IDREF	*id* іншого елемента
IDREFS	Список інших *id*
NMTOKEN	Допустиме XML ім'я
NMTOKENS	Список допустимих XML імен
ENTITY	Спеціальні символи
ENTITIES	Список спеціальних символів
NOTATION	Ім'я нотації
xml:	Зумовлене XML ім'я

Значення за замовчуванням може бути одним з наступних:

Значення	Інтерпретація
#REQUIRED	Обов'язково
#IMPLIED	Не обов'язково
#FIXED *value*	Значення фіксовано

2. XML Schema

Так само як DTD схема XML Schema визначає допустимі будівельні блоки XML-документа.

XML Schema:

- елементи, що входять в документ;
- атрибути, допустимі в документі;
- дочірні елементи;
- порядок дочірніх елементів;
- кількість дочірніх елементів;
- чи може елемент бути порожнім або може містити текст;
- типи елементів і атрибутів;
- фіксовані і значення за замовчуванням елементів і атрибутів.

Передбачається, що в перспективі DTD схеми будуть заміщені XML Schema в більшості веб-додатків оскільки XML-схема:

- Розширювана для майбутніх доповнень.
- Більш багаті і могутні виразні можливості.
- Є реалізацією XML.
- Підтримує типи даних.

- Підтримують простори імен.

XML Schema стала W3C рекомендацією з 2001 року.

Розглянемо як приклад XML-документ:

```
<?xml version="1.0" encoding="Windows-1251"?>
<mail>
  <to>user1@domain.ru</to>
  <from>user2@domain.ru </from>
  <subject>Зустріч</heading>
  <body>Зателефонуй мені завтра вранці</body>
</mail>
```

Структура даного документа може бути розглянута за допомогою наступної *XML Schema*:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.myhp.edu"
xmlns=" http://www.myhp.edu"
elementFormDefault="qualified">
<xs:element name="mail">
  <xs:complexType>
    <xs:sequence>
       <xs:element name="to" type="xs:string"/>
       <xs:element name="from" type="xs:string"/>
       <xs:element name="subject" type="xs:string"/>
       <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

У даному прикладі елемент *mail* має тип **ComplexType**, оскільки містить інші елементи. Інші елементи документа мають простий тип, оскільки не включають інших елементів.

Посилання на схему в XML-документі виглядає наступним чином:

```
<?xml version="1.0"?>
<mail
xmlns="http:// www.myhp.edu "
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.myhp.edu mail.xsd">
  <to>user1@domain.ru</to>
  <from>user2@domain.ru </from>
  <subject>Зустріч</heading>
  <body> Зателефонуй мені завтра вранці body>
        </mail>
```

Елемент <schema> є кореневим елементом будь-якої схеми XML Schema.
Даний елемент може містити кілька атрибутів, наприклад:

```
<?xml version="1.0"?>
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.myhp.edu"
  xmlns="http://www.myhp.edu"
  elementFormDefault="qualified">
  ...
  ...
 </xs:schema>
```

Наступний фрагмент:

```
xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

Вказує на те, що елементи і типи даних, що використовуються в схемі входять в *простір http://www.w3.org/2001/XMLSchema*. Причому, всі елементи і типи даних з цього простору імен повинні мати префікс **xs**.

Фрагмент:

```
targetNamespace="http://www.myhp.edu"
```

вказує на те, що елементи, які визначаються в схемі входять в простір «http://www.myhp.edu».

У фрагменті:

```
xmlns="http://www.myhp.edu"
```

вказується, що простором імен за замовчуванням є «http://www.myhp.edu».

Наступний фрагмент:

```
elementFormDefault="qualified"
```

вказує на те, що будь-які елементи, оголошені в схемі, повинні належати простору імен.

Посилання на зовнішню схему може виглядати наступним чином:

```
<?xml version="1.0"?>
<mail xmlns=" http://www.myhp.edu "
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation=" http://www.myhp.edu mail.xsd">
   <to>user1@domain.ru</to>
   <from>user2@domain.ru </from>
   <subject>Зустріч</heading>
```

```
  <body> Зателефонуй мені завтра вранці </body>
</mail>
```

Тут рядок:

```
xsi:schemaLocation=" http://www.myhp.edu mail.xsd">
```

вказує на місце розташування файлу схеми.

3. Визначення простих елементів

Синтаксис для визначення простого елементу:

```
<xs:element name="xxx" type="yyy"/>
```

де *xxx* - ім'я елементу і *yyy* тип даних елемента.
Вбудованими типами даних елементів є наступні:

- xs:string;
- xs:decimal;
- xs:integer;
- xs:boolean;
- xs:date;
- xs:time.

Наприклад, фрагмент XML-документа:

```
<lastname>Refsnes</lastname>
<age>36</age>
<dateborn>1970-03-27</dateborn>
```

описується в схемі наступним чином:

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

Наступні фрагменти:

```
<xs:element name="color" type="xs:string" default="red"/>
```

і

```
<xs:element name="color" type="xs:string" fixed="red"/>
```

описують значення елемента за замовчуванням і фіксоване значення відповідно.

Усі атрибути описуються простими типами даних.

Прості елементи не можуть мати атрибутів. Якщо елемент має атрибути, він розглядається як той, що має складний тип. Сам атрибут розглядається завжди як має простий тип.

Опис атрибуту зазвичай дається наступним чином:

```
<xs:attribute name="xxx" type="yyy"/>
```

де *xxx* – ім'я атрибута, а *yyy* – визначає тип даних атрибута.

Вбудованими типами даних для атрибутів є наступні:

- xs:string;
- xs:decimal;
- xs:integer;
- xs:boolean;
- xs:date;
- xs:time.

Наприклад, XML-елемент з атрибутом:

```
<lastname lang="EN">Smith</lastname>
```

описується відповідною схемою:

```
<xs:attribute name="lang" type="xs:string"/>
```

Значення за замовчуванням і фіксоване значення атрибуту описуються наступним чином:

```
<xs:attribute name="lang" type="xs:string" default="EN"/>
```

і

```
<xs:attribute name="lang" type="xs:string" fixed="EN"/>
```

відповідно.

Для опису обов'язкових атрибутів (за замовчуванням всі атрибути є необов'язковими) використовується наступна нотація:

```
<xs:attribute name="lang" type="xs:string" use="required"/>
```

4. Обмеження на зміст

Для завдання допустимих значень величин XML-елементів та атрибутів можна використовувати обмеження. Обмеження на значення XML-елементів називаються *фасетами*.

Список можливих обмежень наводиться нижче:

Обмеження	Опис
enumeration	Визначає список допустимих значень
fractionDigits	Вказує максимальну кількість десяткових позицій. Має бути невід'ємним.
length	Вказує точне число символів або елементів у списку. Маєбути невід'ємним
maxExclusive	Вказує верхню межу числових значень
maxInclusive	Вказує верхню межу числових значень
maxLength	Вказує на максимальну кількість символів або елементів у списку. Має бути невід'ємним
minExclusive	Визначає нижню межу числових значень, що не включається
minInclusive	Визначає нижню межу числових значень, що включається
minLength	Визначає мінімальне число символів або елементів у списку. Повинно бути невід'ємним
pattern	Визначає точно послідовність символів (шаблон), яка є допустимою
totalDigits	Визначає точне число цифр. Повинно бути невід'ємним
whiteSpace	Визначає як обробляти не відображаються символи (пробіл, табуляція і ін.)

Приклади обмеження:

```
<xs:element name="age">
<xs:simpleType>
 <xs:restriction base="xs:integer">
  <xs:minInclusive value="0"/>
  <xs:maxInclusive value="120"/>
 </xs:restriction>
</xs:simpleType>
</xs:element>
```

```
<xs:element name="password">
<xs:simpleType>
 <xs:restriction base="xs:string">
  <xs:pattern value="[a-zA-Z0-9]{8}"/>
 </xs:restriction>
</xs:simpleType>
</xs:element>
```

5. Опис складних елементів

При описі документів, що мають складну ієрархічну структуру, можна спочатку визначити всі елементи і атрибути, а потім посилатися на них.

Розглянемо наступний приклад:

```
<employee>
   <firstname>John</firstname>
   <lastname>Smith</lastname>
</employee>
```

Складний елемент у схемі можна визначити наступним чином:

```
<xs:element name="firstname" type="xs:string"/>
<xs:element name="lastname" type="xs:string"/>

<xs:element name="employee">
        <xs:complexType>
                <xs:sequence>
                        <xs:element ref="firstname"/>
                        <xs:element ref="lastname"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>
```

Інший підхід полягає в повторному використанні іменованих елементів.

1. Елемент *«employee»* element може бути оголошений безпосередньо на ім'я:

```
<xs:element name="employee">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Оскільки дочірні елементи «*firstname*» і «*lastname*» оточені *індикатором* <sequence> то вони повинні в документі з'являтися саме в цій послідовності.

2. Елемент «*employee*» може мати атрибут типу, який посилається на ім'я складного типу:

```
<xs:element name="employee" type="personinfo"/>

<xs:complexType name="personinfo">
 <xs:sequence>
  <xs:element name="firstname" type="xs:string"/>
  <xs:element name="lastname" type="xs:string"/>
 </xs:sequence>
</xs:complexType>
```

Кілька елементів можуть посилатися на той же самий складний тип:

```
<xs:element name="employee" type="personinfo"/>
<xs:element name="student" type="personinfo"/>
<xs:element name="member" type="personinfo"/>

<xs:complexType name="personinfo">
 <xs:sequence>
  <xs:element name="firstname" type="xs:string"/>
  <xs:element name="lastname" type="xs:string"/>
 </xs:sequence>
</xs:complexType>
```

Також можна будувати складний елемент на базі існуючого складного елемента з додаванням кількох елементів:

```
<xs:element name="employee" type="fullpersoninfo"/>

<xs:complexType name="personinfo">
 <xs:sequence>
  <xs:element name="firstname" type="xs:string"/>
  <xs:element name="lastname" type="xs:string"/>
 </xs:sequence>
</xs:complexType>

<xs:complexType name="fullpersoninfo">
 <xs:complexContent>
  <xs:extension base="personinfo">
   <xs:sequence>
    <xs:element name="address" type="xs:string"/>
    <xs:element name="city" type="xs:string"/>
    <xs:element name="country" type="xs:string"/>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

Порожній складний елемент не може мати змісту, тільки атрибути.

За допомогою спеціальних індикаторів можна вказувати, як елементи можуть використовуватися в документах.

Всього використовується сім індикаторів:

- Індикатор порядка (*Order*):
  - All;
  - Choice;
  - Sequence.
- Індикатор входження (*Occurrence*):
  - maxOccurs;
  - minOccurs.

- Групові (*Group*) індикатори:
  - Group name;
  - attributeGroup name.

Приклад:

```
<xs:element name="person">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="full_name" type="xs:string"/>
   <xs:element name="child_name" type="xs:string" maxOccurs="10"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

Порядок виконання лабораторної роботи

1. Опис структури XML, документа за допомогою DTD-схеми.

1) Створення XML-документа *timetable.xml*. Зміст документа наведено нижче:

```
<?xml version="1.0"?>
<timetable>
	<day dayOfWeek="Monday">
		<lesson type="practical">
			<timeFrom>08.00</timeFrom>
			<timeTo>09.30</timeTo>
			<subject>Deutsch</subject>
			<teacher>Borisova</teacher>
			<room>216</room>
		</lesson>
		<lesson type="lecture">
			<timeFrom>09.40</timeFrom>
			<timeTo>11.10</timeTo>
			<subject>SAP Administration</subject>
			<teacher>Egorov</teacher>
			<room>384</room>
		</lesson>
		<lesson  type="practical">
			<timeFrom>11.20</timeFrom>
			<timeTo>12.50</timeTo>
			<subject>SAP Administration</subject>
			<teacher>Petrov</teacher>
			<room>384</room>
		</lesson>
	</day>
</timetable>
```

Завантажте цей файл в браузері. Зверніть увагу на ієрархічну структуру документа.

2) Додайте в початок файлу внутрішню DTD-схему:

```
<?xml version="1.0"?>

<!DOCTYPE timetable[
<!ELEMENT timetable (day*)>
<!ELEMENT day (lesson+)>
<!ELEMENT lesson (timeFrom, timeTo, subject, teacher, room?)>
<!ELEMENT timeFrom (#PCDATA)>
<!ELEMENT timeTo (#PCDATA)>
<!ELEMENT subject (#PCDATA)>
<!ELEMENT teacher (#PCDATA)>
<!ELEMENT room (#PCDATA)>
<!ATTLIST day
        dayOfWeek CDATA #REQUIRED>
<!ATTLIST lesson
        type CDATA #REQUIRED>
]>
```

3) Збережіть файл і перевірте його на правильність за допомогою програми-валідатора, наприклад, за адресою: http://validator.w3.org. Спробуйте видалити який-небудь елемент або атрибут, перевірте документ знову за допомогою валідатора.

4) Винесіть схему в зовнішній файл з розміщенням на ньому посилання всередині XML-документа. Перевірте, як буде завантажуватися XML-документ.

2. Опис структури XML-документа за допомогою XML Schema.

1) Створення XML-документа *timetable2.xml*. Зміст документа наведено нижче.

```
<?xml version="1.0"?>

<timetable xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="timetable.xsd">

<day dayOfWeek="Monday">
 <lesson type="practical">
    <timeFrom>08.00</timeFrom>
    <timeTo>09.30</timeTo>
    <subject>Deutsch</subject>
    <teacher>Borisova</teacher>
    <room>216</room>
  </lesson>
  <lesson type="lecture">
    <timeFrom>09.40</timeFrom>
```

```
    <timeTo>11.10</timeTo>
    <subject>SAP Administration</subject>
    <teacher>Egorov</teacher>
    <room>384</room>
   </lesson>
   <lesson  type="practical">
    <timeFrom>11.20</timeFrom>
    <timeTo>12.50</timeTo>
    <subject>SAP Administration</subject>
    <teacher>Petrov</teacher>
    <room>384</room>
   </lesson>
</day>
</timetable>
```

2) Створення файлу *timetable.xsd,* що містить XML Schema:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

        <xs:element name = "timetable">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element ref = "day" minOccurs = "0" maxOccurs =
"unbounded"/>
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
        <xs:element name = "day">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element ref = "lesson" maxOccurs = "unbounded"/>
                        </xs:sequence>
                        <xs:attribute name = "dayofWeek" use = "required" type =
"xs:string"/>
                </xs:complexType>
        </xs:element>
        <xs:element name = "lesson">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element ref = "timeFrom"/>
                                <xs:element ref = "timeTo"/>
                                <xs:element ref = "subject"/>
                                <xs:element ref = "teacher"/>
                                <xs:element ref = "room" minOccurs = "0"/>
                        </xs:sequence>
                </xs:complexType>
```

```
        </xs:element>
        <xs:element name = "timeFrom" type = "xs:string"/>
        <xs:element name = "timeTo" type = "xs:string"/>
        <xs:element name = "subject" type = "xs:string"/>
        <xs:element name = "teacher" type = "xs:string"/>
        <xs:element name = "room" type = "xs:string"/>

</xs:schema>
```

3) Перевірте схему на правильність за допомогою програми-валідатора, наприклад, за адресою: *http://www.w3.org/2001/03/webdata/xsv*.

Контрольне завдання

У наведеному нижче XML-документі описана екзаменаційна відомість:

```
<gradeReport id="120851">
<date>10-06-2008</date>
<subject>Computer Science Fundamentals</subject>
<examiner>prof.Litvinov</examiner>
  <gradeList>
        <gradeRecord id="1">
          <student>Ivanov</student>
          <grade>4</grade>
        </gradeRecord>
        <gradeRecord id="2">
          <student>Petrov</student>
          <grade>3</grade>
        </gradeRecord>
        <gradeRecord id="3">
          <student>Sidorov</student>
          <grade>5</grade>
        </gradeRecord>
  </gradeList>
</gradeReport>
```

Опишіть структуру даного документа за допомогою *DTD* схеми і *XML Schema*. Перевірте обидві схеми за допомогою відповідних програм-валідаторів.

Лабораторна робота №12.

Тема «Програмна обробка XML документів за допомогою XML DOM.»

Мета роботи:

1. Ознайомлення з основними принципами XML DOM.
2. Ознайомлення з методами програмної обробки XML документів шляхом маніпулювання вузлами дерева документа.

Теоретичний матеріал

XML DOM визначає об'єкти і властивості всіх XML елементів і методи (інтерфейс) для доступу до них. Інакше кажучи, XML DOM описує яким чином необхідно отримувати, змінювати, додавати і видаляти XML елементи.

У відповідності з моделлю DOM все, що міститься всередині XML документа – є вузлом. Тобто XML документ представляється у вигляді дерева вузлів, якими є елементи, атрибути та текст.

Оскільки структури HTML і XML документів дуже схожі, а HTML DOM і XML DOM є частинами більш загального стандарту DOM, то і багато аспектів HTML DOM легко переносяться в XML DOM. Тому основну увагу буде приділено саме специфічним особливостям саме XML DOM. Рекомендується попереднє ознайомлення з лабораторною роботою 3.

1. Структурний аналіз (парсинг) XML

Всі сучасні браузери мають вбудовані XML аналізатори (парсери) для читання і обробки XML. Аналізатор зчитує XML документ, розміщує його в пам'яті і перетворює в XML DOM об'єкт, доступний для мов програмування. Всі приклади тут наведені на JavaScript.

Є деякі відмінності між аналізаторами в Microsoft і в інших браузерах. Перший підтримує як завантаження XML файлів, так і текстових рядків, що містять XML код, в той час як в інших браузерах використовуються роздільні аналізатори. При цьому всі аналізатори мають функції для переміщення по дереву XML документа, доступу, вставки і видалення вузлів в дереві.

Розглянемо приклад завантаження XML об'єктів (файлів і рядків) за допомогою XML аналізатора Microsoft.

```
xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
xmlDoc.async="false";
xmlDoc.load("timetable.xml");
```

У першому рядку програми створюється порожній об'єкт XML документа Microsoft. Далі для запобігання роботи сценарію до повного завантаження документа прапор асинхронності встановлюється в «false». У третьому рядку міститься інструкція завантажити XML файл *«timetable.xml»*.

У наступному приклад відбувається завантаження рядки з XML кодом для подальшого аналізу.

```
xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
xmlDoc.async="false";
xmlDoc.loadXML(txt);
```

Слід звернути на різницю між методами *load()* і *loadXML()* за їх призначенням.

Зауваження. Сучасні браузери не допускають міждоменні звернення до файлів з міркувань безпеки, тобто сама веб-сторінка (з програмним кодом) і XML файл повинні фізично знаходитися на одному сервері. В іншому випадку браузер видасть повідомлення про помилку доступу.

Нижче наведені також кросплатформені реалізації завантаження XML *файлу* і XML *рядка* відповідно.

```
<html>
<body>
<script type="text/javascript">
try //Internet Explorer
  {
          xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
  }
catch(e)
  {
  try //Firefox, Mozilla, Opera, etc.
    {
            xmlDoc=document.implementation.createDocument("","",null);
    }
  catch(e) {alert(e.message)}
  }
try
  {
          xmlDoc.async=false;
          xmlDoc.load("timetable.xml");
          document.write("xmlDoc is loaded, ready for use");
  }
catch(e) {alert(e.message)}
</script>
</body>
</html>
```

```
<html>
<body>
<script type="text/javascript">
text="<timetable>";
text=text+"<lesson>";
text=text+"<timeFrom>08.00</timeFrom>";
text=text+"<subject>Deutsch</subject>";
```

```
text=text+"<teacher>Borisova</teacher>";
text=text+"</lesson>";
text=text+"/<timetable>";

try //Internet Explorer
 {
        xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
        xmlDoc.async="false";
        xmlDoc.loadXML(text);
 }
catch(e)
 {
 try //Firefox, Mozilla, Opera, etc.
  {
         parser=new DOMParser();
         xmlDoc=parser.parseFromString(text,"text/xml");
  }
 catch(e) {alert(e.message)}
 }
document.write("xmlDoc is loaded, ready for use");
</script>
</body>
</html>
```

2. Програмний інтерфейс XML DOM

В рамках DOM моделі **XML** можна розглядати як безліч вузлових об'єктів. Доступ до них здійснюється за допомогою *JavaScript* або інших мов програмування. Програмний інтерфейс DOM включає в себе набір стандартних *властивостей* і *методів*.

Властивості представляють деякі сутності (наприклад, <day>), а *методи* – дії над ними (наприклад, додати <lesson>).

У XML DOM використовуються практично ті ж властивості і методи, що і в HTML DOM.

Наприклад, результатом виконання наступного нижче JavaScript коду буде текстовий зміст елемента <subject> у файлі *timetable.xml.*

txt = xmlDoc.getElementsByTagName ("subject") [0] .childNodes [0] .nodeValue;

Результат: *«Deutsch»*.

В рамках DOM XML можливі 3 способи доступу до вузлів:

1. За допомогою методу *getElementsByTagName* (name). При цьому повертаються всі вузли з вказаним ім'ям тегу (у вигляді індексованого списку). Перший елемент в списку має нульовий індекс.
2. Шляхом обходу вузлів дерева з використанням циклічних конструкцій.

3. Шляхом переміщення по дереву з використанням відносин між вузлами.

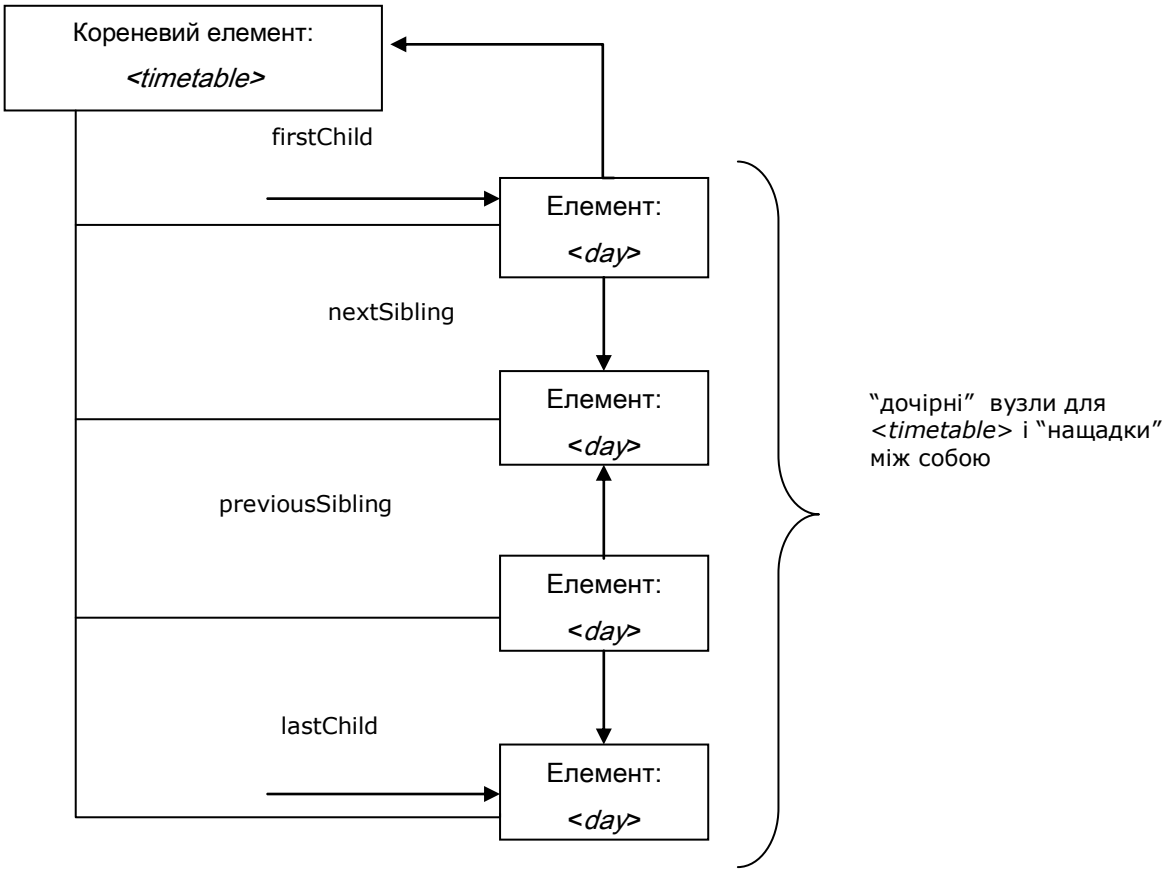
Для визначення довжини списку вузлів використовується властивість *length*.

3. Переміщення між вузлами дерева

У XML DOM відносини між вузлами визначені у вигляді наступних властивостей вузлів:

- parentNode;
- childNodes;
- firstChild;
- lastChild;
- nextSibling;
- previousSibling.

Характер відносин між вузлами представлений на наступному малюнку:



4. Ігнорування порожніх текстових вузлів

Firefox і деякі інші браузери сприймають невідображаємі символи як текстові вузли (на відміну від *Internet Explorer*). Така ситуація призводить до проблем при використанні властивостей *firstChild*, *lastChild*, *nextSibling*, *previousSibling*. Для того, щоб ігнорувати такі порожні текстові вузли можна використовувати наступний прийом:

```
function get_nextSibling(n)
{
        y = n.nextSibling;
        while (y.nodeType!=1)
         {
                y = y.nextSibling;
         }
        return y;
}
```

Оскільки вузли *елементів* мають тип 1, то в тому випадку, коли вузол-нащадок не є вузлом *елемента*, відбуватиметься переміщення до наступного вузла доти, поки не буде знайдений вузол *елементу*.

5. Зміна значення атрибута

Вузли атрибутів можуть брати текстові значення. Зміна цього значення реалізується або через метод *setAttribute ()*, або через властивість вузла атрибуту *nodeValue*

Метод *setAttribute* () змінює значення існуючого атрибуту або створює новий атрибут.

Наприклад:

```
xmlDoc = loadXMLDoc("timetable.xml");
x = xmlDoc.getElementsByTagName('lesson');
x[0].setAttribute("type","lab");
```

Властивість *nodeValue* можна використовувати для зміни значення атрибуту вузла:

```
xmlDoc = loadXMLDoc("timetable.xml");
x = xmlDoc.getElementsByTagName("lesson")[0];
y = x.getAttributeNode("type");
y.nodeValue = "lab";
```

Видалення вузла з дерева реалізується за допомогою методу *removeChild()*:

```
xmlDoc=loadXMLDoc("timetable.xml ");
y = xmlDoc.getElementsByTagName("lesson")[0];
xmlDoc.documentElement.removeChild(y);
```

6. Властивості об'єкту Node

IE: Internet Explorer, ***F***: Firefox, ***O***: Opera, **W3C**: (Стандарт).

Властивість	**Опис**	**IE**	**F**	**O**	**W3C**
		Версія			

baseURI	Повертає абсолютний URI вузла	Ні	1	Ні	Так
childNodes	Повертає властивість NodeList (список дод.вузлів)	5	1	9	Так
firstChild	Повертає перший дочірній вузол	5	1	9	Так
lastChild	Повертає останній дочірній вузол	5	1	9	Так
localName	Повертає локальну частину імені вузла	Ні	1	9	Так
namespaceURI	Повертає URI вузла в просторі імен	Ні	1	9	Так
nextSibling	Повертає наступний дочірній вузол	5	1	9	Так
nodeName	Повертає ім'я вузла в залежності від типу	5	1	9	Так
nodeType	Повертає тип вузла	5	1	9	Так
nodeValue	Встановлює або повертає значення вузла в залежності від типу	5	1	9	Так
ownerDocument	Повертає кореневий елемент (об'єкт document) для вузла	5	1	9	Так
parentNode	Повертає батьківський вузол	5	1	9	Так
prefix	Встановлює або повертає префікс простору імен вузла	Ні	1	9	Так
previousSibling	Повертає безпосередньо передуючий вузол	5	1	9	Так
textContent	Встановлює або повертає текстовий зміст сайту	Ні	1	Ні	Так
xml	Повертає XML код вузла	5	Ні	Ні	Ні

7. Методи об'єкту Node

Метод	Опис	IE	F	O	W3C
appendChild()	Додати новий вузол в кінець списку дочірніх вузлів	5	1	9	Так
cloneNode()	Клонування вузла	5	1	9	Так
compareDocumentPosition()	Порівняння позицій двох вузлів	Ні	1	Ні	Так
getFeature(feature,version)	Повертає об'єкт DOM, який реалізує спеціалізований API			Ні	Так
getUserData(key)	Повертає об'єкт, що асоціюється з ключем поточного вузла. Перед цим об'єкт повинен бути асоційований з поточним вузлом шляхом виклику *setUserData* з тим же ключем			Ні	Так
hasAttributes()	Повертає істинне значення, якщо вузол має атрибути	Ні	1	9	Так

hasChildNodes()	Повертає істинне значення, якщо вузол має дочірні вузли	5	1	9	Так
insertBefore ()	Вставляє новий вузол перед існуючим вузлом	5	1	9	Так
isDefaultNamespace(URI)	Визначає, чи є зазначений *namespaceURI* значенням за замовчуванням			Ні	Так
isEqualNode()	Перевіряє рівність двох вузлів	Ні	Ні	Ні	Так
isSameNode()	Перевіряє ідентичність двох вузлів	Ні	1	Ні	Так
isSupported(feature,version)	Визначає – чи підтримується зазначена характеристика вузлом			9	Так
removeChild()	Видаляє дочірній вузол	5	1	9	Так
replaceChild()	Замінює дочірній вузол	5	1	9	Так
setUserData(key,data,handler)	Асоціює об’єкт з ключем у вузлі			Ні	Так

Порядок виконання лабораторної роботи

При виконанні даної лабораторної роботи потрібно XML-документ *timetable.xml*, який використовувався в *лабораторній роботі 11*:

```
<?xml version="1.0"?>
<timetable>
	<day dayOfWeek="Monday">
		<lesson type="practical">
			<timeFrom>08.00</timeFrom>
			<timeTo>09.30</timeTo>
			<subject>Deutsch</subject>
			<teacher>Borisova</teacher>
			<room>216</room>
		</lesson>
		<lesson type="lecture">
			<timeFrom>09.40</timeFrom>
			<timeTo>11.10</timeTo>
			<subject>SAP Administration</subject>
			<teacher>Egorov</teacher>
			<room>384</room>
		</lesson>
		<lesson  type="practical">
			<timeFrom>11.20</timeFrom>
			<timeTo>12.50</timeTo>
			<subject>SAP Administration</subject>
			<teacher>Petrov</teacher>
			<room>384</room>
		</lesson>
```

```
        </day>
</timetable>
```

1. Створення JavaScript сценарію завантаження XML-документу.

Створіть текстовий файл *loadxmldoc.js*, що містить опис функції завантаження XML-документу:

```
function loadXMLDoc(dname)
{
try //Internet Explorer
  {
          xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
  }
catch(e)
  {
  try //Firefox, Mozilla, Opera, etc.
    {
            xmlDoc=document.implementation.createDocument("","",null);
    }
  catch(e) {alert(e.message)}
  }
try
  {
          xmlDoc.async=false;
          xmlDoc.load(dname);
          return(xmlDoc);
  }
catch(e) {alert(e.message)}
return(null);
}
```

і збережіть його в тій же папці, де знаходиться файл *timetable.xml*.

Код виклику цієї функції може виглядати наступним чином:

```
<html>
<head>
<script type="text/javascript" src="loadxmldoc.js">
</script>
</head>

<body>
<script type="text/javascript">
        xmlDoc=loadXMLDoc("timetable.xml");
        document.write("xmlDoc is loaded, ready for use");
</script>
</body>
```

```
</html>
```

2. Переміщення по дереву вузлів.
Підготуйте наступну HTML сторінку:

```
<html>
<head>
<script type="text/javascript" src="loadxmldoc.js">
</script>
</head>
<body>
<script type="text/javascript">
        xmlDoc = loadXMLDoc("timetable.xml");
        x = xmlDoc.getElementsByTagName("subject");
        for (i=0; i<x.length; i++)
         {
                document.write(x[i].childNodes[0].nodeValue);
                document.write("<br />");
         }
</script>
</body>
</html>
```

Після завантаження сторінки в браузері можна буде побачити наступний результат:

```
Deutsch
SAP Administration
SAP Administration
```

3. Зміна значення елемента.
Наступний приклад демонструє зміну значення елемента <*subject*>:

```
xmlDoc=loadXMLDoc("timetable.xml");
x=xmlDoc.getElementsByTagName("subject")[0].childNodes[0];
x.nodeValue="Java programming";

x = xmlDoc.getElementsByTagName("subject");
for (i=0; i<x.length; i++)
{
        document.write(x[i].childNodes[0].nodeValue);
        document.write("<br />");
}
```

Внесіть відповідні зміни в попередню сторінку і завантажте її в браузері.

4. Переміщення по вузлах дерева з використанням відносин між ними.

Наступний код показує, як використовуючи відносини *firstChild* і *nextSibling* можна отримати для поточного вузла список його дочірніх вузлів:

```
x = xmlDoc.getElementsByTagName("lesson")[0].childNodes;
y = xmlDoc.getElementsByTagName("lesson")[0].firstChild;

for (i = 0; i < x.length; i++)
{
        if (y.nodeType == 1)
          {
                document.write(y.nodeName + "<br />");
          }
        y=y.nextSibling;
}
```

Внесіть необхідні зміни в *html* сторінку і завантажте її в браузері.

Контрольне завдання

У наведеному нижче XML документі описана екзаменаційна відомість:

```
<gradeReport id="120851">
<date>10-06-2008</date>
<subject>Computer Science Fundamentals</subject>
<examiner>prof.Litvinov</examiner>
  <gradeList>
        <gradeRecord id="1">
          <student>Ivanov</student>
          <grade>4</grade>
        </gradeRecord>
        <gradeRecord id="2">
          <student>Petrov</student>
          <grade>3</grade>
        </gradeRecord>
        <gradeRecord id="3">
          <student>Sidorov</student>
          <grade>5</grade>
        </gradeRecord>
  </gradeList>
</gradeReport>
```

1. Використовуючи методи DOM XML, сформуйте HTML сторінку, що містить таблицю з трьох стовпців: *номер, студент, оцінка*.

2. Використовуючи методи DOM XML, замініть цифрові значення оцінок їх словесними еквівалентами, наприклад «4» на «good».

Лабораторна робота №13.

Тема «Форматування і перетворення XML-документу за допомогою XSL. XSLT перетворення XML-документу.»

Мета роботи:

1. Знайомство з методами форматування і перетворення XML документів на основі XSLT перетворень.

Теоретичний матеріал

XSLT можна визначити наступним чином:

- XSLT позначає *XSL Transformations.*
- XSLT є найважливішою частиною XSL перетворення.
- XSLT дозволяє перетворювати один XML в інший XML документ.
- XSLT використовує XPath для переміщення по структурі XML документа.
- XSLT є W3C рекомендацією.

XSLT використовується для перетворення XML-документу в інший XML-документ або в документ іншого розпізнаваного браузерами типу, наприклад HTML або XHTML. Зазвичай XSLT робить це, перетворюючи кожен XML елемент у відповідний йому (X) HTML елемент.

За допомогою XSLT можна додавати або видаляти елементи і атрибути в результуючому документі. Також можливе перегрупування і сортування елементів, фільтрація елементів при відображенні і багато іншого.

Фактично XSLT перетворює вихідне XML дерево в результуюче XML дерево.

XSLT використовує XPath для пошуку інформації в XML документі, тобто XPath є інструментів навігації по елементах і атрибутам XML документів.

У процесі перетворення XSLT використовує XPath для пошуку частин вихідного документа, що відповідають одному або більше заданих шаблонів. Коли відповідність знайдено, XSLT перетворює знайдену частину вихідного документа у відповідну частину результуючого документа.

XSLT є W3C рекомендацією з листопада 1999 року.

Переважна більшість браузерів мають підтримку XML і XSLT.

Internet Explorer

Починаючи з 6 версії, *Internet Explorer* підтримує XML, простори імен, CSS, XSLT і XPath. Версія 5 не сумісна з офіційною *W3C XSL Рекомендацією*.

Mozilla Firefox

Починаючи з версії 1.0.2, Firefox підтримує XML і XSLT (CSS).

Mozilla

Mozilla містить *Expat for XML* парсер підтримує відображення XML + CSS. Також має підтримку простору імен. Реалізує XSLT перетворення.

Netscape

Починаючи з версії 8, *Netscape* використовує ядро на основі *Mozilla*, і тому має таку ж підтримку XML / XSLT.

Opera

Починаючи з версії 9, *Opera* підтримує XML і XSLT (CSS). Версія 8 підтримує тільки XML + CSS.

1. Оголошення XSL

Кореневим елементом, що вказує на те, чи документ є XSL таблицею стилів є:

<Xsl: stylesheet>

або повністю рівноцінний йому

<Xsl: transform>

Відповідно до W3C XSLT Рекомендацією, коректний спосіб оголошення таблиці стилів XSL виглядає наступним чином:

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

або

```
<xsl:transform version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

Для того щоб XSLT елементи, атрибути та характеристики були доступні на початку документа необхідно оголосити простір імен XSLT:

xmlns: xsl = "http://www.w3.org/1999/XSL/Transform"

що вказує на офіційний простір імен W3C XSLT. При цьому також слід вказати атрибут *version = "1.0"*.

2. Реалізація перетворення за допомогою сценарію.

XSLT перетворення з XML в XHTML, що виконується самими браузерами на основі таблиці стилів XSL не завжди є бажаним, оскільки може підтримуватися не усіма браузерами.

Використання як альтернативи JavaScript дозволяє:

- Виконувати перевірку типу браузера.
- Використовувати відповідні таблиці стилів залежно від типу браузера і потреб користувачів.

Іншим рішенням для браузерів, що не підтримують XSLT є перетворення XML в XHTML на веб-сервері.

Порядок виконання лабораторної роботи

1. Реалізація XSLT перетворення XML документа в XHTML засобами браузера.

1) Створіть файл *ttable.xml* такого змісту:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="ttable.xsl"?>
<timetable>
   <lesson>
    <timeFrom>09.30</timeFrom>
    <timeTo>11.00</timeTo>
    <subject>Deutsch</subject>
    <teacher>Borisova</teacher>
   </lesson>
   <lesson>
    <timeFrom>11.10</timeFrom>
    <timeTo>12.20</timeTo>
    <subject>SAP Administration</subject>
    <teacher>Petrov</teacher>
   </lesson>
   <lesson>
    <timeFrom>12.40</timeFrom>
    <timeTo>14.00</timeTo>
    <subject>SAP Administration</subject>
    <teacher>Ivanov</teacher>
   </lesson>
   <lesson>
    <timeFrom>14.00</timeFrom>
    <timeTo>15.20</timeTo>
    <subject>Wen-technology</subject>
    <teacher>Loktev</teacher>
   </lesson>
</timetable>
```

2) Підготуйте для нього відповідний файл таблиці стилів *(ttable.xsl)*:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<xsl:stylesheet                                                    version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
 <html>
 <body>
   <h2>My academical timetable</h2>
   <table border="1">
```

```
   <tr bgcolor="#a5abfa">
    <th align="left">Time From</th>
    <th align="left">Subject</th>
    <th align="left">Teacher</th>
   </tr>
   <xsl:for-each select="timetable/lesson">
   <tr>
    <td><xsl:value-of select="timeFrom"/></td>
    <td><xsl:value-of select="subject"/></td>
    <td><xsl:value-of select="teacher"/></td>
   </tr>
   </xsl:for-each>
   </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Перший рядок у файлі *ttable.xml* виду:

<? Xml-stylesheet type = "text / xsl" href = "ttable.xsl" ?>

являє собою посилання на відповідну таблицю стилів.

У даному прикладі для опису шаблону був використаний елемент <*xsl: template*>.

Атрибут *match* застосовується для зв'язування XML елемента з шаблоном. Значення атрибута match є вираз *XPath*. В даному випадку *match* = "/" вказує на весь документ.

Зміст елемента <*xsl:template*> описує фрагмент HTML коду у вихідному документі. Елемент <*xsl:value-of*> може бути використаний для вилучення значення XML елементу і додавання його у вихідний потік при перетворенні. Значення атрибута *select* є вираженням *XPath*, яке нагадує нотацію, що використовується в файловій системі; знак (/) вказує на звернення до підкаталогу.

Елемент <*xsl:for-each*> дозволяє обирати кожен елемент XML з великої кількості вузлів.

3) При наявності відповідного браузера можна буде побачити наступний результат після завантаження файлу *ttable.xml:*

My academical timetable

Time From	Subject	Teacher
09.30	Deutsch	Borisova
11.10	SAP Administration	Petrov
12.40	SAP Administration	Ivanov
14.00	Wen-technology	Loktev

Для фільтрації виведення елементів можна додати атрибут елемента <*xsl: for-each*>, що задає критерій відбору елементів. Наприклад:

<*Xsl: for-each select* = "*timetable / lesson [subject* = '*SAP Administration*']">

Перевірте роботу даного фільтра.

Допустиме використання наступних операторів для опису фільтру:
- = (дорівнює);
- ! = (не дорівнює);
- & lt; (менше ніж);
- & gt; (більше ніж).

2. Реалізація перетворення за допомогою *JavaScript*.
1) Підготуйте *html* файл, що містить код:

```
<html>
        <head>
          <script>
                function loadXMLDoc(fname)
                {
                    var xmlDoc;
// Код для IE
                    if (window.ActiveXObject)
                     {
                        xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
                     }
// Код для Mozilla, Firefox, Opera и др.
                    else if (document.implementation
                    && document.implementation.createDocument)
                     {
                        xmlDoc=document.implementation.createDocument("","",null);
                     }
                    else
                     {
                        alert('Your browser cannot handle this script');
                     }
                    xmlDoc.async=false;
                    xmlDoc.load(fname);
                    return(xmlDoc);
                }

                function displayResult()
                {
                    xml=loadXMLDoc("ttable.xml");
                    xsl=loadXMLDoc("ttable.xsl");
// Код для IE
```

```
                    if (window.ActiveXObject)
                     {
                         ex=xml.transformNode(xsl);
                         document.getElementById("example").innerHTML=ex;
                     }
// Код для Mozilla, Firefox, Opera и др.
                    else if (document.implementation
                    && document.implementation.createDocument)
                     {
                       xsltProcessor=new XSLTProcessor();
                       xsltProcessor.importStylesheet(xsl);
                       resultDocument                                    =
                      xsltProcessor.transformToFragment(xml,document);

                      document.getElementById("example").appendChild(resultDocum
                      ent);
                     }
                }
          </script>
        </head>
        <body id="example" onLoad="displayResult()">
        </body>
</html>
```

Функція *loadXMLDoc*() завантажує XML і XSL файли залежно від типу браузера.

Функція *displayResult*() використовується для відображення XML файлу в стилі, що задається XSL файлом. Вона виконує наступні дії:

- завантажує XML і XSL;
- визначає тип браузеру;
- якщо браузер підтримує *ActiveX* об'єкти:
  - за допомогою методу *transformNode*() таблиця стилів XSL застосовується до XML документу;
  - формується тіло поточного документу.
- якщо браузер клієнта не підтримує *ActiveX* об'єкти:
  - створюється новий об'єкт *XSLTProcessor* і в нього імпортується XSL файл;
  - за допомогою методу *transformToFragment*() таблиця стилів XSL застосовується до XML документу;
  - формується тіло поточного документа.

2) Завантажте цей документ у веб-браузері.

3. Реалізація перетворення за допомогою JavaScript

1) Підготуйте файл, що містить наступний код сценарію на мові ASP:

```
<%
'Load XML
```

```
set xml = Server.CreateObject("Microsoft.XMLDOM")
xml.async = false
xml.load(Server.MapPath("ttable.xml"))

'Load XSL
set xsl = Server.CreateObject("Microsoft.XMLDOM")
xsl.async = false
xsl.load(Server.MapPath("ttable.xsl"))

'Transform file
Response.Write(xml.transformNode(xsl))
%>
```

На початку сценарію створюється екземпляр парсеру *Microsoft XML parser* (XMLDOM), і XML файл завантажується в пам'ять. Далі створюється ще один примірник парсеру, і XSL файл завантажується в пам'ять. В останньому рядку виконується еретворення XML файлу з використанням XSL файлу в XHTML, який відправляється назад браузеру.

2) Перевірте роботу сценарію в веб-браузері.

Контрольне завдання

У наведеному нижче XML документі описана екзаменаційна відомість:

```
<gradeReport id="120851">
<date>10-06-2008</date>
<subject>Computer Science Fundamentals</subject>
<examiner>prof.Litvinov</examiner>
  <gradeList>
        <gradeRecord id="1">
          <student>Ivanov</student>
          <grade>4</grade>
        </gradeRecord>
        <gradeRecord id="2">
          <student>Petrov</student>
          <grade>3</grade>
        </gradeRecord>
        <gradeRecord id="3">
          <student>Sidorov</student>
          <grade>5</grade>
        </gradeRecord>
  </gradeList>
</gradeReport>
```

Побудуйте для нього XSL файл, необхідний для XSLT перетворення початкового XML документа в HTML сторінку, що містить заголовок відомості і таблицю оцінок зі стовпцями: *номер, студент, оцінка.*

Лабораторна робота №14.

Тема «Розробка веб-служби в ASP.NET.»

Мета роботи:

1. Знайомство з інструментом розробки ASP.NET веб-служб в середовищі Microsoft Visual Studio.

Порядок виконання лабораторної роботи

Розглянемо як приклад створення за допомогою ASP.NET веб-служби, яка переводить будь-яке ціле десяткове число в один з форматів за вибором: двійковий, вісімковий, десятковий.

1. Створіть новий проект (тип: *ASP Web Service*), наприклад, під іменем *ASPNETCalcWebService.* У результаті буде автоматично згенеровано файл *Service1.asmx.cs.*

Програмна логіка веб-служби буде реалізована на мові C # в CodeBehind файлі *Service1.asmx.cs:*

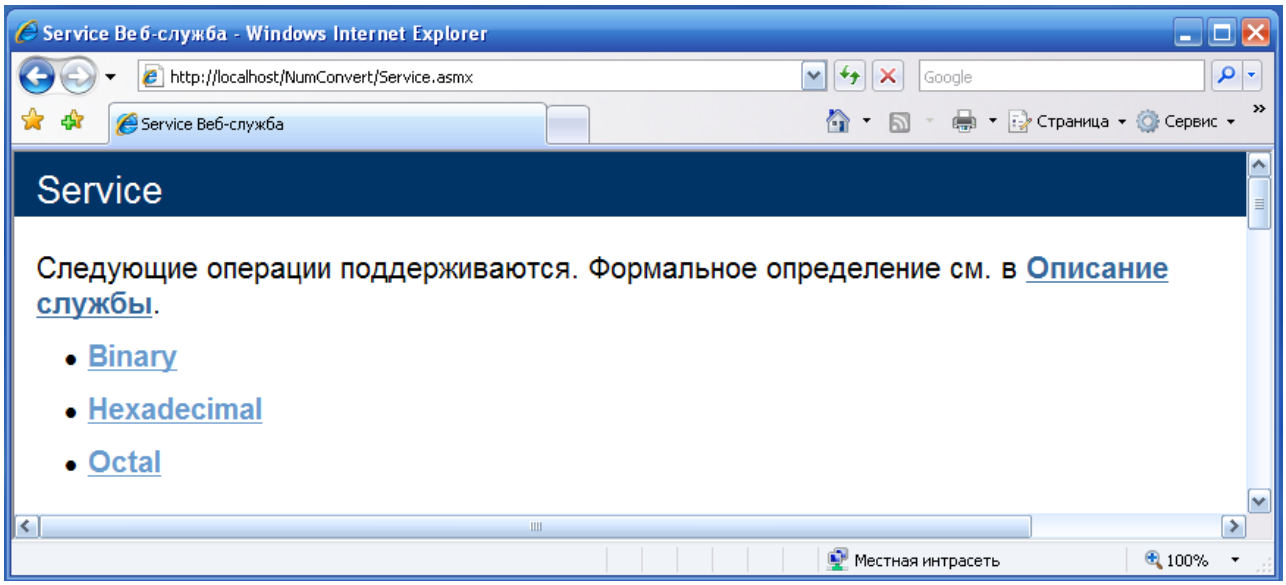
```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;

namespace ASPNETCalcWebService
{
   /// <summary>
   /// Summary description for Service1
   /// </summary>
   [WebService(Namespace = "http://tempuri.org/")]
   [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
   [ToolboxItem(false)]
   public class Service1 : System.Web.Services.WebService
   {
      [WebMethod]
      public string HelloWorld()
      {
         return "Hello World";
      }
   }
}
```

2. Для реалізації логіки веб-служби в цьому файлі, замініть у файлі *Service1.asmx.cs* метод *HelloWorld ()* на 3 нових методи, за допомогою яких будуть виконуватися всі перетворення:

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

namespace ASPNETCalcWebService
{
   /// <summary>
   /// Summary description for Service1
   /// </summary>
   [WebService(Namespace = "http://tempuri.org/")]
   [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
   [ToolboxItem(false)]
   public class Service1 : System.Web.Services.WebService
   {
      //Uncomment the following line if using designed components
      //InitializeComponent();
   }

// Перетворення в двійкову систему числення

   [WebMethod]
   public string Binary(int x) {
      return Convert.ToString(x, 2);
   }

// Перетворення в вісімкову систему числення

   [WebMethod]
   public string Octal(int x)
   {
      return Convert.ToString(x, 8);
   }

// Перетворення в шістнадцяткову систему числення

   [WebMethod]
   public string Hexadecimal(int x)
   {
      return Convert.ToString(x, 16);
   }
}
```

Атрибут *WebMethod* у цьому файлі вказує на те, що описуваний метод повинен бути доступний за протоколом HTTP для користувачів.

3. Відкомпілюйте і запустіть проект. У результаті в браузері повинна з'явитися наступна сторінка:



4. Активуйте гіперпосилання «Опис служби». У вікні браузера має з'явитися опис веб-служби у форматі *WSDL*:

```
<?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://tempuri.org/"
    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://tempuri.org/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified"
        targetNamespace="http://tempuri.org/">
      <s:element name="Binary">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="x"
              type="s:int" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="BinaryResponse">
        <s:complexType>
```

```
            <s:sequence>
                <s:element minOccurs="0" maxOccurs="1"
                name="BinaryResult" type="s:string" />
            </s:sequence>
          </s:complexType>
        </s:element>
        <s:element name="Octal">
          <s:complexType>
            <s:sequence>
              <s:element minOccurs="1" maxOccurs="1" name="x"
                type="s:int" />
            </s:sequence>
          </s:complexType>
        </s:element>
        <s:element name="OctalResponse">
          <s:complexType>
            <s:sequence>
                <s:element minOccurs="0" maxOccurs="1"
                name="OctalResult" type="s:string" />
            </s:sequence>
          </s:complexType>
        </s:element>
        <s:element name="Hexadecimal">
          <s:complexType>
            <s:sequence>
              <s:element minOccurs="1" maxOccurs="1" name="x"
                type="s:int" />
            </s:sequence>
          </s:complexType>
        </s:element>
        <s:element name="HexadecimalResponse">
          <s:complexType>
            <s:sequence>
                <s:element minOccurs="0" maxOccurs="1"
                name="HexadecimalResult" type="s:string" />
            </s:sequence>
          </s:complexType>
        </s:element>
      </s:schema>
    </wsdl:types>
    <wsdl:message name="BinarySoapIn">
      <wsdl:part name="parameters" element="tns:Binary" />
    </wsdl:message>
    <wsdl:message name="BinarySoapOut">
      <wsdl:part name="parameters" element="tns:BinaryResponse" />
    </wsdl:message>
    <wsdl:message name="OctalSoapIn">
```

```
    <wsdl:part name="parameters" element="tns:Octal" />
  </wsdl:message>
  <wsdl:message name="OctalSoapOut">
    <wsdl:part name="parameters" element="tns:OctalResponse" />
  </wsdl:message>
  <wsdl:message name="HexadecimalSoapIn">
    <wsdl:part name="parameters" element="tns:Hexadecimal" />
  </wsdl:message>
  <wsdl:message name="HexadecimalSoapOut">
    <wsdl:part  name="parameters"  element="tns:HexadecimalResponse"
      />
  </wsdl:message>
 <wsdl:portType name="ServiceSoap">
   <wsdl:operation name="Binary">
     <wsdl:input message="tns:BinarySoapIn" />
     <wsdl:output message="tns:BinarySoapOut" />
   </wsdl:operation>
   <wsdl:operation name="Octal">
     <wsdl:input message="tns:OctalSoapIn" />
     <wsdl:output message="tns:OctalSoapOut" />
   </wsdl:operation>
   <wsdl:operation name="Hexadecimal">
     <wsdl:input message="tns:HexadecimalSoapIn" />
     <wsdl:output message="tns:HexadecimalSoapOut" />
   </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="ServiceSoap" type="tns:ServiceSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
   <wsdl:operation name="Binary">
          <soap:operation      soapAction="http://tempuri.org/Binary"
       style="document" />
     <wsdl:input>
       <soap:body use="literal" />
     </wsdl:input>
     <wsdl:output>
       <soap:body use="literal" />
     </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="Octal">
          <soap:operation      soapAction="http://tempuri.org/Octal"
       style="document" />
     <wsdl:input>
       <soap:body use="literal" />
     </wsdl:input>
     <wsdl:output>
       <soap:body use="literal" />
     </wsdl:output>
```

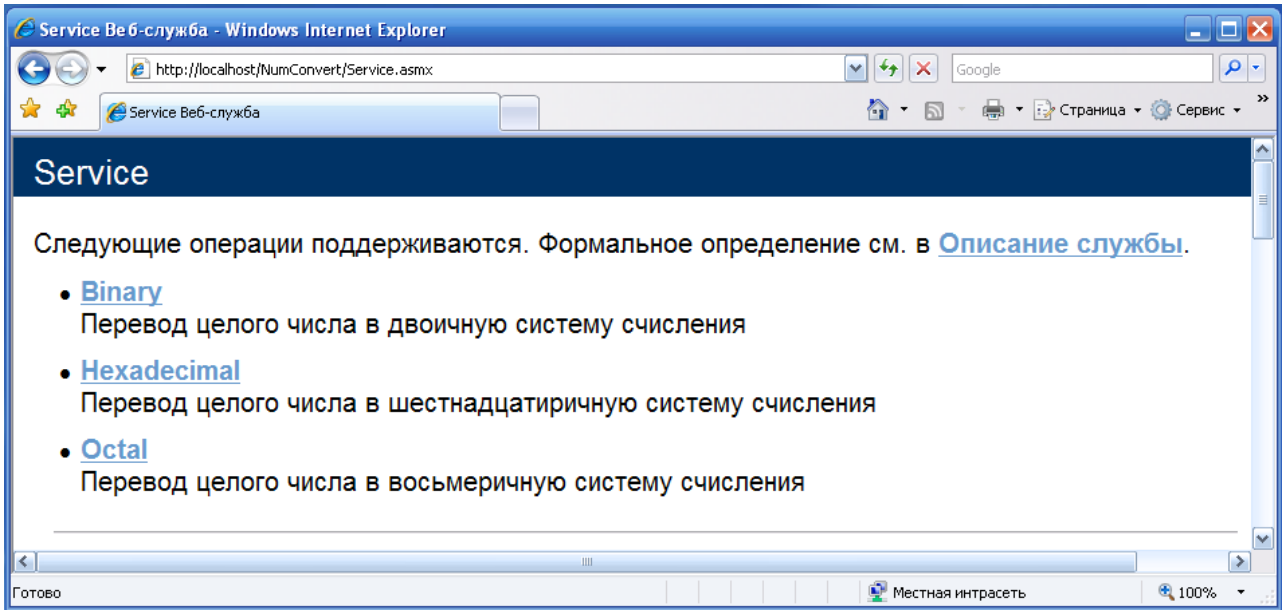
```
      </wsdl:operation>
      <wsdl:operation name="Hexadecimal">
            <soap:operation   soapAction="http://tempuri.org/Hexadecimal"
              style="document" />
        <wsdl:input>
            <soap:body use="literal" />
         </wsdl:input>
        <wsdl:output>
            <soap:body use="literal" />
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:binding name="ServiceSoap12" type="tns:ServiceSoap">
      <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
      <wsdl:operation name="Binary">
            <soap12:operation      soapAction="http://tempuri.org/Binary"
            style="document" />
        <wsdl:input>
            <soap12:body use="literal" />
         </wsdl:input>
        <wsdl:output>
            <soap12:body use="literal" />
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="Octal">
            <soap12:operation      soapAction="http://tempuri.org/Octal"
            style="document" />
        <wsdl:input>
            <soap12:body use="literal" />
         </wsdl:input>
        <wsdl:output>
            <soap12:body use="literal" />
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="Hexadecimal">
          <soap12:operation  soapAction="http://tempuri.org/Hexadecimal"
              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="Service">
      <wsdl:port name="ServiceSoap" binding="tns:ServiceSoap">
```

```
                                                    <soap:address
            location="http://localhost/NumConvert/Service.asmx" />
        </wsdl:port>
        <wsdl:port name="ServiceSoap12" binding="tns:ServiceSoap12">
                                                    <soap12:address
            location="http://localhost/NumConvert/Service.asmx" />
        </wsdl:port>
    </wsdl:service>
        </wsdl:definitions>
```

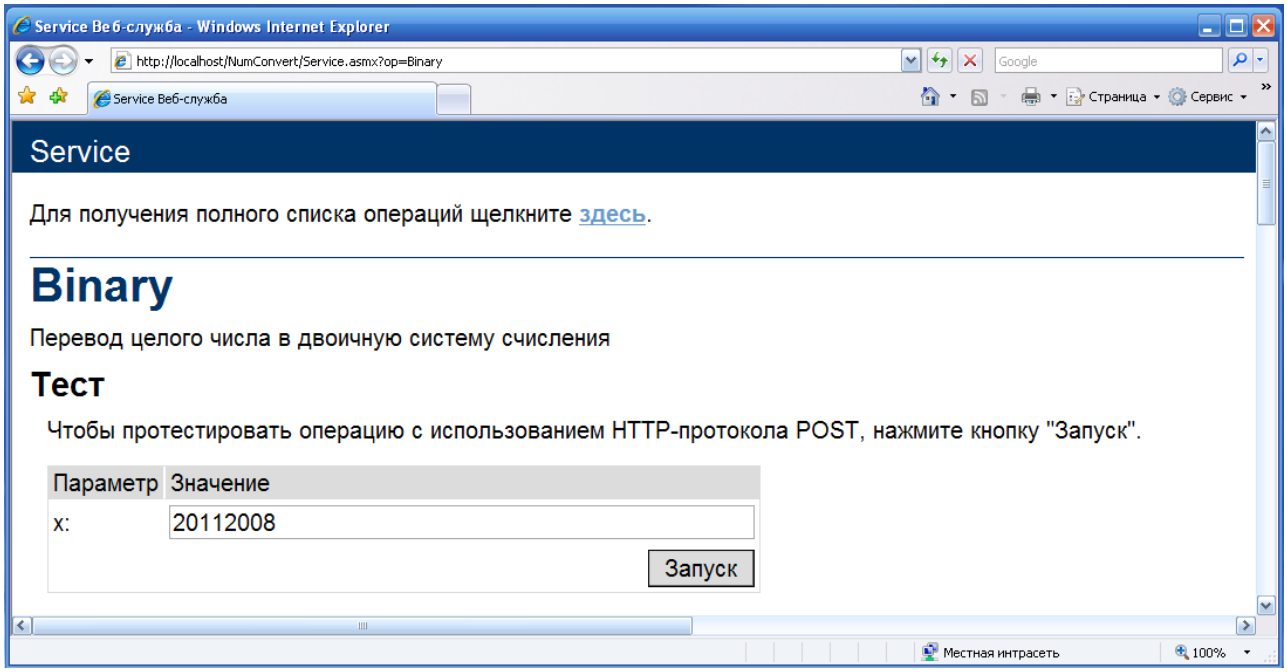
5. Якщо необхідно повідомити більш докладну інформацію для користувачів по кожному з доступних в цій веб-службі методів, достатньо буде додати параметр *Description* в атрибуті *WebMethod*, наприклад:

```
[WebMethod (Description = "Переведення цілого числа в двійкову систему числення")]
public string Binary(int x)
{
        return Convert.ToString(x, 2);
         }
```

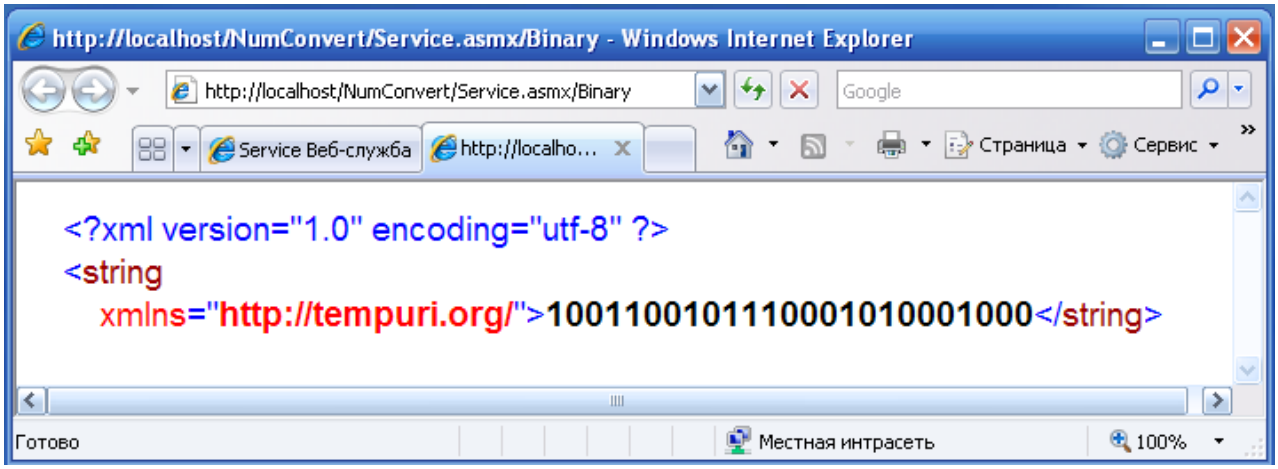
В результаті в браузері буде отримана наступна сторінка:



6. Виберіть метод Binary. При цьому повинна завантажитися веб-сторінка виду:



У результаті роботи веб-служби (після введення числа і натиснення кнопки «Запуск») буде створена наступна сторінка:



Контрольне завдання

Створіть ASP.NET веб-службу, яка буде підтримувати три методи, які повертають відповідно *назву* поточного дня *тижня*, *номер* поточного дня в *місяці*, *номер* поточного дня в *році*.

Для отримання необхідних даних на стороні сервера можна використовувати властивості і методи:

```
Label1.Text = DateTime.Now.DayOfWeek.ToString();
Label1.Text = DateTime.Now.Day.ToString();
Label1.Text = DateTime.Now.DayOfYear.ToString();
```

Лабораторна робота №15.

Тема «Розробка веб-служби в ASP.NET. Створення проксі-збірки для веб-служби.»

Мета роботи:

1. Продовження знайомства з інструментами розробки ASP.NET веб-служб і їх використання за допомогою проксі-збірки в середовищі Microsoft Visual Studio.

Теоретичний матеріал

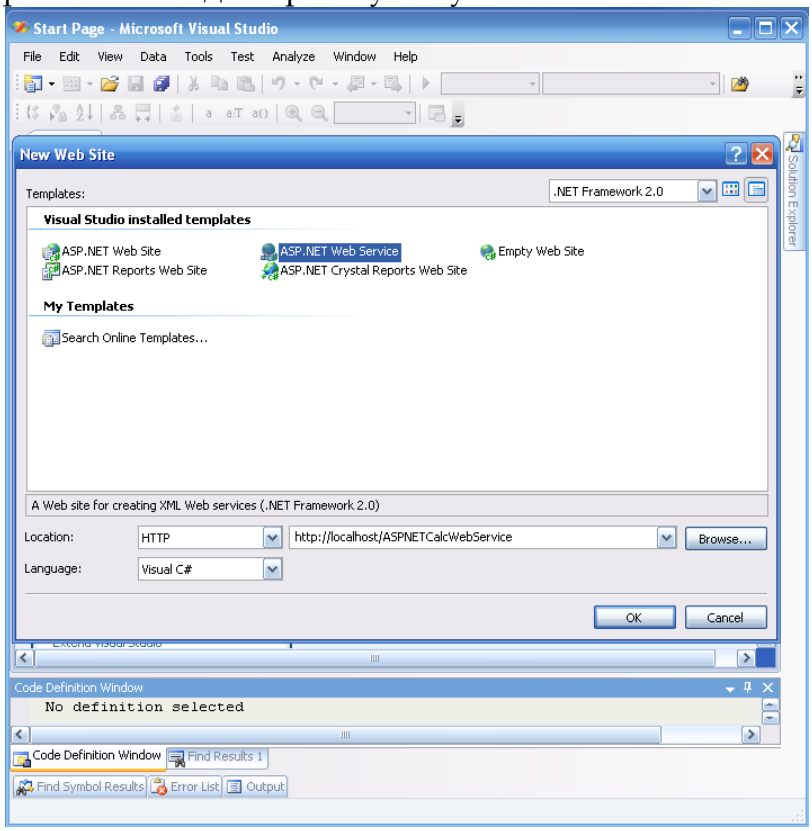
У попередній лабораторній роботі розглядалося створення веб-служби *ASPNETCalcWebService*, що виконувала переклад десяткового цілого числа в системі числення з основою 2, 8, 16. При цьому передача даних клієнтом (по протоколу SOAP) і отримання їх вироблялося в XML форматі. Це може бути незручним для клієнта і зажадати розробку додаткових компонентів для конвертації даних в звичний для кінцевого користувача формат.

Уникнути цього можна, якщо звертатися до веб-службі не безпосередньо, а за допомогою проксі-збірки на C# або іншою мовою програмування. Така проксі-збірка створюється автоматично в Visual Studio.NET, наприклад, у вигляді клієнта *Windows Forms*, і бере на себе всю роботу по взаємодії з веб-службою.

Порядок виконання лабораторної роботи

1. Створення веб-служби.

1) Створіть веб-сайт для проекту типу "*ASP.NET Web Service*":



У складі нового проекту будуть автоматично створені файли *Service.asmx* (містить код подання) і *Service.cs* (містить програмний код).

2) Замініть у файлі *Service.cs* метод *HelloWorld()* на методи, що реалізують всі перетворення (*див. Лабораторну роботу 14*):

```
// Перетворення в двійкову систему числення

    [WebMethod]
    public string Binary(int x) {
        return Convert.ToString(x, 2);
    }

// Перетворення в вісімкову систему числення

    [WebMethod]
    public string Octal(int x)
    {
        return Convert.ToString(x, 8);
    }

// Перетворення в шістнадцяткову систему числення
    [WebMethod]
    public string Hexadecimal(int x)
    {
        return Convert.ToString(x, 16);
    }
```

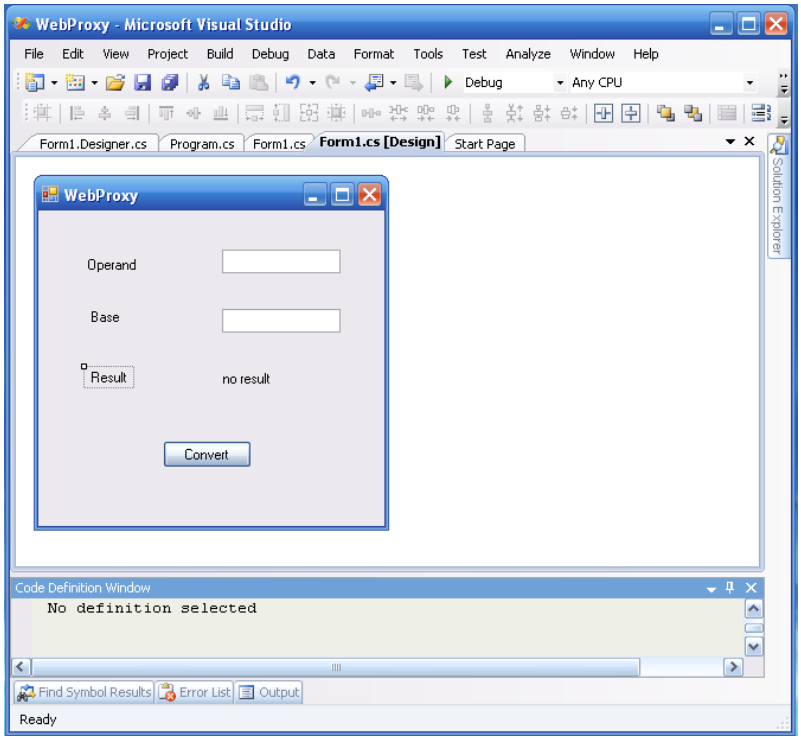
3) Відкомпілюйте проект і виконайте пробний запуск веб-служби. URL служби для завантаження в браузері:

http: //localhost/ASPNETCalcWebService/Service.asmx

2. Створення проксі-збірки.

1) Створіть проект з ім'ям, наприклад *WebProxy* (тип проекту *Windows Forms Application*).

2) За допомогою дизайнера форм MS Visual Studio підготуйте інтерфейс взаємодії з користувачем у вигляді форми з двома текстовими полями введення даних, полем відображення результату і кнопкою:



Код користувальницької форми, автоматично згенерований середовищем MS VS, знаходиться у файлі *Form1.Designer.cs*:

```
namespace WebProxy
{
   partial class WebProxy
   {
      private System.ComponentModel.IContainer components = null;

      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }

      #region Windows Form Designer generated code

      private void InitializeComponent()
      {
         this.label1 = new System.Windows.Forms.Label();
         this.textBox1 = new System.Windows.Forms.TextBox();
         this.textBox2 = new System.Windows.Forms.TextBox();
         this.button1 = new System.Windows.Forms.Button();
         this.label2 = new System.Windows.Forms.Label();
         this.label3 = new System.Windows.Forms.Label();
```

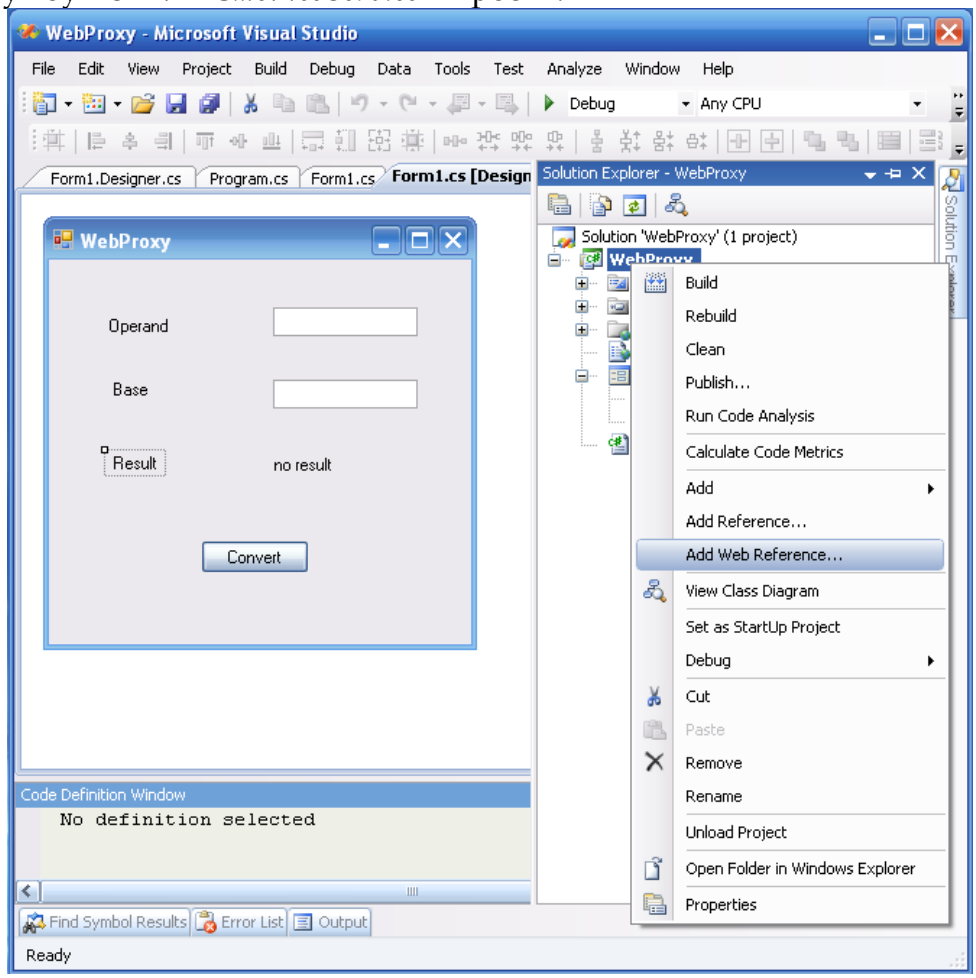
```
            this.label4 = new System.Windows.Forms.Label();
            this.SuspendLayout();
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(152, 135);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(47, 13);
            this.label1.TabIndex = 0;
            this.label1.Text = "no result";
            //
            // textBox1
            //
            this.textBox1.Location = new System.Drawing.Point(155, 33);
            this.textBox1.Name = "textBox1";
            this.textBox1.Size = new System.Drawing.Size(100, 20);
            this.textBox1.TabIndex = 1;
            //
            // textBox2
            //
            this.textBox2.Location = new System.Drawing.Point(155, 83);
            this.textBox2.Name = "textBox2";
            this.textBox2.Size = new System.Drawing.Size(100, 20);
            this.textBox2.TabIndex = 2;
            //
            // button1
            //
            this.button1.Location = new System.Drawing.Point(105, 194);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(75, 23);
            this.button1.TabIndex = 3;
            this.button1.Text = "Convert";
            this.button1.UseVisualStyleBackColor = true;
            this.button1.Click += new System.EventHandler(this.button1_Click);
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(38, 39);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(48, 13);
            this.label2.TabIndex = 4;
            this.label2.Text = "Operand";
            //
            // label3
            //
```

```
            this.label3.AutoSize = true;
            this.label3.Location = new System.Drawing.Point(41, 83);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(31, 13);
            this.label3.TabIndex = 5;
            this.label3.Text = "Base";
            //
            // label4
            //
            this.label4.AutoSize = true;
            this.label4.Location = new System.Drawing.Point(41, 134);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(37, 13);
            this.label4.TabIndex = 6;
            this.label4.Text = "Result";
            //
            // WebProxy
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(292, 266);
            this.Controls.Add(this.label4);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.button1);
            this.Controls.Add(this.textBox2);
            this.Controls.Add(this.textBox1);
            this.Controls.Add(this.label1);
            this.Name = "WebProxy";
            this.Text = "WebProxy";
            this.Load += new System.EventHandler(this.WebProxy_Load);
            this.ResumeLayout(false);
            this.PerformLayout();
        }

        #endregion

        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.TextBox textBox2;
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Label label4;
    }
}
```

3) Додайте веб-посилання:

http://localhost/ASPNETCalcWebService/Service.asmx

на веб-службу *ASPNETCalcWebService* в проект:



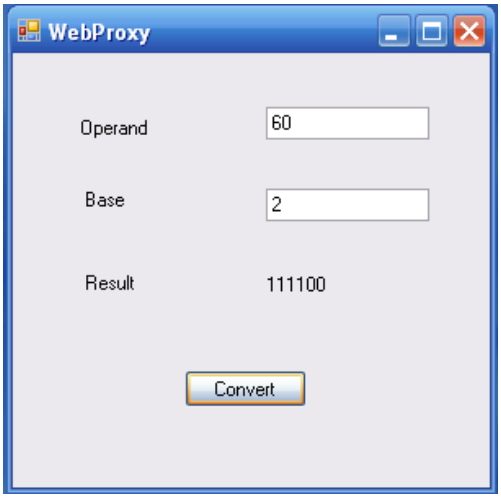
4) Додайте обробник події *«натискання кнопки»* у файлі *Form1.cs*:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace WebProxy
{

// додано простір імен localhost

   using localhost;
   public partial class WebProxy : Form
```

```
    {
        public WebProxy()
        {
            InitializeComponent();
        }

// обробка події натискання кнопки

        private void button1_Click(object sender, EventArgs e)
        {

// створення об'єкта, пов'язаного з веб-службою

            localhost.Service w = new localhost.Service();

            string ans;

// перетворення даних з текстових полів введення в формі в цілі числа
// operand – операнд, NumBase - основа системи числення

            int NumBase = int.Parse(textBox2.Text);
            int operand = int.Parse(textBox1.Text);

// вибір відповідної функції веб-служби та взаємодія з нею

            switch (NumBase)
            {
                case 2: ans = w.Binary(operand); break;
                case 8: ans = w.Octal(operand); break;
                case 16: ans = w.Hexadecimal(operand); break;
                default: ans = "base is undefined"; break;
            }

//  відображення результату обчислення у формі

            label1.Text = ans;
        }

    }
}
```

5) Відкомпілюйте проект.

6) Перевірка роботи програми. На скріншоті нижче показана робота програми *WebProxy* (спільно з веб-службою *ASPNETCalcWebService*):



Контрольне завдання

Створіть ASP.NET проксі-збірку на мові C# для веб-служби, що підтримує три методи, які повертають відповідно *назву* поточного дня *тижня*, номер поточного дня в *місяці*, *номер* поточного дня в *році*.

Для отримання необхідних даних на стороні сервера можна використовувати властивості і методи:

```
Label1.Text = DateTime.Now.DayOfWeek.ToString();
Label1.Text = DateTime.Now.Day.ToString();
Label1.Text = DateTime.Now.DayOfYear.ToString();
```

Вибір методу веб-служби повинен бути реалізований через форму з елементом типу *ListBox* або *RadioButton*.

Лабораторна робота №16.

Тема «Приклади розробки RSS-джерел і RSS-рідерів.»

Мета роботи:

1. Введення в технологію RSS.
2. Вивчення структури RSS документів, їх генерації і публікації.

Теоретичний матеріал

RSS – це спосіб поширення веб-контенту з веб-сайту на інших веб-сайтах. RSS дозволяє виконувати швидкий перегляд новин і змін.

На кого орієнтований формат RSS? В першу чергу RSS призначений для використання на дуже часто оновлюваних веб-сайтах, наприклад:

- Новинні сайти (списки новин).
- Сайти компаній (списки новин і продуктів).
- Календарі (списки майбутніх подій і знаменних днів).
- Зміни сайтів (список оновлених і нових сторінок).

Офіційного стандарту RSS як такого не існує. Реально використовуються наступні формати:

- Порядку 50% RSS-потоків використовують RSS 0.91.
- Порядку 25% використовують RSS 1.0.
- Решта 25% RSS-потоків розподілені між RSS 0.9 і RSS 2.0.

Як працює RSS?

1. Створюється RSS документ у вигляді файлу з розширенням .xml.
2. Цей файл розміщується на веб-сайті.
3. RSS-потік реєструється в RSS-агрегаторах.

1. Приклад RSS-документу

Розглянемо приклад RSS-документа.

```
<?xml version="1.0" encoding="windows-1251" ?>
<rss version="2.0">
<channel>
 <title>My Home Page</title>
 <link>http://www.MyHP.edu</link>
 <description>Free web building tutorials</description>
 <item>
  <title>RSS Demo</title>
  <link>http:// www.MyHP.edu/rss</link>
  <description>New RSS demo on my home page</description>
 </item>
 <item>
  <title>XML Demo</title>
  <link> www.MyHP.edu/xml</link>
  <description>New XML demo on my home page</description>
 </item>
```

```
</channel>
</rss>
```

У першому рядку розміщено оголошення версії XML і кодування документа.

Другий рядок ідентифікує даний документ як RSS документ версії 2.0.

У третьому рядку міститься елемент <channel>, що описує RSS потік. Він у свою чергу містить дочірні елементи:

- <title> – описує заголовок RSS каналу.
- <link> – описує гіперпосилання на канал.
- <description> – містить коротку характеристику каналу.

Кожен елемент <channel> може містити один або більше <item> елементів. Кожен елемент <item> описує окрему статтю RSS джерела. У свою чергу кожен елемент <item> має три обов'язкових дочірніх елементи:

- <title>;
- <link>;
- <description>.

Крім обов'язкових елементів <channel> може містити деякі додаткові дочірні елементи.

Елемент <category> описує категорію потоку і використовується RSS агрегатор для угруповання сайтів на основі категорій.

Елемент <copyright> інформує про авторське право на даний документ.

Елемент <image> призначений для відображення зображення при показі доки агрегатором. У свою чергу він містить три обов'язкові дочірніх елементи:

- <url> – URL зображення;
- <title> – альтернативний текст, що відображається при неможливості показу зображення;
- <link> – містить гіперпосилання на веб-сайт, що містить канал.

Приклад:

```
<image>
 <url>http://www.myhp.edu/images/me.gif</url>
 <title>My home page</title>
 <link>http://www.myhp.edu</link>
</image>
```

Елемент <language> інформує про мову документа. Може бути використаний агрегаторами для угруповання сайтів за мовою.

2. Список дочірніх елементів для елемента <channel>

Елемент	Опис	Статус
<category>	Вказується одна або кілька категорій потоку	Необов'язковий
<cloud>	Реєструє процеси, які отримують негайне повідомлення в разі поновлення потоку	Необов'язковий
<copyright>	Сповіщає про авторське право на	Необов'язковий

	документ	
<description>	Короткий опис каналу	Обов'язковий
<docs>	Визначає URL документів за форматами, використовуваним в потоці	Необов'язковий
<generator>	Вказує на програму, що генерує потік	Необов'язковий
<image>	Дозволяє показувати зображення, що представляє потік в аггрегатор	Необов'язковий
<language>	Вказується мова потоку	Необов'язковий
<lastBuildDate>	Визначається дата останнього оновлення змісту потоку	Необов'язковий
<link>	Вказується гіперпосилання на канал	Обов'язковий
<managingEditor>	Міститься поштову адресу редактора змісту потоку	Необов'язковий
<pubDate>	Визначає дату останньої публікації змісту потоку	Необов'язковий
<rating>	Міститься PICS рейтинг потоку	Необов'язковий
<skipDays>	Вказує дні, в які аггрегатори не повинні оновлювати потік.	Необов'язковий
<skipHours>	Вказується кількість годин протягом яких аггрегатори не повинні оновлювати потік	Необов'язковий
<textInput>	Описується текстове поле введення, яка відображається в потоці	Необов'язковий
<title>	Розміщується ім'я каналу	Обов'язковий
<ttl>	Зазначається кількість хвилин протягом яких потік може знаходиться в кеші до поновлення з джерела	Необов'язковий
<webMaster>	Поміщається адресу електронної пошти веб-дизайнера потоку	Необов'язковий

3. Список дочірніх елементів для елемента <item>

Елемент	Опис	Статус
<author>	Адреса електронної пошти автора теми	Необов'язковий
<category>	Описує одну або більше категорій для даної теми	Необов'язковий
<comments>	Дозволяє зв'язати тему з коментарями з даної теми	Необов'язковий
<description>	Короткий опис теми	Обов'язковий
<enclosure>	Дозволяє включити мультимедіа файл в дану тему	Необов'язковий
<guid>	Містить унікальний ідентифікатор теми	Необов'язковий
<link>	Містить гіперпосилання на тему	Обов'язковий
<pubDate>	Містить останню дату публікації теми	Необов'язковий
<source>	Вказує зовнішнє джерело для даної теми	Необов'язковий
<title>	Назва теми	Обов'язковий

4. Публікація RSS файлу

Створення RSS потоку не обмежується розробкою RSS документа. Необхідно ще опублікувати цей файл. Для цього буде потрібно виконати наступну послідовність дій:

1. Вибрати відповідну назву для RSS файлу. Розширення повинно бути *.xml*.
2. Перевірити RSS файл на правильність за допомогою відповідної програми-валідатора, наприклад, взятої за адресою *http://www.feedvalidator.org*.
3. Розмістити RSS файл у відповідному веб-каталозі веб-сервера.
4. Скопіювати одну з «кнопок»: RSS або XML у веб-каталог.
5. Вставити обрану «кнопку» на вихідну сторінку RSS потоку у вигляді гіперпосилання на RSS файл, наприклад:

```
<a href = "www.myhp.edu/rss/myrss.xml">
      < img src="www.myhp.edu/rss/rss.gif" width="35" height="15">
</a>
```

6. Розмістити створений RSS потік в популярних Каталогах RSS потоків, при цьому URL потік повинен посилатися саме на сам XML файл потоку, тобто *http://www.myhp.edu/rss/myrss.xml*.
7. Можна також зареєструвати потік в популярних пошукових системах, наприклад:
   - Yahoo – http://publisher.yahoo.com/rss_guide/submit.php;
   - Google – http://www.google.com/ig;
   - MSN – http://w.moreover.com/site/products/ind/pingserver.html
8. Періодичне оновлення потоку.

В принципі, можна самостійно займатися формуванням даних у відповідному форматі RSS для роботи потоку. Однак є досить зручні високорівневі засоби для автоматизації даної роботи. Наприклад:

- **MyRSSCreator** – http://www.myrsscreator.com/;
- **FeedFire** – http://www.feedfire.com/site/index.html.

Для читання RSS потоків використовуються спеціальні програми читання – RSS рідери. Деякі з браузерів мають вбудовані RSS рідери. Зокрема, веб-браузер *MS Internet Explorer 7* може інтерпретувати XML файли RSS потоків і коректно їх відображати.

5. Додавання RSS-каналу за допомогою Microsoft Internet Explorer версії

При використанні MS Internet Explorer версії 7 і Office Outlook 2007 можна додавати та переглядати RSS-канали за допомогою будь-якої з цих програм.

1. Під час перегляду веб-сторінки, що містить RSS, в *Internet Explorer* поряд з кнопкою **(Домашня сторінка)** відображається кнопка .
2. Натисніть кнопку .
3. На веб-сторінці з'явиться список доступних RSS-каналів.

4. Виберіть RSS-канал, який необхідно додати.

Можна також натиснути кнопки  , 

або 

, які розташовані на веб-сторінці.

Порядок виконання лабораторної роботи

1. Створення RSS документа за допомогою PHP сценарію і з використанням інтерфейсу DOM XML.

1) Підготуйте файл, що містить сценарій на мові PHP:

```
<?php
// Створюється новий XML документ.
$dfeed = new DOMDocument('1.0', 'utf-8');

// Создание корневого элемента  <rss>
$erss = $dfeed->createElement('rss');
$erss->setAttribute('version', '2.0');

// І додавання його в дерево документа
$dfeed->appendChild($erss);

// Створення елемента <channel> і додавання його до <rss>
$echannel = $dfeed->createElement('channel');
$erss->appendChild($echannel);

// Створення і додавання в channel
// вузлів <title>, <link>, <description>, <language>, <pubDate>
$echannel->appendChild( $dfeed->createElement('title', 'RSS-channel title') );
$echannel->appendChild( $dfeed ->createElement('link', 'http://www.myhp.edu') );
$echannel->appendChild( $dfeed ->createElement('description', 'my RSS demo') );
$echannel->appendChild( $dfeed ->createElement('language', 'en') );
$echannel->appendChild( $dfeed ->createElement('pubDate', date('r')) );

// Додавання до вузла <channel> 2 вузлів <item>
for ( $i = 1; $i <= 2; $i++ )
{
   $eitem = $dfeed->createElement('item');
   $echannel->appendChild($eitem);

   // Створення дочірніх елементів для <item>
   $eitem->appendChild( $dfeed->createElement('title', 'Item'.$i) );
   $eitem->appendChild(                                    $dfeed->createElement('link',
'http://www.myhp.edu/rss/'.$i.'.xml') );
   $eitem->appendChild( $dfeed->createElement('description', 'Description for '.$i.'
item') );
}
```

```
// Збереження документа у файлі demo.rss
$dfeed->save('demo.rss');
?>
```

2) Створений файл розмістіть на веб-сервері, налаштуйте права доступу (право на запис) для веб-сервера до директорії, в якій розміщується сценарій, або до спеціальної директорії, в якій буде створений файл *demo.rss* (це більш безпечно).

3) Виконайте сценарій за запитом з веб-браузера.

4) У результаті виконання сценарію вийде наступний документ:

```
<?xml version="1.0" encoding="utf-8" ?>
<rss version="2.0">
   <channel>
      <title>RSS-channel title</title>
      <link>http://www.myhp.edu</link>
      <description>my RSS demo</description>
      <language>en</language>
      <pubDate>Thu, 06 Nov 2008 01:50:53 +0300</pubDate>
     <item>
        <title>Item1</title>
        <link>http://www.myhp.edu/rss/1.xml</link>
        <description>Description for 1 item</description>
      </item>
     <item>
        <title>Item2</title>
        <link>http://www.myhp.edu/rss/2.xml</link>
        <description>Description for 2 item</description>
      </item>
   </channel>
 </rss>
```

2. Читання RSS документа за допомогою PHP сценарію і з використанням інтерфейсу DOM XML.

1) Підготуйте файл, що містить сценарій на мові PHP:

```
<?php
// Завантаження документа
$dfeed = DOMDocument::load('demo.rss');

// Читання елемента <channel>
$echannel = $dfeed->firstChild->firstChild;

// Читання першого дочірнього елемента вузла <channel> т.е.< title>
$cchild = $echannel->firstChild;

// Виведення вмісту дочірніх елементів вузла <channel> до елемента <item>:
```

```
print "<strong><br>";
while ( $cchild->tagName != 'item' )
{
   print $cchild->nodeValue. "<br>";
   $cchild = $cchild->nextSibling;
}
print "</strong><br>";

// Отримання списку вузлів <item>
$litems = $echannel->getElementsByTagName('item');

print "<br>";
foreach ( $litems as $eitem )
{
        $title = $eitem->firstChild->nodeValue;
        $link = $eitem->firstChild->nextSibling->nodeValue;
        $descr = $eitem->lastChild->nodeValue;

        print "<a href=".$link.">".$title."</a> :: ".$descr."<br>";
}
print "<br>";

?>
```

У методі *load()*в значені параметра вкажіть шлях до RSS-файлу.

2) Створений файл розмістіть на веб-сервері, налаштуйте права доступу (право на запис) для веб-сервера до директорії, в якій розміщується сценарій, або до спеціальної директорії, в якій буде створений файл demo.rss (це більш безпечно).

3) Виконайте сценарій за запитом з веб-браузера.

4) Результат повинен виглядати наступним чином:

RSS-channel title
http://www.myhp.edu
my RSS demo
en
Thu, 06 Nov 2008 02:15:55 +0300

Item1 :: Description for 1 item
Item2 :: Description for 2 item

Контрольне завдання

Розробіть PHP-сценарій, що генерує RSS документ, що містить опис веб-сторінок на вашому локальному веб-сайті. Як джерело інформації для сценарію можна використовувати або заздалегідь підготовлений текстовий файл, або функції PHP з бібліотеки для роботи з файловою системою (див., Наприклад: *http://www.softtime.ru/bookphp/help.php*)

Лабораторна робота №17.

Тема «Реалізація асинхронної взаємодії веб-браузера з веб-сервером за допомогою технології AJAX.»

Мета роботи:

1. Ознайомлення з принципами асинхронної взаємодії між веб-клієнтом і веб-сервером в рамках технології AJAX.

Теоретичний матеріал

AJAX = Asynchronous JavaScript and XML.

AJAX – не є новою мовою програмування, але технологія створення покращених, більш швидких і більшою мірою інтерактивних веб-додатків.

JavaScript сценарій за допомогою AJAX може безпосередньо взаємодіяти з сервером за допомогою об'єкту *XMLHttpRequest*. Використання даного об'єкту, обмін даними з веб-сервером можуть відбуватися без перезавантаження сторінки.

AJAX дозволяє веб-сторінкам запитувати невеликі обсяги інформації з сервера ніж цілком всю сторінку в результаті асинхронної передачі даних (в рамках HTTP протоколу) між браузером і сервером.

AJAX не залежить від програмного забезпечення веб-сервера і заснований на наступних веб-стандартах:

- JavaScript;
- XML;
- HTML;
- CSS.

Оскільки ці веб-стандарти чітко визначені і мають підтримку в найбільш поширених веб-браузерах, то AJAX додатки є браузеро- і платформо-незалежними.

Популярність AJAX пов'язана з появою сервісу *Google Suggest* в 2005 році. Даний сервіс на основі об'єкту *XMLHttpRequest* надає в розпорядження користувача досить динамічний веб-інтерфейс. У процесі введення символів користувачем в поле пошукового запиту JavaScript відправляє їх на сервер і отримує від нього список підказок.

Об'єкт *XMLHttpRequest* підтримується в *Internet Explorer* (починаючи з 5 версії і вище), *Safari 1.2, Mozilla 1.0 / Firefox, Opera 8+ і Netscape 7*.

Порядок виконання лабораторної роботи

У даній лабораторній роботі розглядається приклад системи, що імітує роботу сервісу *Google Suggest* на основі AJAX.

1. Реалізація клієнтської частини.

Передбачається, що користувач може вводити в текстове поле форми назву автомобільної марки, отримуючи при цьому динамічно список варіантів назв, відповідних вже введеним символам, без перезавантаження сторінки.

1) Створіть наступну веб-сторінку:

```
<html>
```

```
<head>
   <script src="chint.js"></script>
</head>
        <body>
           <form>
              First Name:
              <input type="text" id="txt1" onkeyup="showHint(this.value)">
           </form>
           <p>Suggestions: <span id="txtHint"></span></p>
        </body>
</html>
```

Як видно з коду, при настанні події *onkeyup* (відтискання клавіші) викликається обробник *showHint()*.

2) У файлі *chint.js* збережіть наступний код обробника:

```
var xmlHttp;
function showHint(str)
{
        if (str.length==0)
          {
                document.getElementById("txtHint").innerHTML="";
                return;
          }
        xmlHttp=GetXmlHttpObject();
        if (xmlHttp==null)
          {
                alert ("Your browser does not support AJAX!");
                return;
          }
        var url = "ghint.php";
        url = url + "?q=" + str;
        url = url + "&sid=" + Math.random();
        xmlHttp.onreadystatechange = stateChanged;
        xmlHttp.open("GET", url, true);
        xmlHttp.send(null);
}
function stateChanged()
{
        if (xmlHttp.readyState==4)
        {
                document.getElementById("txtHint").innerHTML                    =
                xmlHttp.responseText;
        }
}
function GetXmlHttpObject()
{
```

```
        var xmlHttp=null;
        try
         {
                // Firefox, Opera 8.0+, Safari
                xmlHttp = new XMLHttpRequest();
         }
        catch (e)
         {
                // Internet Explorer
                try
                 {
                        xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
                 }
                catch (e)
                 {
                        xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
                 }
         }
        return xmlHttp;
}
```

З коду видно, що кожен раз, коли вводиться символ, викликається функція-обробник. Якщо при цьому зміст текстового поля форми непорожній (*str.length> 0*), функція виконує наступні дії:

- Формується *url* для відправки веб-серверу.
- Додається значення параметра *q*, рівне вмісту текстового поля, до *url.*
- Додається до *url* випадкове число для запобігання кешування.
- Створюється об'єкт *XMLHTTP*, при цьому вказується функція (*stateChanged*) підлягає виконанню при настанні події введення символу.
- Відкривається об'єкт *XMLHTTP* з вказаним значенням *url*.
- Відправляється HTTP запит веб-серверу.

Якщо поле введення пусте, відбувається очищення змісту розділу *txtHint* на веб-сторінці.

Ключовим моментом в даній системі є використання об'єкту *XMLHttpRequest*.

Даний об'єкт по-різному створюється в різних браузерах. Так, *Internet Explorer* для цього використовує *ActiveXObject*, в той час як інші браузери використовують вбудований в JavaScript об'єкт *XMLHttpRequest*.

Для підтримки роботи системи в різних браузерах використаний оператор *«try-catch».*

- Спочатку робиться спроба створити об'єкт *XMLHttpRequest* для браузерів *Firefox, Opera* або *Safari*: *xmlHttp = new XMLHttpRequest().*
- У разі невдачі, робиться наступна спроба створення об'єкту для *Internet Explorer 6.0+*: *xmlHttp = new ActiveXObject ("Msxml2.XMLHTTP").*

- Якщо це також не вдається, то робиться спроба створення об'єкту вже для *Internet Explorer 5.5+*: *xmlHttp = new ActiveXObject ("Microsoft.XMLHTTP").*
- У випадку, якщо жодна з цих спроб не принесла успіху, видається повідомлення про відсутність підтримки AJAX браузером.

2. Реалізація серверної частини.

Розмістіть на веб-сервері у файлі з ім'ям *ghint.php* наступний PHP сценарій:

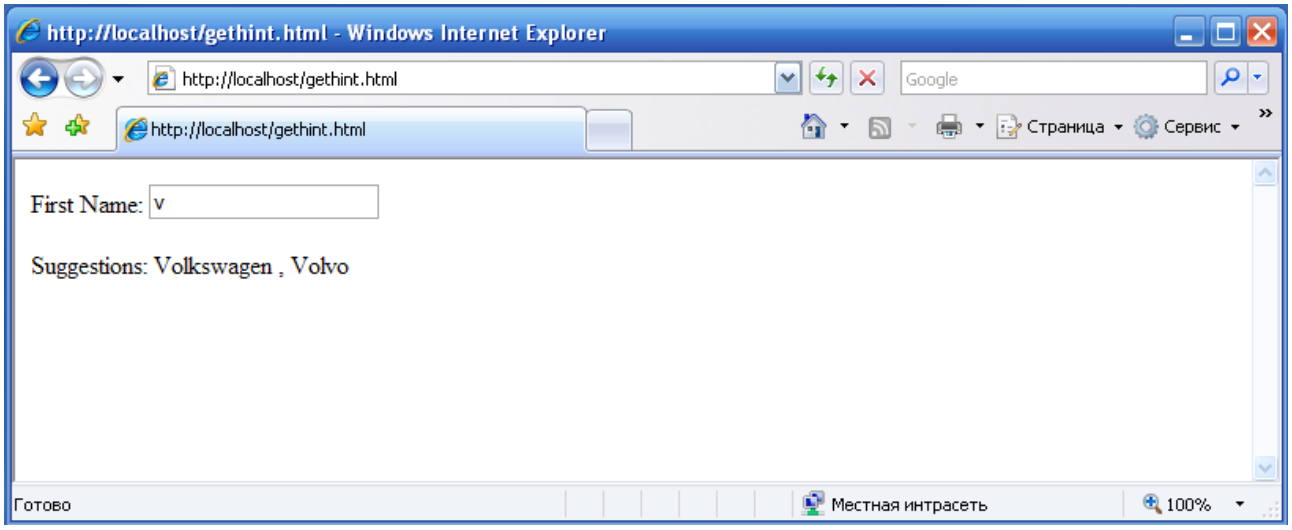
```
<?php
	header("Cache-Control: no-cache, must-revalidate");
	 // Минулий дата
	header("Expires: Mon, 1 Sep 2008 07:30:00 GMT");
	// Ініціалізація масиву назв
	$a[]="Audi";
	$a[]="BMW";
	$a[]="Buick";
	$a[]="Chevrolet";
	$a[]="Citroen";
	$a[]="Dodge";
	$a[]="Ferrari";
	$a[]="Fiat";
	$a[]="Ford";
	$a[]="Honda";
	$a[]="Hyundai";
	$a[]="Cherokee";
	$a[]="Cherry";
	$a[]="Lada";
	$a[]="Lamborghini";
	$a[]="Lincoln";
	$a[]="Mazda";
	$a[]="Mercedes";
	$a[]="Mitsubishi";
	$a[]="Nissan";
	$a[]="Opel";
	$a[]="Peugeot";
	$a[]="Plymoth";
	$a[]="Pontiac";
	$a[]="Renault";
	$a[]="Rover";
	$a[]="Saab";
	$a[]="Subaru";
	$a[]="Suzuki";
	$a[]="Toyota";
	$a[]="Volkswagen";
	$a[]="Volvo";
	// отримання параметра q з URL
	$q = $_GET["q"];
```

```
        // пошук відповідників з масиву якщо довжина q > 0
        if (strlen($q) > 0)
        {
          $hint = "";
          for($i = 0; $i<count($a); $i++)
          {
                  if (strtolower($q) == strtolower(substr($a[$i],0,strlen($q))))
                    {
                            if ($hint == "")
                              {
                                      $hint=$a[$i];
                              }
                            else
                              {
                                      $hint=$hint." , ".$a[$i];
                              }
                    }
          }
        }
        // Повернення рядка "немає варіантів" якщо відповідностей, не знайдено
        // або знайдене відповідність
        if ($hint == "")
        {
                $response = "no suggestion";
        }
        else
        {
                $response = $hint;
        }
        // висновок результату
        echo $response;
?>
```

3. Перевірка працездатності системи.

Перевірте за допомогою відповідного веб-браузера роботу системи.

На скріншоті показана робота в браузері з веб-сторінкою *ghint.html*, що використовує AJAX:



Контрольне завдання

З використанням PHP-сценарію №7 з лабораторної роботи №5 розробіть модифікований варіант калькулятора для чотирьох арифметичних операцій, в якому відбувається асинхронна передача даних між браузером і сервером (за допомогою AJAX).

Лабораторна робота №18.

Тема «Створення мобільних веб-додатків за допомогою ASP.NET.»

Мета роботи:

1. Демонстрація деяких можливостей розробки мобільних веб-додатків за допомогою керуючих елементів ASP.NET Mobile.

Теоретичний матеріал

У даній лабораторній роботі демонструється застосування керуючих елементів ASP.NET Mobile для створення веб-додатків для мобільних пристроїв.

.NET Mobile представляє собою розширення Microsoft ASP.NET і Microsoft .NET Framework. По суті, .NET Mobile представляє собою набір серверних керуючих елементів для форм, що орієнтовані на застосування в бездротових мобільних пристроях. Ці елементи управління генерують різний код для різних пристроїв на мовах WML, HTML або Compact HTML (cHTML).

Керуючі елементи ASP .NET Mobile розширюють функції SDP (Smart Device Programmability) і платформу .NET Compact Framework. Вони дозволяють використовувати можливості платформи .NET Compact Framework і середовища розробки Visual Studio .NET для створення мобільних веб-додатків завдяки можливості доставки даних на різноманітні мобільні пристрої за допомогою технології ASP.NET. Цей підхід дозволяє створити в середовищі Visual Studio .NET єдиний мобільний веб-додаток, що автоматично виконує формування даних для відображення на різноманітних пристроях: мобільних телефонах, пейджерах, смартфонах, Pocket PC і ін. При цьому інтегроване середовище розробки дозволяє створювати мобільні веб-додатки, просто перетягуючи керуючі елементи на дизайнер форм.

Система .NET Mobile не встановлює жодних компонентів на клієнтський пристрій. Для адаптації форматування під конкретні браузери використовується серверна логіка. Вона генерує дані у форматах WML, HTML, і cHTML. Крім адаптивної генерації веб-сторінок, технологія .NET Mobile надає багатий набір засобів індивідуалізації та розширення, забезпечуючи простий спосіб здійснення підтримки нових пристроїв. Крім того, технологія .NET Mobile дозволяє розробникам керувати поданням даних для конкретного пристрою або класу пристроїв в одній програмній моделі і забезпечує легку реалізацію підтримки нових пристроїв без переробки існуючих веб-додатків.

Загальна послідовність роботи *.NET Mobile* виглядає наступним чином:

- *Мобільний пристрій*. Він посилає запит на завантаження веб-сторінки.
- *Інтернет.* Запит передається через мережу відповідному веб-серверу.
- *Веб-сервер* IIS отримує запит від мобільного пристрою і передає його обробнику.
- *.NET Framework* виконує обробку запиту.
- *ASP.NET* компілює запитуваний пристроєм документ.
- *.NET Mobile.* Реалізує елементи веб-сторінки з урахуванням вимог конкретного типу мобільного пристрою.
- *Веб-сторінка* повертається назад до клієнтського пристрою.

Найбільш поширеними елементами управління форм для мобільних пристроїв є наступні:

Название	**Функция**
Command	Выполнение действия
Form	Контейнер для элементов управления
Image	Изображение
Label	Текст
Link	Гиперссылка
List	Список
MobilePage	Контейнер для элементов управления
ObjectList	Список объектов
Panel	Контайнер для других элементов управления
SelectionList	Список с выбором
StyleSheet	Стили для других элем ентов управления
TextBox	Поле для ввода однострочного текста
TextView	Поле для ввода многострочного текста

Порядок виконання лабораторної роботи

Для роботи з прикладами, які приводилися в даній лабораторній роботі крім середовища розробки Visual Studio .NET потрібне встановлення компонентів .NET Mobile і WAP емулятора.

В якості WAP емулятора був використаний *Series 40 5th Edition SDK* (на основі *Nokia 6267*). Для встановлення емулятора необхідно завантажити з сайту *Series 40 5th Edition SDK* (необхідна попередня реєстрація) і встановити програму *Nokia Mobile Internet Toolkit* і сам емулятор *Nokia Series 40 5th Edition SDK.*

Загальна послідовність розробки веб-додатку з використанням .NET Mobile виглядає наступним чином:

- Створення проекту веб-додатку ASP.NET.
- Підключення інтерфейсу System.Mobile.UI.
- Вставка в веб-сторінку керуючих елементів .NET Mobile.

Розглянемо приклад створення найпростішої веб-сторінки, що відображає на екрані мобільного пристрою повідомлення «Hello world!».

1. Створіть новий проект типу **ASP.NET Web Application.**

2. Активуйте за допомогою меню вікно «*Solution Explorer*». Зі списку файлів проекту видаліть файли *Default.aspx.cs* і *Default.aspx.designer.cs.*

3. Сторінка *Default.aspx* повинна містити наступний код:

```
<%@ Page Inherits= "System.Web.UI.MobileControls.MobilePage " Language= "C#
"%>
<%@ Register TagPrefix= "mobile" Namespace= "System.Web.UI.MobileControls "
Assembly= "System.Web.Mobile " %>
```

```
<mobile:Form runat="server">
  <mobile:Label runat="server">Hello world!</mobile:Label>
</mobile:Form>
```

Директива в першому рядку повідомляє оброблювачу ASP про необхідність обробки даної сторінки, яка призначена для мобільних пристроїв.

Директива у другому рядку реєструє префікс «mobile» для звернення до елементів управління для мобільних пристроїв.

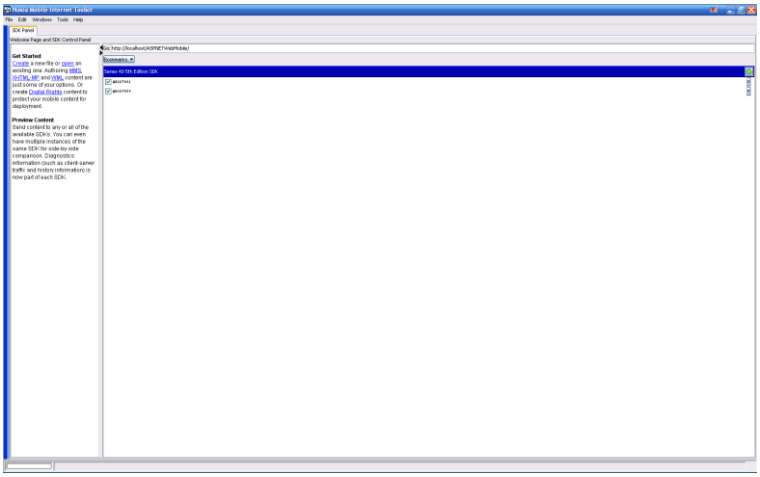
Тег <mobile: Form> вказує серверу на необхідність вставки форми.

Тег <mobile: Label> вказує серверу на необхідність вставки мітки з текстом «Hello world!».

4. Відкомпілюйте проект.

5. Створіть новий віртуальний каталог, який пов'язаний з директорією поточного проекту. Додайте до списку документів за замовчуванням сторінку *Default.aspx*.

6. Запустіть емулятор *Nokia Series 40 5th Edition SDK* і програму *Nokia Mobile Internet Toolkit*. У програмі *Nokia Mobile Internet Toolkit* вкажіть повний URL створеної за допомогою ASP.NET сторінки *Default.aspx*.



7. Після обробки даної сторінки для мобільного пристрою, що підтримує протокол WAP, буде згенерована сторінка, вид якої в емуляторі наведено нижче:

8. У емуляторі виберіть розділ меню «Tools> Diagnostics». У вікні діагностики, що з'явилося можна буде побачити наступний код сторінки, яка завантажена в емуляторі:

```
<?xml version='1.0'?>
<!DOCTYPE wml PUBLIC '-//WAPFORUM//DTD WML 1.1//EN'
'http://www.wapforum.org/DTD/wml_1.1.xml'><wml><head>
<meta http-equiv="Cache-Control" content="max-age=0" />
</head>
<card>
<p>Hello world!</p></card>
</wml>
```

Наступний приклад демонструє використання простого елементу форми для введення тексту (в режимі пароля):

```
<%@ Page Inherits= "System.Web.UI.MobileControls.MobilePage " Language= "C#
"%>
<%@ Register TagPrefix= "mobile " Namespace= "System.Web.UI.MobileControls "
Assembly= "System.Web.Mobile " %>

<script runat="server" language="C#">

string uname;

protected void AgeClick(Object sender, EventArgs e)
{
   uname=text1.Text;
   ActiveForm=form2;
}

protected void  Form2_Activate(Object sender, EventArgs e)
{
  message.Text="Wellcome " + uname + "!";
}
</script>

<Mobile:Form id="form1" runat="server">
<Mobile:Label runat="server">Your name, please</Mobile:Label>
<Mobile:TextBox runat="server" id="text1" password="true"/>
<Mobile:Command runat="server" OnClick="AgeClick" Text="Submit" />
</Mobile:Form>

<Mobile:Form id="form2" runat="server" OnActivate="Form2_Activate">
<Mobile:Label runat="server" id="message" />
</Mobile:Form>
```

Якщо виконати як у попередньому прикладі всі кроки з 1 по 8, то після завантаження отриманої сторінки в емуляторі на екрані відобразиться перша форма:

Після введення рядка і натискання кнопки *«Submit»* активується друга форма:

Код сторінки для першої форми, що завантажений в мобільний пристрій, виглядає наступним чином:

```
<?xml version='1.0'?>
<!DOCTYPE wml PUBLIC '-//WAPFORUM//DTD WML 1.1//EN'
'http://www.wapforum.org/DTD/wml_1.1.xml'><wml><head>
<meta http-equiv="Cache-Control" content="max-age=0" />
</head>
<card>
<do type="prev" label="PќP°P·P°Pr"><prev /></do>
        <p>Your name, please<br/>
        <input name="mcsvprtws0" type="password" value="" />
                <anchor title="P'PїPµCЂPµPr">Submit
                        <go href="Default.aspx?__ufps=058985" method="post">
                                <postfield name="__EVENTTARGET" value="ctl01" />
                                <postfield name="text1" value="$(mcsvprtws0)" />
```

```
                    </go>
                </anchor>
        </p>
</card>
         </wml>
```

Контрольне завдання

Розробіть за допомогою MS Visual Studio .NET серверний веб-додаток, що використовує керуючий ASP.NET Mobile елемент: <*Mobile: Calendar*>.

Додайте для цього елемента обробник події *OnSelectionChanged*, який повинен відображати вибраний користувачем мобільного пристрою день календаря за допомогою елемента <*Mobile: Label*>.

Код обробника події може виглядати наступним чином:

```
<script runat="server">

protected void CalChanged(Object sender, EventArgs e)
{
   label1.Text = "You selected: " + calendar1.SelectedDate;
}
</script>
```

Створіть необхідний віртуальний каталог на веб-сервері і перевірте роботу програми за допомогою емулятора мобільного пристрою.

ЛІТЕРАТУРА

1. Беллиньясо М. Разработка Web-приложений в среде ASP.NET 2.0: с примерами на C# / М. Беллиньясо. – М. : «Диалектика», 2007. – 640 с.
2. Дунаев В. В. JavaScript / В. В. Дунаев. – СПб. : «Питер», 2003. – 394 с.
3. Крейн Д. AJAX в действии: технология – Asynchronous JavaScript and XML = AJAX in Action / Д. Крейн, Э. Паскарелло, Д. Джеймс. – М. : «Вільямс», 2006. –640 с.
4. Мельтцер К. Разработка CGI-приложений на Perl / К. Мельтцер. – М. : «Вильямс», 2001. – 395 с.
5. Ньюкомер Э. Веб-сервисы: XML, WSDL, SOAP и UDDL / Э. Ньюкомер. – СПб. : Питер, 2003. – 345 с.
6. Роджерс Д. Программирование на Microsoft JScript.NET / Д. Роджерс. – М. : «Вильямс», 2002. – 352 с.
7. Салмре И. Программирование мобильных устройств на платформе .NET Compact Framework / И. Салмре. – М. : «Вильямс», 2006. – 736 с.
8. Старыгин А. XML: разработка Web-приложений / А. Старыгин. – СПб. : «БХВ-Петербург», 2003. – 585 с.
9. Троелсен Э. C# и платформа .NET. Библиотека программиста / Э. Троелсен. – СПб. : «Питер», 2007. – 796 с.
10. Форта Б. Освой самостоятельно регулярные выражения. 10 минут на урок / Б. Форта. – М. : «Вильямс», 2005. – 184 с.
11. Храмцов П. Б. Основы WEB-технологий. / П. Б. Храмцов, С. А. Брик, А. М. Русак, А. И. Сурин – М. : ИТУИТ.РУ, 2003. – 512 с.
12. Шапошников И. PHP 5.1: учебный курс / И. Шапошников. – СПб. : «Питер», 2007.– 192 с.
13. Шеперд Д. Освой самостоятельно XML за 21 день / Д. Шеперд. – М. : «Вильямс», 2002. – 432 с.
14. Шорт С. Разработка XML Web-сервисов средствами Microsoft.NET / С. Шорт. – СПб. : «БХВ-Петербург», 2003. – 480 с.

ЗМІСТ

Навчальне видання

Онищенко Сергій Вікторович

Методичні рекомендації
до виконання лабораторних робіт
з дисципліни
«WEB-ТЕХНОЛОГІЇ»

Літературний редактор

Комп'ютерна верстка та дизайн обкладинки

Технічний редактор

Надруковано з оригінал-макету, наданого автором

Підписано до друку
Формат 60x84/16. Папір офсетний.
Гарнітура "Book Antiqua". Друк – лазерний.
Ум.-друк. арк. 8. Наклад 50 прим. Зам. № 08.

Бердянський державний педагогічний університет
Вул. Шмідта 4, м. Бердянськ, 71100
Свідоцтво суб'єкта видавничої справи ДК № 2961 від 05.09.2007 р.