

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторних робіт з курсової дисципліни
«Технологія створення WEB-застосунків»

для студентів денної та заочної форми навчання
спеціальності 122 «Комп'ютерна інженерія»

Затверджено
на засіданні групи забезпечення студентів
Протокол № 15 від 10.05.2021р.
Голова групи _____ Метерков В.І.

Затверджено
на засіданні кафедри _____
Протокол № 5 від 10.05.2021р.
Завідувач кафедри _____ Казисна Н.Ф.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторних робіт з навчальної дисципліни
«Технології створення WEB–застосунків»

для студентів денної та заочної форми навчання
спеціальності 122 «Комп'ютерні науки»

Затверджено
на засіданні групи
забезпечення спеціальності
Протокол № _____
від «____» _____ 2021р.

Одеса – 2021

Методичні вказівки до лабораторних робіт з дисципліни „**Технології створення WEB-застосунків**” для студентів III року навчання денної та заочної форми за спеціальністю 122 «Комп’ютерні науки», рівень вищої освіти бакалавр /Терещенко Т.М., Гнатовська Г.А. – Одеса, ОДЕКУ, 2021.

ЗМІСТ

ВСТУП.....	5
ЛАБОРАТОРНА РОБОТА №1 Основи роботи у середовищі розробки додатків Microsoft Visual Studio 2012	8
ЛАБОРАТОРНА РОБОТА № 2 Створення WEB–форми у середовищі розробки Visual Studio 2012. Робота з масивами, рядками, процедурами і функціям на мові С#.....	21
ЛАБОРАТОРНА РОБОТА № 3. Основи роботи у середовищі Microsoft Visual Studio 2012.....	40
ЛАБОРАТОРНА РОБОТА № 4. Робота з базами даних MS SQL Server в Microsoft Visual Studio 2012	53
ЛАБОРАТОРНА РОБОТА № 5. Вивчення серверних WEB– елементів управління	64
ЛАБОРАТОРНА РОБОТА № 6. Створення дизайну сторінок та системи навігації веб–додатка в Microsoft Visual Studio 2015	76
ЛІТЕРАТУРА.....	100

ВСТУП

Метою дисципліни є ознайомлення студентів з основними принципами, методами та можливостями технологій побудови сучасних інформаційних систем. У курсі передбачається вивчення вибраної технології – платформа Microsoft.NET.

Учебний курс присвячений розробці веб–додатків на платформі .NET Framework, вивчаються технології .NET, ASP.NET, архітектура, конфігурація і розгортання інтернет–систем.

ASP.NET є основною і найбільш повною технологією для веб–розробки з усіх, які коли–небудь створювалися для побудови серверних елементів веб–додатків. Вона є повнофункціональною платформою, що дозволяє створювати складні і надзвичайно швидкі веб–додатки.

У результаті вивчення дисципліни студенти повинні надбати:

знання:

- сучасних інтернет–технологій побудови систем з архітектурою клієнт–сервер;
- архітектури платформи Microsoft.NET;
- бібліотек класів .NET–Framework Class Library;
- основ web–програмування з використанням технології ASP.NET;
- способів та інструментів використання баз даних в web–додатках ASP.NET;
- технології ADO.Net. та технології створення web–форм.

уміння:

- використовувати С# в .NET–додатках;
- використовувати Visual Studio в процесі створення веб–застосунків;
- використовувати технологію веб–форм;
- працювати з технологією ADO.NET та базами даних в веб–додатках ASP.NET.

компетентність:

- знання загально–методичних принципів побудови, технологій реалізації сучасних інформаційних систем з територіально–розподіленою архітектурою.

Практична частина спрямована на вивчення питань створення клієнтських і серверних додатків .NET. Практична частина курсу передбачає

освоєння Visual Studio.NET 2012 та Visual Studio 2015 при створенні WEB-додатків в межах виконання лабораторних робіт, самостійної роботи, в яких запроваджено проектування та реалізація інформаційної інтернет-системи з базою даних.

Дані методичні вказівки до виконання лабораторних робіт містять теоретичні відомості та методику виконання 6 лабораторних робіт з дисципліни, які входять до практичного модулю П1:

Лабораторна робота №1 – Основи роботи у середовищі розробки додатків Microsoft Visual Studio 2012.

Лабораторна робота №2 – Створення WEB-форми у середовищі розробки Visual Studio 2012. Робота з масивами, рядками, процедурами і функціям на мові C#.

Лабораторна робота №3 – Основи роботи у середовищі Microsoft Visual Studio 2012. Класи, структури та колекції C#.

Лабораторна робота №4 – Основи роботи з базами даних MS SQL Server в Microsoft Visual Studio 2012.

Лабораторна робота №5 – Вивчення серверних WEB-елементів управління.

Лабораторна робота № 6 – Створення дизайну сторінок та системи навігації веб-додатка в Microsoft Visual Studio 2015.

Після вивчення ЗМ-П1 студент повинен вміти: налаштовувати Microsoft Visual Studio 2012 та Microsoft Visual Studio 2015 для розробки Web-додатків; використовувати серверні WEB-елементи управління для конструювання дизайну додатку і застосовувати C# для створення обробника подій.

Контролюючим заходом передбаченим для цього змістовного модуля є усне опитування. По кожній лабораторній роботі студент повинен скласти звіт, якій містить в собі:

- назву роботи,
- мету роботи,
- загальне та індивідуальне завдання згідно варіанта,
- послідовний алгоритм розв'язання задачі, який обов'язково ілюструється екранними формами з поясненнями що до виконаних дій,
- текст основних програмних модулів,
- відповіді на контрольні питання. Варіант індивідуального завдання узгоджується з викладачем.

Оформлений звіт захищається студентом усно. Студент повинен чітко і грамотно відповідати на контрольні питання, які оголошені наприкінці кожної лабораторної роботи. Виконана та захищена лабораторна робота оцінюється згідно з умовами, які викладені в силабусі дисципліни.

ПРАВИЛА ТЕХНІКИ БЕЗПЕКИ ТА ОХОРОНА ПРАЦІ

Лабораторні роботи з дисципліни проводяться у лабораторіях кафедри інформаційних технологій або кафедри інформатики, які оснащені комп'ютерною технікою з відповідним програмним забезпеченням. Студенти зобов'язані дотримуватися правил техніки безпеки та правил користування обчислювальною технікою в лабораторіях кафедр.

Згідно з «Правилами техніки безпеки в лабораторіях» студентам забороняється:

- з'являтися та знаходитись приміщенні в нетверезому стані;
- ставити поруч з клавіатурою ємності з рідиною;
- перебувати в приміщенні у верхньому одязі та завалювати ним робочі столи та стільці;
- працювати в лабораторії більше 6-ти годин на день (для вагітних жінок більше 4-х годин);
- за власною ініціативою змінювати закріплені за ними робочі місця та знаходитись в приміщенні під час роботи іншої учбової групи;
- самостійно виконувати вмикання електроживлення лабораторії та заміну складових частин ПК, що вийшли із ладу.

У випадку виявлення несправностей обчислювальної техніки студент повинен сповістити про це викладача чи будь-кого з навчально-допоміжного персоналу лабораторії.

ЛАБОРАТОРНА РОБОТА №1

Основи роботи у середовищі розробки додатків Microsoft Visual Studio 2012

Мета роботи: ознайомлення і вивчення елементарних понять та прийомів роботи в інтегрованому середовищі розробки Microsoft Visual Studio .NET 2012 для створення, документування, запуску та налаштування програм, які написані на мові .NET. – C#.

Постановка завдання: створити в Microsoft Visual Studio 2012 Web–додаток для виконання простих розрахунків з використанням покрокової інструкції до виконання, що наведена в теоретичній частині цієї лабораторної роботи.

Теоретичні відомості

Microsoft Visual Studio 2012 – повнофункціональне середовище розробки Web–додатків, гнучкий і універсальний інструмент проектування і створення закінчених додатків для платформи Windows.

Microsoft Visual Studio 2012 включає засоби управління проектами, редактор початкового тексту, конструктори інтерфейсу користувача, майстри, компілятори, інструменти, утиліти, документацію і відладчик. Вона дозволяє створювати застосування для Windows–платформ, а також для платформи .NET Framework. Одне з найважливіших удосконалень – можливість роботи з різними мовами і додатками різних типів в єдиному середовищі розробки. Оболонка (середовище розробки застосувань – IDE) Microsoft Visual Studio 2012 містить безліч віконних утиліт, що дозволяють отримувати зрізи інформації про проєкт і управляти розробкою додатків в зручній формі.

При завантаженні Microsoft Visual Studio 2012 відображається вікно, що складається з декількох областей (рис. 1.1). Зона вікна «Последние» містить посилання на 6 останніх проєктів.

Розміщена під нею область «Пуск» містить посилання на завдання, що найчастіше виникають у користувача. Клацання по такому посиланню приведе до негайного виконання даного завдання.

Наприклад, клацання по посиланню «Создать проєкт» приведе до негайного створення нового Web–сайта на основі ASP.NET.

Вся зона вікна, що задіяна для відображення наведених вище розділів, надалі використовується для відображення вмісту документа (файлу) і називається вікном документів.

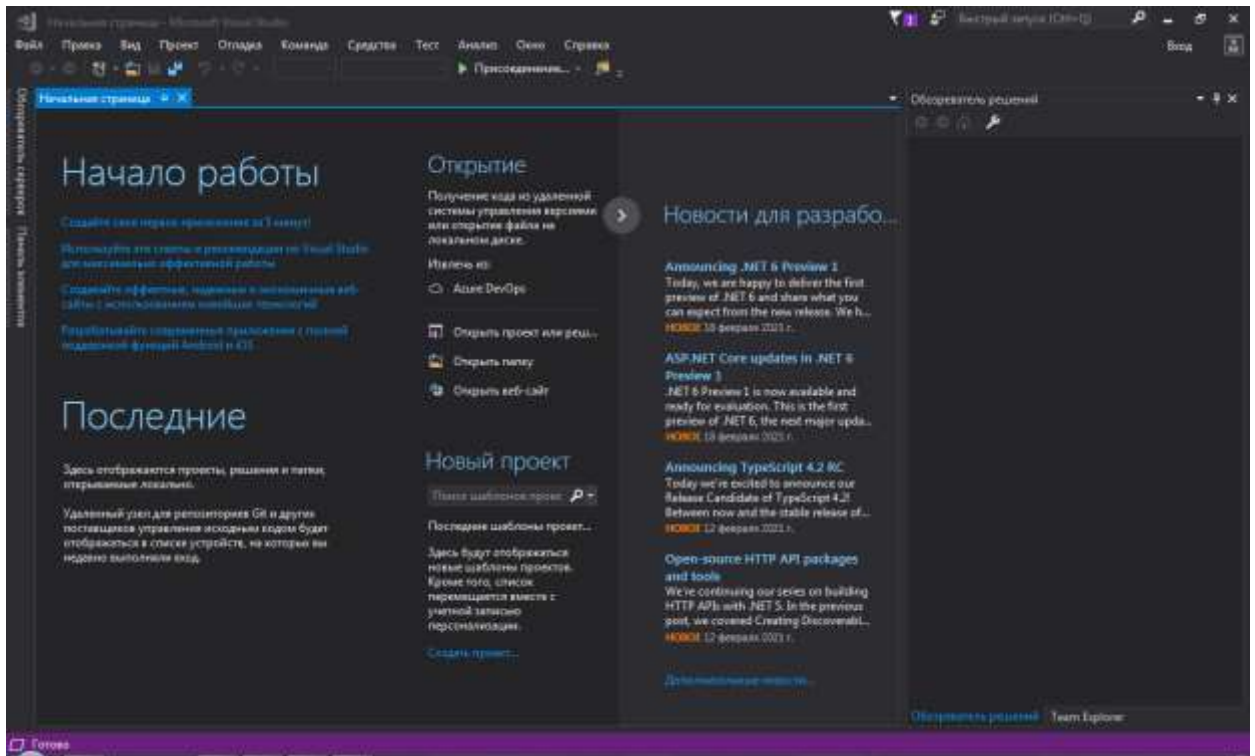


Рис. 1.1 – Головне вікно Visual Studio 2012

Права частина вікна Microsoft Visual Studio 2012 зайнята так званими інструментальними вікнами. В них відображаються компоненти додатків: елементи управління, з'єднання з базами даних, класи і властивості, що використовуються в проектах.

Інструментальні вікна можна налагодити на свій розсуд для максимальної зручності використання в процесі роботи. Їх можна перетягувати, розташовувати в будь-якому місці екрану. Інструментальні вікна можна також налаштовувати так, щоб вони постійно знаходилися на екрані. Для цього необхідно включити кнопку «Автоматически скрывать» (Auto Hide), що розташована в правому верхньому кутку вікна.

Важливим компонентом інтерфейсу є також спосіб відображення вікон документів всередині інтегрованого середовища розробки.

Існує два способи відображення вікон: стандартний інтерфейс з певною кількістю вікон, який використовується багатьма програмами, коли всередині основного вікна створюється нове вікно, яке є окремим вікном із заголовком, воно не може бути винесене за межі головного вікна; та інтерфейс на основі закладок, коли вікно документа завжди займає весь робочий простір всередині головного вікна, а перемикання між вікнами можливо шляхом клацання по

відповідній вкладці.

Для створення нового додатка ASP.NET слід виконати команду Файл–Создать–Проект (File–New–Project).

Слід звернути увагу на те, що команда New Project використовується у Visual Studio 2017 на відміну від Visual Studio 2012, де Web–додаток не є проектом.

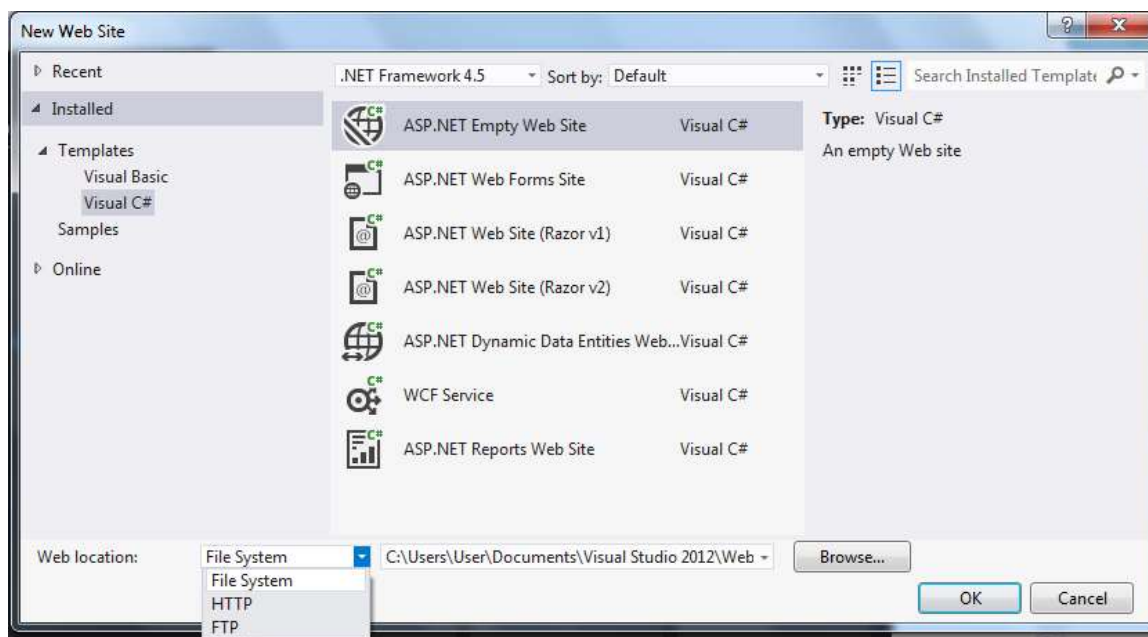


Рис. 1.2 – Вибір типу додатка, який необхідно створити

У вікні New Web Site (рис. 1.2), що відкрилося, можна вказати наступні відомості:

– шаблон (Templates) – визначає файли, з яких починається Web–сайт. Підтримуються різні типи базових додатків ASP.NET – додаток Web–сайту і додаток Web–служб. Шаблони цих додатків компонуються і компілюються аналогічно. Фактично можливе додавання Web–сторінки до додатку Web–служби, а також Web–служби до звичайного Web–дodatка. Відмінність полягає у файлах, які Microsoft Visual Studio 2012 створює за замовчуванням. У Web–дodatку робота починається із зразка Web–сторінки, в додатки Web–служби — із зразка Web–служби. Крім того, є складніші шаблони для певних типів сайтів. Можна також створювати власні шаблони або завантажувати готові шаблони сторонніх постачальників;

– місцезнаходження (Location) — визначає місце зберігання файлів Web–сайту. Звичайно це File System (файлова система) — вказується папка на локальному комп'ютері або шлях. Але можливо також редагування сайту

безпосередньо по HTTP– або FTP–протоколу;

– мова (Language) — визначає мову програмування .NET, яка використовується за замовчуванням для кодування Web–сайта.

Альтернативою ручному введенню місцерозташування є натискання кнопки "Browse" (Огляд), що відображає діалогове вікно "Web location" (Вибір місця розташування). Список, що дозволяє обрати тип місцерозташування має наступні пункти:

– File System (файлова система) — дозволяє проглядати дерево дисків і каталогів або спільних ресурсів, що надаються іншими комп'ютерами мережі.

– HTTP — надає доступ до Web–сайту по певному URL з використанням HTTP.

– FTP (FTP–сайт) — менш зручно, оскільки перед підключенням слід вказати інформацію про з'єднання, включаючи FTP–сайт, порт, каталог, ім'я користувача і пароль.

Після повернення в діалогове вікно створення Web–сайта і вказівки всіх необхідних опцій, Visual Studio створить новий Web–додаток. Новий Web–сайт починається з одного файлу — початкової сторінки за замовчуванням default.aspx.

Наявність розширення .aspx у файлі говорить про те, що він містить набір директив ASP.NET і має бути виконаний платформою .NET Framework. При цьому, дуже важливими елементами цього файлу є директиви середовища виконання, а також вбудовані в сторінку серверні елементи управління. Ці елементи повинні обов'язково розташовуватися всередині HTML–елемента form, що виконується на стороні сервера, і позначається тегом <asp: параметри елемента />. Наприклад, опис серверного елемента Button, який вбудовано в сторінку, виглядає наступним чином:

```
<asp:Button ID="Button1" runat="server" Text="Button" />.
```

Як видно з цього прикладу, після ключового слова asp вказується тип елемента, який відповідає його класу, описаному в .NET Framework, потім ID елемента, вказівка на його обробку на стороні сервера та інші параметри.

Особливе значення при розробці додатків грає вікно панелі компонентів «Toolbox» (Панель елементов). У Visual Studio вікно Toolbox відображає елементи управління і компоненти, які можна перетягувати у вікно документа. Вміст Toolbox залежить від типу документа для редагування. Наприклад, при редагуванні Web–форми Toolbox містить серверні елементи управління, HTML–елементи управління, елементи управління, зв'язки з даними, та інші компоненти, які можна розташовувати в Web–формі.

Файл Default.aspx.cs містить програмний код, який прив'язується до сторінки. Організація цього файлу практично повністю повторює організацію аналогічного файлу для проекту Windows–додатка. Файл починається з підключення різних просторів імен, що містять описи тих класів .NET Framework, які необхідно використовувати в даному модулі. Потім файл містить опис класу сторінки, що складається з різних функцій, зокрема таких, що прив'язані до обробки подій даної форми. За замовчуванням створено шаблон функції — обробника події відкриття сторінки Page_Load.

Принцип розробки застосування в ASP.NET повністю відповідає об'єктно–орієнтованому підходу. Програміст в процесі створення Web–додатка оперує класами та визначає їх атрибути і значення, а також методи, що призначені для виконання об'єктами класу дій, які відповідають подіям сторінки.

Завдання для виконання

Виконання лабораторної роботи полягає в створенні Web–сайту, який містить одну сторінку з формою для введення параметрів та розрахунку результатів за індивідуальним варіантом. Варіанти індивідуальних завдань наведені в табл. 1.1. Звіт повинен містити покроковий опис процесу створення Web–додатка розрахунку за допомогою Microsoft Visual Studio 2012.

Таблиця 1.1

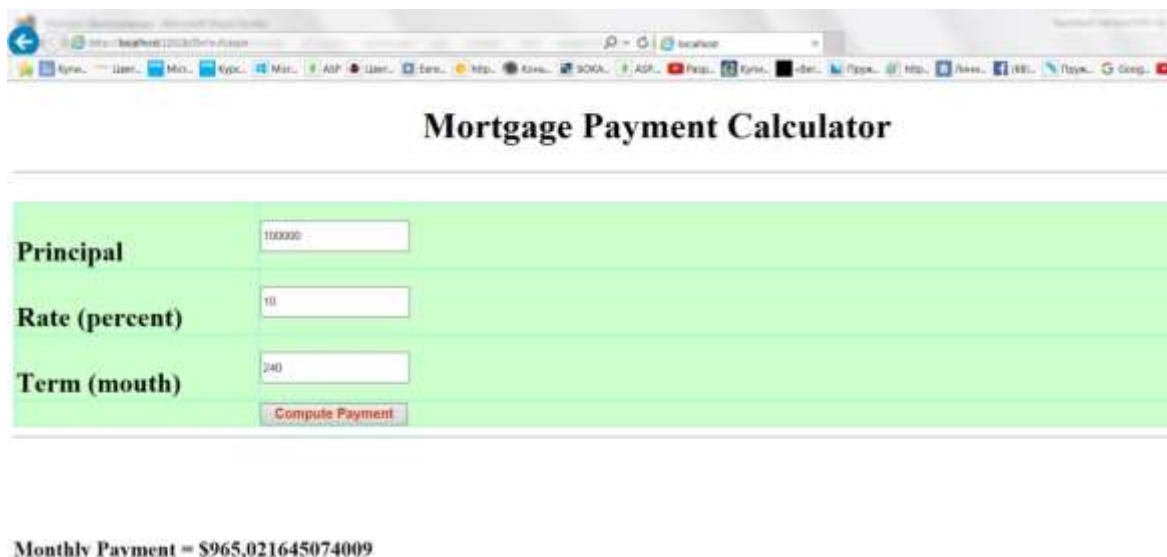
Варіанти індивідуальних завдань

№	Поле 1 (Textbox1)	Поле 2 (Textbox2)	Функція обробки полів (Button)
1	Номинал купюри (1, 2, 5, 10 і т.д.)	Кількість купюр	Обчислити суму купюр
2	Ціна товару	Кількість одиниць товару	Обчислити загальну вартість товару
3	Калорійність 100г продукту	Вага продукту в грамах	Обчислити загальну калорійність продукту
4	Кількість хвилин	Кількість секунд	Обчислити загальну кількість секунд
5	Дійсне число – швидкість руху (м / с)	Ціле число – час руху в хвилинах	Обчислити відстань (в метрах)
6	Дійсне число – перший катет прямокутного трикутника	Дійсне число – другий катет прямокутного трикутника	Обчислити довжину гіпотенузи прямокутного трикутника

№	Поле 1 (Textbox1)	Поле 2 (Textbox2)	Функція обробки полів (Button)
7	Ціле число – нижня частина трапеції	Ціле число – верхня частина трапеції	Обчислити півсуму частин трапеції
8	Ціле число – x	Ціле число – y	Обчислити цілу частину від ділення x на y
9	Тривалість телефонної розмови в хвилиnach	Вартість однієї хвилини розмови	Обчислити загальну вартість розмови
10	Дійсне число – a	Дійсне число – b	Обчислити різницю квадратів чисел $a^2 - b^2$
11	Кількість годин роботи	Тариф оплати за годину роботи	Загальна вартість роботи
12	Радіус кола основи	Висота циліндра	Обчислити площу поверхні циліндра
13	Напруга (в вольтах)	Опір (в Омaх)	Обчислити значення струму (в амперах)
14	Струм в амперах	Опір резистора R1 (в Омaх)	Обчислити потужність на ділянці електричного кола (в Вaтах)
15	Маса тіла – m (в грамах)	Швидкість руху – v (в м / с)	Обчислити кінетичну енергію тіла, що рухаєт $W_k = \frac{mv^2}{2}$
16	Дійсне число – перший катет прямокутного трикутника	Дійсне число – другий катет прямокутного трикутника	Обчислити тангенс кута, протилежного другому катету прямокутного трикутника
17	Радіус кола	Кут в радіанах	Обчислити довжину дуги
18	Номінал монети (1, 2, 5, 10 та ін.)	Кількість монет	Обчислити суму монет
19	Дійсне число – ліва межа діапазону	Дійсне число – права межа діапазону	Квадрат довжини діапазону
20	Ціле число – x	Ціле число – y	Обчислити куб більшого з чисел

Приклад виконання індивідуального завдання

Введіть розмір позики, процентну ставку, термін позики (у місяцях) і клацніть кнопку Compute Payment. Розрахований місячний платіж буде відображений внизу сторінки.



Mortgage Payment Calculator

Principal	<input type="text" value="100000"/>
Rate (percent)	<input type="text" value="10"/>
Term (month)	<input type="text" value="240"/>

Monthly Payment = \$965,021645074009

Рис.1.3 – Web–додаток розрахунку платежів

1) Створення проєкту Web–додатка

Запустите Visual Studio.NET. В меню виберіть команду Файл Создать Веб–сайт(File/New Web Site). В діалоговому вікні, що відкрилося, задайте Templates: Пустой Веб–сайт ASP.NET, Language: Visual C# і шлях до каталогу, де буде збережений проєкт.

На рисунку 1.4 приведено вікно зі створеним веб–сайтом за шаблоном Пустой Веб–сайт ASP.NET. В цьому випадку необхідно самостійно додавати сторінки. Для цього виконуються наступні дії:

- клацнути правою кнопкою миші на імені Веб–сайту, в меню вибрати **Добавить новый элемент**;
- зі списку шаблонів вибрати шаблон **Форма Web– Form**, поставити галочку в клітинці **Разместить код в отдельном файле** і натиснути кнопку **Добавить**.

2) Додавання таблиці

Клацніть вікно конструктора Web–форм Конструктор (Design), щоб встановити на ньому фокус введення. Потім вибрати команду Visual Studio.NET Таблица/ Вставить таблицу (Table/InsertTable), додати до Web–форми HTML–таблицю. Заповнити діалогове вікно Вставка таблицы (Insert

Table), що з'явилося. Зокрема, встановити Строк (Rows) в 4, Столбцов (Columns) в 2, Задать ширину (Width) в 100 %, Після клацання ОК таблиця з'явиться у вікні дизайнера форм.

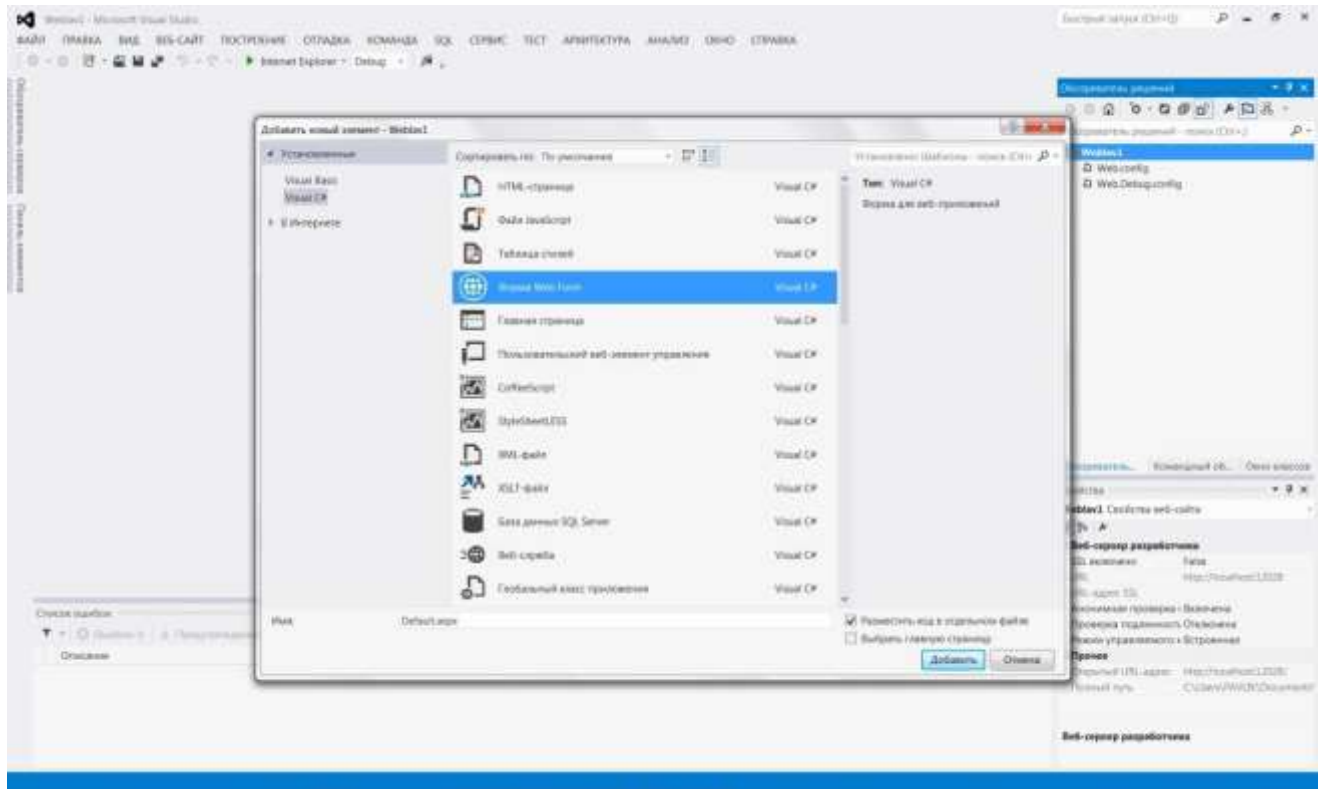


Рис.1.4 – Додавання сторінки в Web-додаток

3) Додавання тексту

Клацніть лівою кнопкою миші в лівому верхньому кутку таблиці. На екрані з'явиться курсор – будь-який введений текст буде поміщено в даний елемент таблиці. Введіть «Principal». Потім перейдіть у вікно Properties і змініть властивість text-align на «right», щоб вирівняти текст по правому краю. Аналогічно додайте «Rate (percent)» в наступний рядок і «Term (months)» в рядок під ним. Перетягніть вертикальний роздільник елементів таблиці так, щоб ліва колонка таблиці стала достатньо широкою і могла вміщати введений текст. На рис.1.6 показаний остаточний вид таблиці.

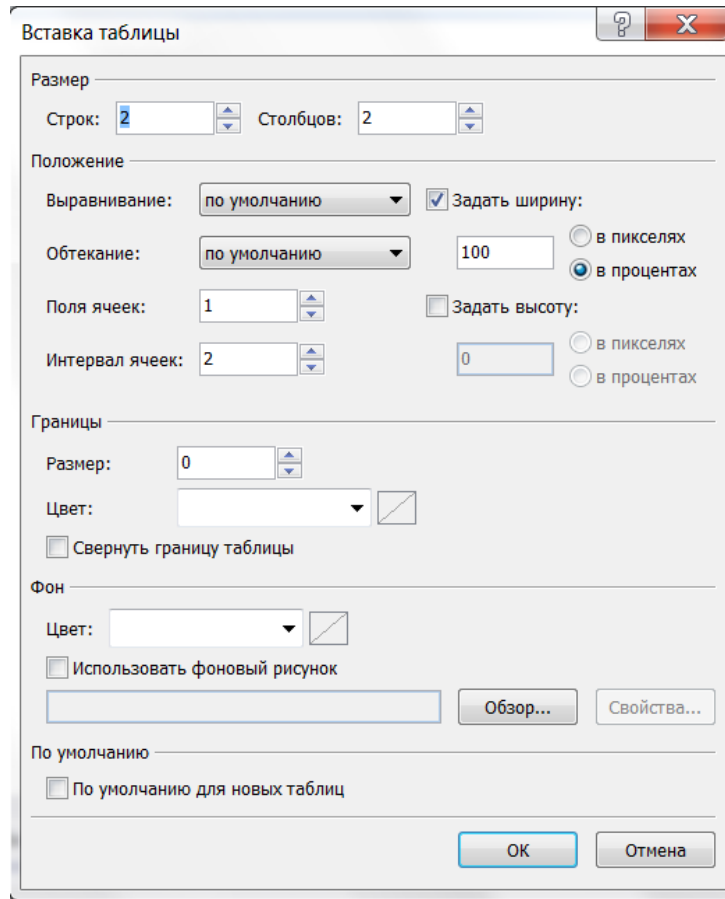


Рис.1.5 – Додавання таблиці до Web-форми

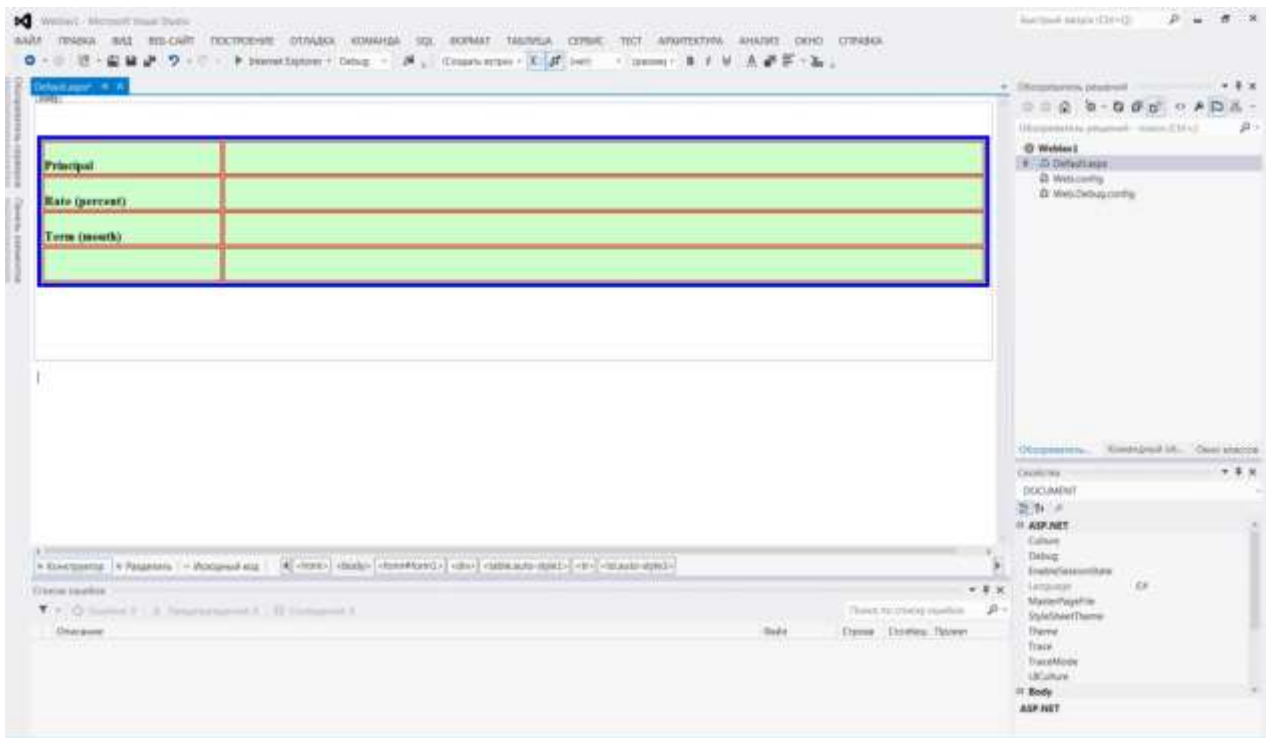


Рис.1.6– Форма LoanCalc після додавання тексту

4) Додавання елементів управління TextBox

Перетягніть в праву частину таблиці три елемента управління TextBox з вкладки Standard.

У вікні Properties встановіть ідентифікатори нових елементів управління в «Principal», «Rate» і «Term» відповідно.

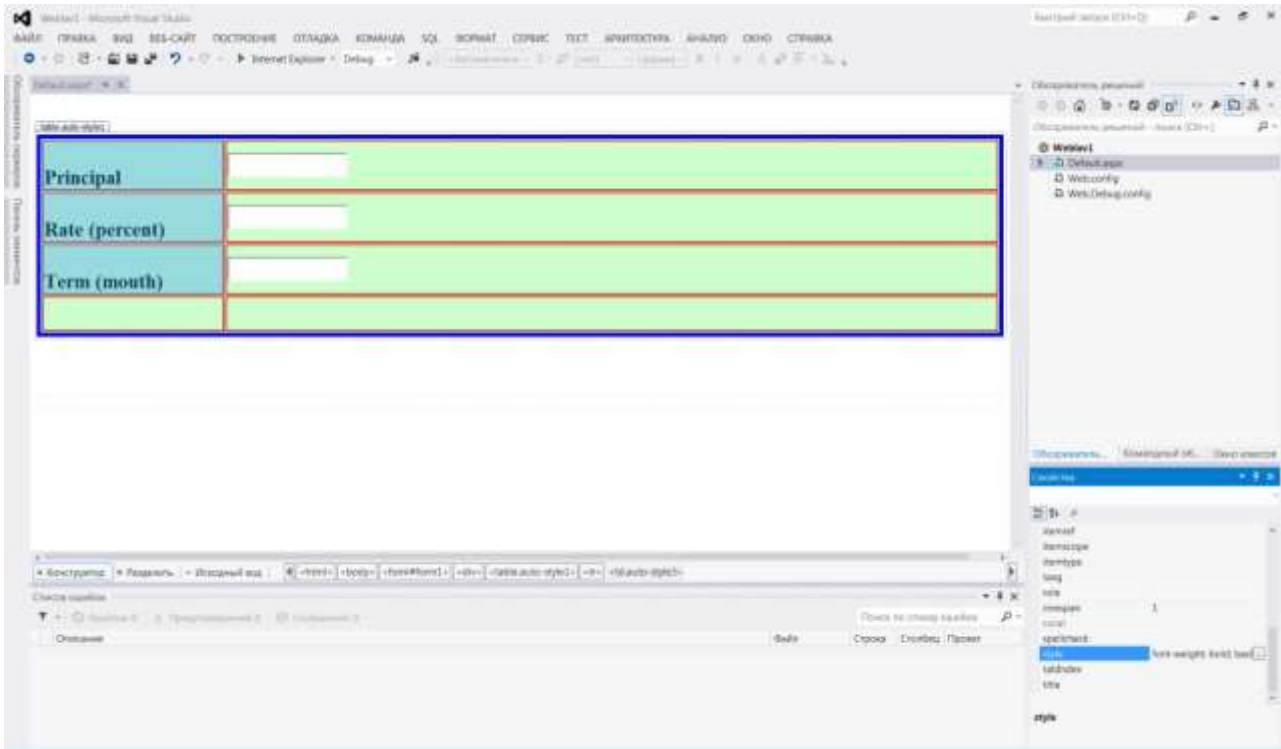


Рис.1.7 – Форма LoanCalc після додавання елементів TextBox

5) Додавання елемента управління Button

Додайте елемент управління Button в праву частину нижнього рядка таблиці. Змініть розмір кнопки так, щоб її ширина збігалася з шириною текстового поля, розташованого над нею. Змініть текст кнопки на «Compute Payment», а ідентифікатора — на «PaymentButton».

6) Додавання елемента управління Label

Виберіть у вікні Toolbox елемент управління Label і додайте його до форми прямо під таблицею. Змініть текст елемента управління на порожній рядок, а ідентифікатору на «Output».

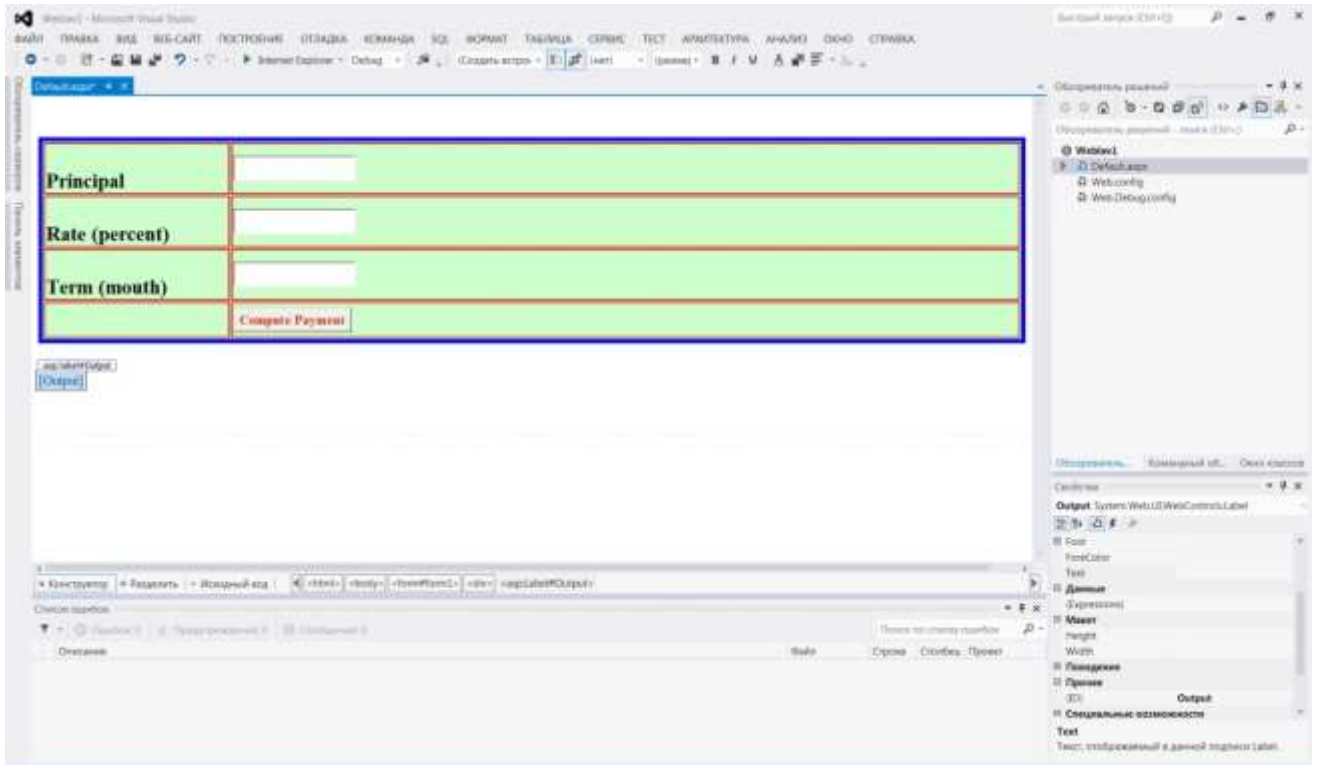


Рис.1.8 – Форма LoanCalc після додавання елемента Label

7) Редагування HTML

Для поліпшення зовнішнього вигляду форми додайте декілька елементів HTML. Спочатку клацніть кнопку Исходный код (Source) внизу вікна дизайнера, щоб відобразити HTML, який було створено для форми. Між тегами `<body>` і `<form>` додайте наступні теги:

```
<h1>Mortgage Payment Calculator</h1> <hr>
```

Потім прокрутите текст до кінця файлу і між тегами `</table>` і

```
<asp:Label> додайте:  
<b> <hr> <br> <h3>
```

Перенесіть тег `</h3>`, який було вставлено Visual Studio .NET, після тега `<asp:Label>`. Тепер клацніть кнопку Конструктор (Design) внизу дизайнера форм, щоб вийти з режиму Source в режим малювання форми. На рис.1.9 показано, як повинна виглядати форма після внесених змін.

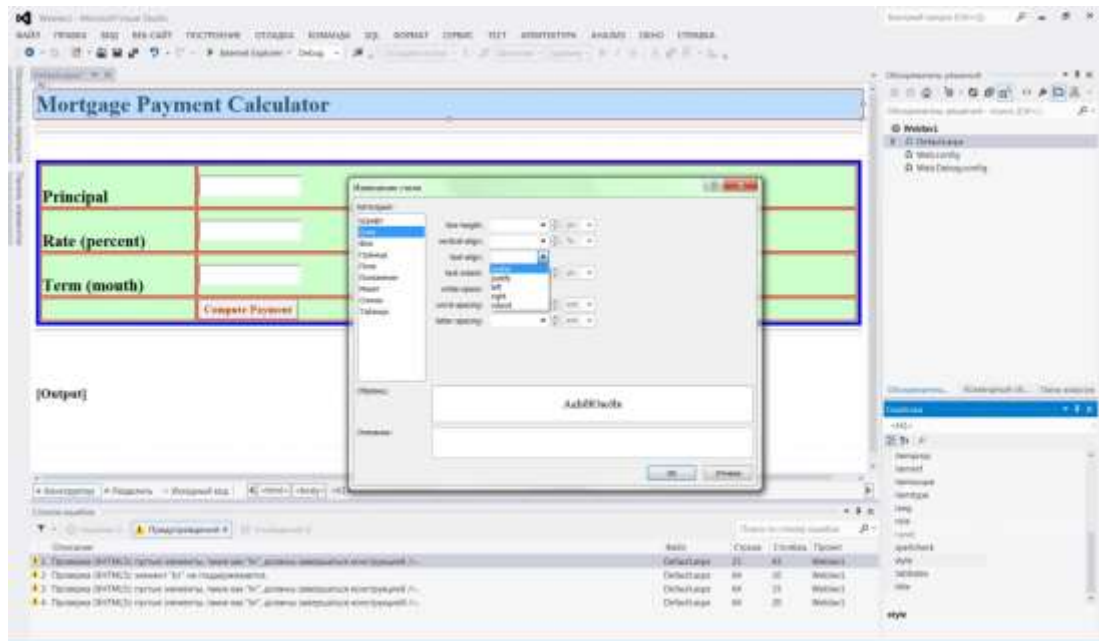


Рис.1.9– Форма LoanCalc після додавання HTML-тегів

8) Додавання обробника Click

Двічі клацніть кнопку Compute Payment на формі. Visual Studio .NET додасть в WebForm1.aspx.cs метод PaymentButton_Click і відобразить його в редакторі програм. Додайте цей код в порожнє тіло методу:

```
try {
    double principal = Convert.ToDouble (Principal.Text); double rate =
    Convert.ToDouble (Rate.Text) / 100; double term = Convert.ToDouble (Term.Text);
    double tmp = System.Math.Pow (1 + (rate / 12), term); double payment =
    principal * (((rate / 12) * tmp) / (tmp - 1));
    Output.Text = "Monthly Payment = " + payment.ToString ("$");
}
catch (Exception) { Output.Text = "Error";
}
```

PaymentButton_Click — не просто метод, це обробник події. Обробник реагує на подію Click, витягує з елементів управління TextBox дані, які були введені користувачем, визначає відповідний щомісячний платіж і відображає результат в елементі управління Label.

9) Компоновка і тестування

Для компіляції коду виберіть Начать отладку в меню Отладка (Build). Якщо компоновка пройшла без помилок, виберіть в меню Debug пункт Start

(або Start Without Debugging) для запуску додатка. Коли Web-форма відобразиться в Internet Explorer переконайтеся, що вона працює правильно, введіть такі значення:

Principal: 100000

Rate: 10

Term: 240

Клацніть кнопку Compute Payment. Внизу сторінки з'явиться «Monthly Payment = \$965.02».

Серед безлічі файлів в каталозі LoanCalc найбільший інтерес представляють WebForm1.aspx і WebForm1.aspx.cs. Вони містять початковий код LoanCalc. Більшість з цих додаткових файлів в даному прикладі не потрібні, але Visual Studio .NET все одно створює їх.

WebForm1.aspx не містить програмного коду, а містить тільки HTML. Visual Studio .NET завжди використовує в своїх Web-формах фоновий код, тому весь код на C# знаходиться в WebForm1.aspx.cs. Велика частина їх вмісту згенерувала Visual Studio.NET. Оператори, які додав користувач, показані курсивом.

ASPX-файл визначає інтерфейс користувача за допомогою HTML і Web-елементів управління, а CSS-файл містить обробник події Click від кнопки Compute Payment, а також код, що підключає цей обробник до кнопки. Жоден з файлів не містить нічого такого, чого користувач не може написати вручну, але абсолютно очевидно, що візуальне створення Web-форм набагато швидше і менше схильне до помилок, чим їх ручне кодування.

Контрольні питання:

1. Які засоби управління включені у середовище розробки Visual Studio 2012?
2. Як здійснити налаштування відображення вікон документів усередині інтегрованого середовища розробки?
3. Як створити новий додаток ASP.NET?
4. Призначення та основні функції вікна панелі компонентів Toolbox?
5. Як організовано і що містить файл WebForm1.aspx?
6. Як організовано і що містить файл і WebForm1.aspx.cs?
7. Визначте і перерахуйте переваги застосування технології ASP.NET. для розробки Web-додатку.

ЛАБОРАТОРНА РОБОТА №2

Створення WEB-форми у середовищі розробки Visual Studio2012.

Робота з масивами, рядками, процедурами і функціям на мові С#

Мета роботи: розробка веб-додатків на мові С# в інтегрованому середовищі Microsoft Visual Studio 2012. Робота з масивами та рядками. Вивчення і практичне застосування основних принципів роботи з рядками, які орієнтовані на вирішення практичних завдань. Визначення принципів опису, виклику і передачі параметрів в процедури і функції.

Постановка завдання: створити в Microsoft Visual Studio 2012 Web-додаток, що містить форму з заданою кількістю полів для введення значень елементів числового або символьного масиву, скласти програму (мовою С#) для обробки елементів масиву та рядків. Виконати *Завдання №1, 2* згідно з варіантом завдань.

Теоретичні відомості

Робота з масивами в С#

Масив – це послідовність однотипних елементів, яка названа одним ім'ям. Ці дані легко відсортувати, перебрати і обробити. Масив в С# є об'єктом посилального типу. Створення масиву – це двоступінчатий процес; спочатку оголошується посилальна змінна на масив, а потім для нього виділяється пам'ять і змінній привласнюється посилання на цю пам'ять.

Одновимірні масиви

Для одновимірного масиву синтаксис створення буде таким:

```
тип [ ] ім'я_масиву = new тип[розмір];
```

Наприклад, зарезервувати пам'ять під 10 елементів цілого типу можна так:

```
int[ ] array = new int[10];
```

Адресація елементів всередині масиву виконується за допомогою числового індексу, нумерація якого починається з нуля. Вихід індексу за межі компілятором не контролюється і виявляється середовищем CLR.

Якщо масив оголошується як поле класу, то всі елементи за замовчуванням набувають найближчого за типом значення (для числового – нуль, для строкового – порожньо, для bool – false, для посилального – null). Якщо масив оголошується всередині методу як локальна змінна, то його перед використанням потрібно явно ініціалізувати.

Масив в C# – це тип, що походить від класу System.Array, тому всі масиви успадковують від цього класу фіксований набір об'єктних властивостей і методів. Крім того, сам клас Array містить ряд статичних методів (рівня класу), що дозволяють виконувати обробку масивів як об'єктів.

Одне і те ж посилання на масив можна використовувати багато разів при створенні декількох сумісних з її типом масивів. У такому разі адресація колишнього масиву буде загублена. Наприклад:

```
int[ ] array = new int[10];  
array = new int[20];
```

Можна виконувати відразу і оголошення і ініціалізацію масиву, тоді створення масиву і підрахунок його розмірності виконає компілятор за списком ініціалізації. В цьому випадку ключове слово new не потрібне. Наприклад:

```
int[ ] array = { 1, 2, 3, 4, 5 };
```

Можна створити масив фіксованої розмірності і відразу його ініціалізувати. Тоді розмір списку ініціалізації повинен строго відповідати замовленій розмірності масиву. Наприклад:

```
int[ ] array = new int[5]{ 1, 2, 3, 4, 5 };
```

Можливо використовувати і такий синтаксис – без вказівки розмірності, але з ініціалізацією:

```
int[ ] array = new int[] { 1, 2, 3, 4, 5 };
```

Можна спочатку створити масив, а потім привласнити йому значення. І теж потрібно стежити, щоб не вийти за межі встановленої розмірності масиву.

```
int[ ] array = new int[5];  
for (int i = 0; i < array.Length; i++) array[i] = i;
```

Елементами масиву можуть бути значення довільного типу, зокрема масиви, класи, структури і інтерфейси. Масиви можуть бути як одновимірними, так і багатовимірними.

Приклад створення масиву, що складається з дванадцяти елементів типу string:

```
string[ ] st = new string[12];
```

При створенні масивів необхідно враховувати, що для створення масиву фіксованої довжини, як в попередніх прикладах, необхідно завжди використовувати ключове слово new. Таким чином, наступне визначення масиву неприпустимо і приведе до виникнення помилки ще на етапі компіляції:

```
int[3] a={3,5,6};
```

Багатовимірні масиви

Окрім простих одновимірних масивів С# підтримує також багатовимірні масиви, які у свою чергу діляться на дві категорії. До першої категорії відносяться масиви, кількість елементів кожного рядка яких складається з однакової кількості елементів. Таким чином, масив можна розглядати як прямокутник, а самі такі масиви називають "прямокутними". До другої категорії відносяться масиви, кількість елементів в рядках яких не однаково. Такі масиви утворюють прямокутник, у якого одна сторона представляє ламану лінію, тому такі масиви називають "ламаними".

Оголошення і заповнення масиву проводиться стандартним способом, для цього організується два цикли:

```
Random rnd=new Random(); int[,] Matrix;  
Matrix = new int[5, 4]; for (int i = 0; i < 5; i++) for (int j=0;j<4;j++)  
Matrix[i,j]= rnd.Next(10,99);
```

При необхідності виведення значень елементів масиву на екран також організується цикл, що дозволяє послідовно перебирати всі елементи масиву:

```
for (int i = 0; i < 5; i++)  
{  
for (int j = 0; j < 4; j++)  
{  
Response.Write(Matrix[i, j].ToString()); Response.Write(" ");  
}  
Response.Write("<br/>");  
}
```

"Ламаний" масив може розглядатися як масив, кожен елемент якого є масивом. Як приклад створимо масив із змінною кількістю елементів в рядках:

```
Random rnd = new Random(); int[][] JMatrix = new int[5][];  
for (int i = 0; i < JMatrix.Length; i++)  
{  
JMatrix[i]= new int[rnd.Next(1,7)];  
}
```

Кількість елементів в рядку визначається випадковим чином у момент формування масиву. Кількість рядків задана жорстко і рівна п'яти. Масив не заповнюється ніякими числами, тому на екран виводитимуться значення, якими заповнюється масив за замовчуванням (в даному випадку це нуль). Для

виведення інформації на екран необхідно використовувати два цикли, як це показано нижче:

```
for (int i = 0; i < JMatrix.Length; i++)
{
    Response.Write("Кількість елементів в рядку " + i.ToString() + "=" +
JMatrix[i].Length.ToString()+" "); for (int j = 0; j < JMatrix[i].Length; j++)
    {
        Response.Write(JMatrix[i][j].ToString() + " ");
    }
    Response.Write("<br/>");
}
```

У циклі по j відбувається визначення кількості елементів в рядку за допомогою властивості Length.

Клас Array

Для детальнішого розуміння особливостей використання масивів в C# необхідно розглянути спеціалізований клас, що реалізує функції масиву. Всі типи даних в C# є класами, для яких як базовий виступає клас Object.

Клас Array – не виключення. Він реалізує всі базові властивості класів і є предком для всіх типів масивів, які визначені в мові C++ і синтаксис опису яких був приведений вище. Те, що клас Array є нащадком класу Object, дає можливість в класі Array визначати безліч різноманітних операцій, таких як копіювання, пошук, звернення, сортування та ін. В табл. 2.1 приведені найцікавіші методи класу Array.

Розглянемо приклад використання класу Array. Для цього створимо масив і забезпечимо можливість пошуку в ньому елементів.

Оголосимо масив myArray як статичний член класу Page, що складається з шести елементів типу int:

```
static Array myArray = Array.CreateInstance(typeof(Int32), 6);
```

Розмістимо на формі елементи TextBox і Button, яким привласнимо імена

```
tb_value і btn_find
```

відповідно.

Методи класу Array

Метод	Дії, які виконує метод
BinarySearch()	Пошук елементів в одновимірному відсортованому масиві
Sort()	Сортування елементів одновимірного масиву
Clear()	Очищення елементів масиву в заданому діапазоні індексів
CopyTo()	Копіювання елементів початкового масиву в масив-призначення
GetLength(), Length	Визначення кількості елементів у вказаному вимірюванні масиву
GetLowerBound()	Визначення нижньої межі масиву
GetUpperBound()	Визначення верхньої межі масиву
GetValue()	Повертає значення вказаного індексу для масиву
SetValue()	Встановлює значення вказаного індексу для масиву
Reverse()	Розставляє елементи одновимірного масиву в зворотному порядку
Rank	Визначення кількості вимірювань вказаного масиву

В обробник процедури натиснення на кнопку введемо наступний код, що заповнює масив випадковими числами, сортує його і здійснює пошук введеного в текстове поле елемента. Код процедури натиснення на кнопку:

```
protected void btn_find_Click(object sender, EventArgs e)
{
    Random rnd = new Random(); Response.Write("Початковий масив
<br/>"); for (int i = myArray.GetLowerBound(0); i <=
myArray.GetUpperBound(0); i++)
    {
        myArray.SetValue(rnd.Next(1, 10), i); Response.Write(myArray.GetValue(i)
+ "\t");
    }
    Response.Write("<br/>");
    Array.Sort(myArray);
    Response.Write("Після сортування:<br/>"); for (int i =
myArray.GetLowerBound(0); i <= myArray.GetUpperBound(0); i++)
    {
        Response.Write(myArray.GetValue(i) + "\t");
    }
    object npo = Convert.ToInt32(tb_value.Text);
```

```

int findIndex = Array.BinarySearch(myArray, nro); if (findIndex<0)
{
Response.Write("<br/>Елемент не знайдений");
}
else
{
Response.Write("<br/>Елемент знайдений у позиції <b>" + findIndex.
ToString() + "</b>");
}
}
}

```

Приклад роботи програми представлений на рис.2.1.

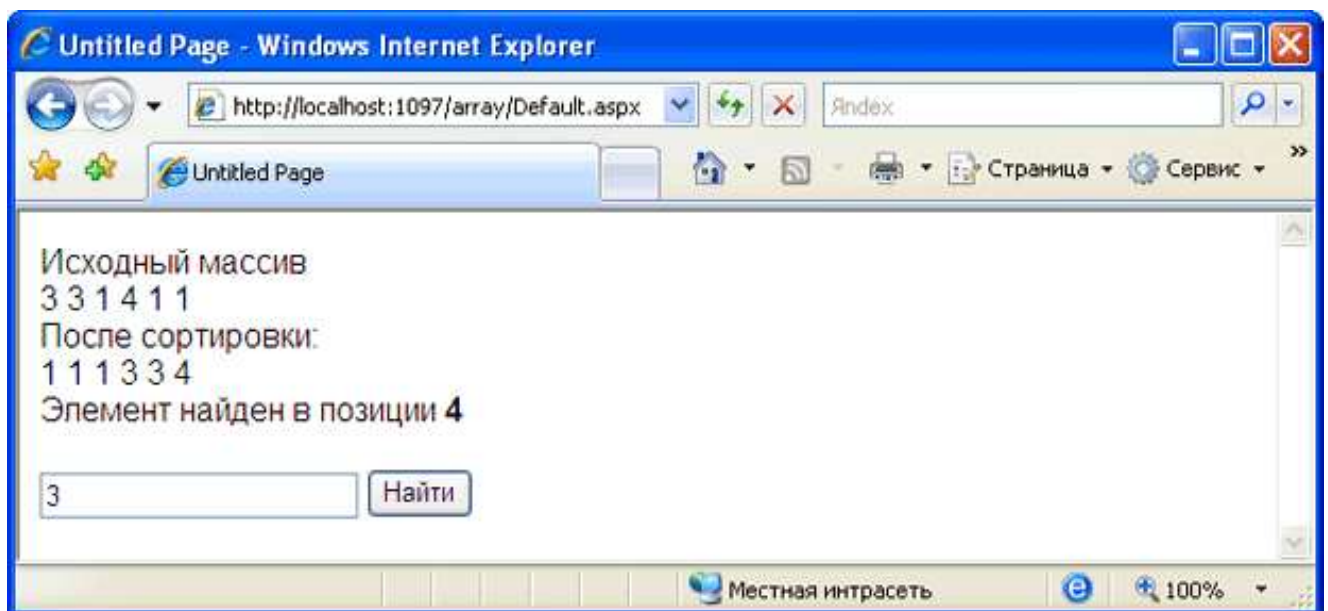


Рис. 2.1– Результат роботи програми заповнення, сортування і пошуку елементів масиву

Як видно, працювати з масивами в С# досить просто, при цьому слід враховувати досить великі можливості класу Array.

Робота з рядками в С#

По аналогії з масивами всі рядки в С# походять від одного базового класу – System.String, у якому реалізовано багато різних методів, що здійснюють всілякі операції над рядками. Найцікавіші методи класу String представлені в табл. 2.2.

Методи класу String

Метод	Дії, які виконує метод
Length	Дозволяє отримати кількість символів в рядку.
Concat()	Дозволяє з'єднати декілька рядків або змінних типу object.
CompareTo()	Дозволяє порівняти два рядки. У разі рівності рядків результат виконання функції дорівнює нулю. При позитивному значенні функції більшим є рядок, для якого викликався метод.
Copy()	Створює нову копію існуючого рядка.
Format()	Застосовується для форматування рядка з використанням різних примітивів (рядків і числових даних) і виразів вигляду {0}.
Insert()	Дозволяє вставити один рядок всередину існуючого.
Remove() Replace()	Видаляють або замінюють символи в рядку.
ToUpper() ToLower()	Перетворюють всі символи рядка в рядкові або прописні.
Chars	Дозволяє отримати символ, що знаходиться в певній позиції рядка.
Join()	Створює рядок сполученням заданих рядків і розділяє їх рядком-роздільником.
Replace()	Замінює один символ рядка іншим.
Split()	Повертає масив рядків з елементами – підрядками основного рядка, між якими знаходяться символи-роздільники.
Substring()	Дозволяє отримати підрядок основного рядка, що починається з певного символу і що має задану довжину.
Trim()	Видаляє пропуски або набір заданих символів на початку і в кінці основного рядка.
ToCharArray()	Створює масив символів і поміщає в нього символи початкового рядка.

При роботі з рядками в C# необхідно враховувати наступне. Тип String є посилальним типом. Проте, не дивлячись на це, при використанні операцій порівняння відбувається порівняння значень строкових об'єктів, а не адрес цих об'єктів, які розміщені в оперативній пам'яті. Крім того, оператор "+" об'єкта string переобтяжений так, що при його необхідності використовується метод Concat().

Виконаємо декілька прикладів використання можливостей роботи з рядками.

Приклад 1

Реалізувати функцію `MakeLine`, яка буде створювати рядок, що складається з символів, які визначаються шляхом нескладних обчислень. Результати обчислень заносяться в масив рядків. Таким чином, масив `sArr` після завершення циклу містить значення всіх отриманих чисел. Після цього, використовуючи роздільник, що передається в функцію як аргумент, а також функцію `Join`, значення всіх елементів масиву `sArr` об'єднуються в рядок:

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write(MakeLine(0, 5, " ", "));    Response.Write("</br>");
    Response.Write(MakeLine(1, 6, " " " "));    Response.Write("</br>");
    Response.Write(MakeLine(9, 9, ":", " "));    Response.Write("</br>");
    Response.Write(MakeLine(4, 7, "< " "));
}
private static string MakeLine(int initVal, int multVal, string sep)
{
    string[] sArr = new string[10];
    for (int i = initVal; i < initVal+10; i++)
        sArr[i-initVal]= String.Format("{0-3}", i * multVal); return String.Join(sep,
sArr);
}
```

Результат роботи програми представлений на рис.2.2.

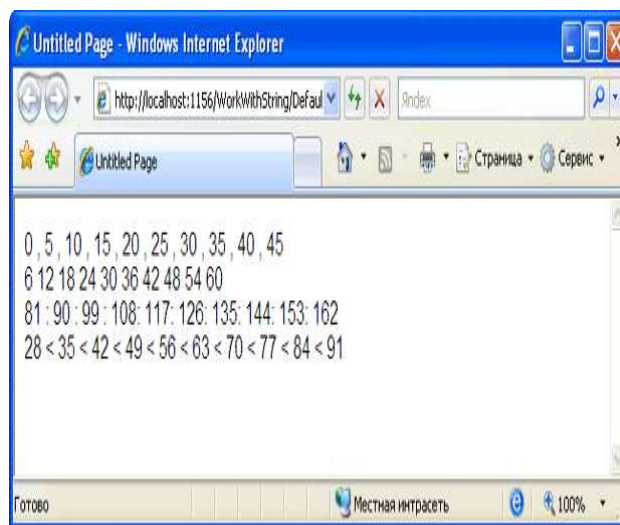


Рис.2.2 – Результат роботи програми з використанням функції `Join`

Досить часто при роботі з рядками виникає необхідність розділити рядок

на підрядки, які відокремлені один від одного заданими символами–роздільниками.

Приклад 2

Створити рядок символів, в якому присутні декілька символів–роздільників. За допомогою функції Split даний рядок розділяється на підрядки, які потім виводяться на екран, кожен в окремому рядку. Для завдання символів–роздільників використовується масив символів. В даному прикладі також застосовується функція Trim для того, щоб переконатися, що заданий рядок не складається з одних лише пропусків.

string words = "рядок, що містить декілька слів, а також знаків пунктуації: таких як двокранка і крапка.";

string [] split = words.Split(new Char [] { ' ', ',', '! ':'}); foreach (string s in split)

```
{  
if (s.Trim() != "") Response.Write(s+"<br>");  
}
```

Підсумком роботи даної програми буде наступний результат, зображений на рис. 2.3.

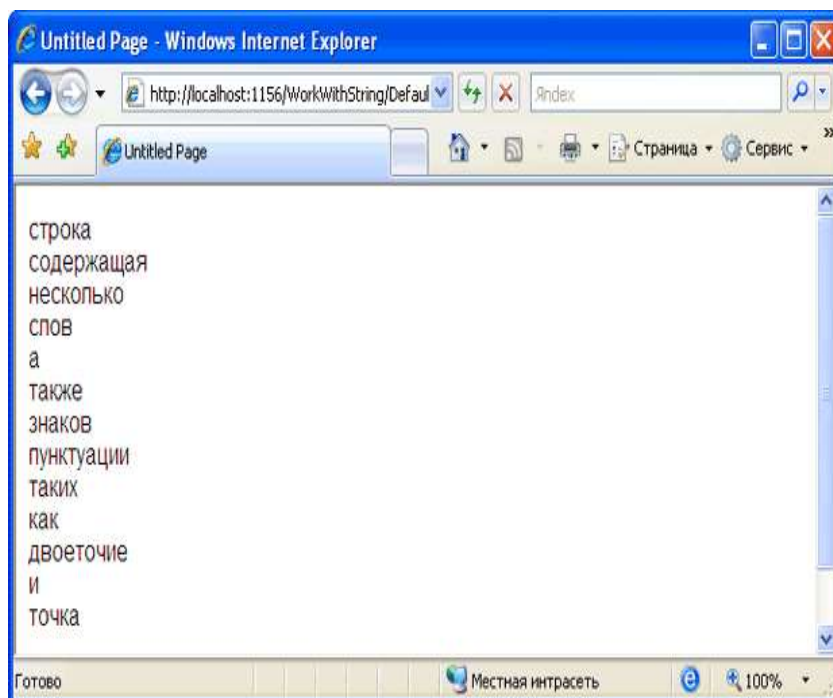


Рис. 2.3 – Результат роботи програми з використанням функції Split

Використання класу System.Text.StringBuilder

При роботі з рядками в C# необхідно враховувати те, що рядки є незмінними. Всі дії, направлені на зміну рядків, насправді не змінюють їхній початковий варіант. Вони лише повертають змінену копію рядка. Це можна побачити на наступному прикладі.

```
string s1 = "Приклад рядка"; string s2 = s1.ToUpper(); Response.Write(s1);  
Response.Write("<br>"); Response.Write(s2);
```

В даному випадку рядок `s1` не зазнав ніяких змін. Метод `ToUpper()` створив копію рядка `s1` і застосував до неї необхідні перетворення, тому на екран буде виведений як початковий рядок, так і рядок, що піддався зміні. Таке ж твердження справедливе і для звичайної операції конкатенації рядків. Результатом виконання операції конкатенації рядків є також новий рядок.

У ряді випадків слід уникати ситуацій, коли в результаті виконання операції створюється новий рядок, оскільки це неминуче пов'язано з додатковими витратами пам'яті і інших ресурсів комп'ютера при виконанні операції. C# містить спеціальний клас `StringBuilder`, використовуючи який можна уникнути створення копій рядків при їх обробці. Всі зміни, що вносяться до об'єкта даного класу, негайно відображаються в ньому, що у цілому ряді випадків набагато ефективніше, ніж робота з множиною копій рядка.

Основною операцією, яка найчастіше використовується класом `StringBuilder`, є операція додавання до рядка вмісту. Для цього існує метод `Append`. Наступний код додає один рядок до іншого і виводить результат у вікно браузера. При цьому змінюється оригінал рядка, копія не створюється:

```
StringBuilder sb = new StringBuilder("Я студент Одеського державного  
екологічного університету");  
sb.Append(", факультету комп'ютерних наук"); Response.Write(sb);
```

Окрім додавання клас `StringBuilder` містить безліч інших методів, найбільш значущі з яких перераховані нижче. Після того, як всі необхідні дії, пов'язані з обробкою рядка, були виконані, необхідно викликати метод `ToString()` для перекладу вмісту об'єкта в звичайний тип даних `string`.

При інтенсивній роботі з рядками рекомендується використовувати клас `StringBuilder`, оскільки це дозволяє зменшити витрати пам'яті, що пов'язані із створенням копії рядка при виконанні кожної операції.

Параметри методу ToString()

Параметр	Дії, які виконує метод
Length	Дозволяє отримати кількість символів в рядку.
Append	Додавання заданого рядка в кінець рядка об'єкта
AppendFormat	Додавання заданого рядка, що містить символи управління в кінець рядка об'єкта
CopyTo	Копіювання символів заданого сегменту рядка в задані елементи масиву символів
Insert	Додавання рядка в задану позицію рядка об'єкта
Remove	Видалення заданої кількості символів з рядка об'єкта
Replace	Заміна заданого символу або рядка об'єкта на інший заданий символ або рядок

Процедури і функції С#

Процедури і функції С# практично повністю відповідають їх визначенням в мові С++. Основна відмінність процедури від функції полягає в тому, що функція повинна завжди повертати деякий результат, і, крім того, має бути викликана у виразі. Процедура не повертає ніякого результату, її виклик здійснюється за допомогою оператора; крім того, вона має вхідні і вихідні аргументи, при цьому вихідних аргументів може бути досить багато. Таким чином, створювати функцію доцільно тільки у тому випадку, коли необхідно провести якісь-небудь обчислення, в результаті яких має бути отриманий результат у вигляді одного значення. Процедуру – відповідно у разі, коли може бути отриманий результат, що є набором якихсь-небудь значень.

Проте існує ряд нововведень, які стосуються функцій, що з'явилися в С#. Найбільш важливою особливістю використання функцій і процедур в С# є те, що, оскільки С# є повністю об'єктно-орієнтованою мовою програмування, функції не можуть існувати у відриві від класів. Тому функцію бібліотек процедур і функцій в С# виконують бібліотеки класів.

Оскільки відмінності між процедурами і функціями достатньо умовні, а існувати вони можуть тільки всередині класів, надалі як синонім процедур і функцій слід використовувати термін "метод".

Синтаксис опису методів:

*[атрибути][модифікатори]{void | тип результату функції} ім'я методу
([список формальних аргументів])*

Ім'я методу і список формальних аргументів представляють сигнатуру методу. Квадратні дужки, як це прийнято, показують, що їх вміст може бути

опущено при описі методу.

Атрибути і модифікатори є дуже важливою складовою опису будь-якого методу, проте вони будуть розглянуті при описі класів, оскільки мають до цього пряме відношення. Під модифікаторами маються на увазі модифікатори доступу, з яких буде розглянуто два: `private` і `public`.

`Private` означає, що даний метод є закритим, відповідно доступ до нього можуть отримати тільки методи того класу, в якому він оголошений. `Public` – навпаки, означає, що доступ до даного методу є відкритим і загальнодоступним з будь-якої точки застосування.

При визначенні методу обов'язковою є вказівка типу значення, що повертається, імені методу, а також круглих дужок. У випадку, якщо метод не повинен повертати якого-небудь значення в програму, яка викликає метод, вказується тип `void`, що є ознакою приналежності методу до розряду функцій. Простими прикладами опису функцій є наступні:

```
private void A()
{}
public int B()
{}
public long Stepen(int a, int b)
{
long r;
r = (long)Math.Pow(a, b); return (r);
}
```

В наведеному прикладі метод `A` є закритою процедурою, методи `B` і `Stepen` – відкритими функціями. У методів `A` і `B` не визначені формальні параметри, тоді як у методу `Stepen` два формальні параметри: `a` і `b`.

Як видно з цього прикладу, список формальних аргументів може бути порожнім, що є типовою ситуацією для методів класу. Проте формальні аргументи, що визначені для метода, можуть містити і деякі додаткові атрибути, що впливають на їх поведінку.

Повний синтаксис оголошення формального аргументу виглядає таким чином:

```
[ref|out|params] тип_аргумента имя_аргумента
```

При цьому обов'язковою є вказівка типу аргументу, який може бути скалярним, масивом, класом, структурою, – будь-яким типом, який передбачено в `C#`.

Іноді при виклику методу виникає необхідність передавати в метод

довільну кількість фактичних аргументів, не зважаючи на фіксовану кількість аргументів, що містяться в його визначенні. Для реалізації такої можливості необхідно задати ключове слово `params`. Воно задається один раз і вказується тільки для останнього аргументу списку, що оголошується як масив довільного типу. При виклику методу цьому формальному аргументу відповідає довільна кількість фактичних аргументів.

Всі аргументи методів можна розділити на три групи: вхідні, вихідні, такі, що оновлюються.

Вхідні, необхідні для передачі інформації методу, їх значення в тілі методу доступні тільки для читання. Вихідними є результати методу, вони отримують значення в ході роботи методу.

Аргументи третього типу здатні виконувати обидві функції.

Вихідні аргументи мають бути помічені ключовим словом `out`, при цьому в тілі методу має бути обов'язково присутнім оператор надання значення цьому аргументу; аргументи, які оновлюються, позначаються за допомогою ключового слова `ref`.

Для прикладу внесемо наступні зміни в описаний раніше метод `Stepen`:

```
public void Stepen(out long r,int a, int b)
{
    r = (long)Math.Pow(a, b);
}
```

Як видно з прикладу, функція `Stepen` була фактично перетворена в процедуру (тип значення функції був змінений на `void`). Також був доданий формальний аргумент `r`, який використовується як вихідний. У тілі методу цьому аргументу привласнюється значення, яке згодом може бути використане в програмі.

Виклик цього методу може бути здійснений таким чином:

```
long s;
Stepen(out s, 2, 6); Response.Write(s.ToString());
```

Важливо, що при виклику цього методу перший параметр також вказується з ключовим словом `out`. З прикладу видно, що при завершенні роботи методу `Stepen` і повернення в програму відбувається передача значення змінній `r` з методу в змінну `s`, що знаходиться в адресному просторі основної програми.

Для передачі довільної кількості значень початкових даних в метод для їх обробки необхідно використовувати ключове слово `params`.

Приклад 3

Створіть метод `Stepen`, в якому може передаватися довільна кількість чисел для знаходження суми їх квадратів. Оголошення методу в цьому випадку виглядає таким чином:

```
public void Stepen(out long r, int a, params int[] b)
{
    r = 0;
    foreach (int i in b)
        r += (long)Math.Pow(i, a);
}
```

Виклик методу може бути здійснений так:

```
int[] digits = {1,8,4}; Stepen(out s, 2, digits); Response.Write(s.ToString());
```

Для виклику методу необхідно сформувати масив цілих чисел, який потім слід передати як третій аргумент в метод. Результат обчислення суми накопичується в змінній `r`, а потім передається в змінну `s` програми. Результатом роботи програми буде повернення результату у вигляді числа – 81.

Іноді виникає необхідність не тільки передавати довільну кількість початкових даних для розрахунку, але і отримувати із процедури довільну кількість змінних, що містять результати розрахунку.

Приклад 4

Створіть процедуру, що дозволяє отримувати масив чисел, підносити кожне з них до певного степеня, передавати результат розрахунку в програму і виводити результат на екран. Процедура в цьому випадку виглядатиме таким чином:

```
public void Stepen(out long[] r, int a, params int[] b)
{
    r = new long[0]; Array.Resize(ref r, b.Length); int j = 0;
    foreach (int i in b)
        r[j++] = (long)Math.Pow(i, a);
}
```

Необхідно вказати на наступні важливі особливості. По-перше, як параметр, що повертає значення в програму, в даному випадку використовується масив чисел типу `long`. По-друге, перш ніж стане можливим використовувати цей масив, його розмір необхідно привести у відповідність з розміром масиву `b`. Для цього можна скористатися методом `Resize` об'єкту

Array. Проте цей метод дозволяє змінювати кількість елементів такого масиву, для якого ця кількість вже визначена, тому перед викликом методу Resize створюється новий масив r, що складається з нуля елементів.

Варто відзначити також те, що перед першим параметром методу Resize знаходиться ключове слово ref, що говорить про те, що даний метод приймає посилання на масив r, – саме тому всередині методу стає можливою зміна параметрів самого масиву, а не його копії. Виклик методу можна здійснити таким чином:

```
Response.Write("Результати обчислення значень масиву:<br/>"); long [] result;
```

```
int [] data={2,3,4,5}; Stepen3(out result, 2, data); foreach (long i in result)
```

```
Response.Write(i.ToString()+"<br/>");
```

Результат роботи програми зображений на рис.2.4.

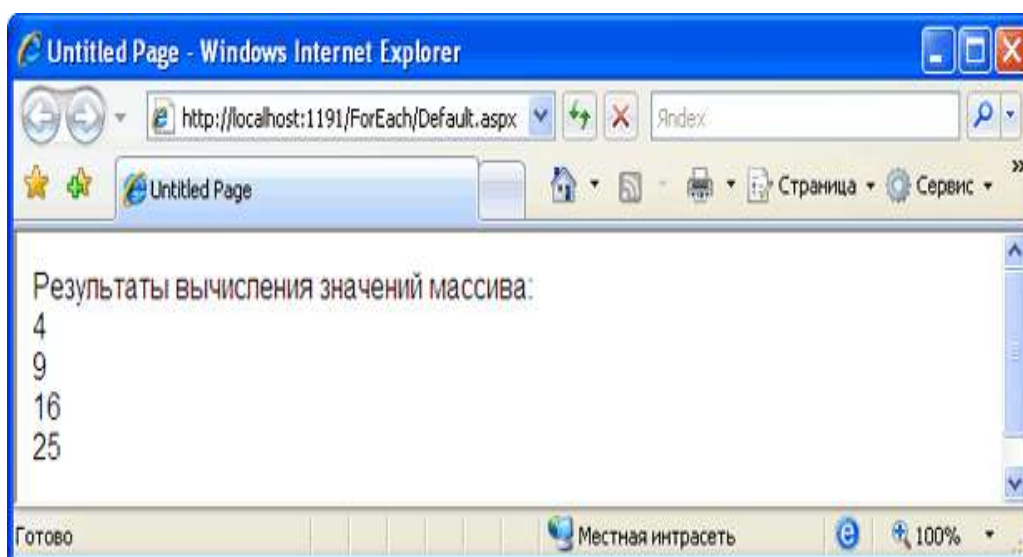


Рис. 2.4 – Результат роботи методу, що підносить до заданого степеня масив чисел

Наприкінці слід додати, що в об'єктно–орієнтованих мовах, таких як C#, основну роль грають посилальні типи, тому коли методу передається об'єкт посилального типу, всі поля цього об'єкту можуть мінятися в методі, тобто програмний код методу може дістати повний доступ до всіх полів об'єкту. Це відбувається не дивлячись на те, що формально об'єкт не є вихідним і не має ключових слів ref і out, тобто використовує семантику виклику за значенням. При такому виклику саме посилання на об'єкт залишається незмінним, але стан об'єкту, значення його полів можуть повністю змінитися. Така ситуація

достатньо поширена і є типовою, тому при роботі з посилальними типами даних ключові слова ref і out нечасто з'являються в описі аргументів методу.

Варіанти індивідуальних завдань

ЗАВДАННЯ 1:

Створити в Microsoft Visual Studio 2012 форму, що містить задану кількість полів для введення значень елементів числового або символічного масиву, і кнопку, при натисненні на яку виконується обробка масиву згідно з варіантом завдання:

1. Задати масив, що складається з 10 чисел. Знайти найменший з позитивних елементів масиву.
2. Задати масив з 10 елементів, що містить назви країн. Підрахувати кількість елементів масиву, що починаються на літеру «А».
3. Задати масив з 10 елементів, що містить назви країн. Підрахувати кількість елементів масиву, які закінчуються літерою «Я».
4. Задати масив, що складається з 10 цілих чисел. Підрахувати кількість невід'ємних парних елементів масиву.
5. Задати масив, що складається з 10 цілих чисел. Обчислити добуток всіх непарних від'ємних елементів масиву.
6. Задати масив з 10 елементів, що містить чоловічі і жіночі імена. Підрахувати кількість букв в щонайдовшому імені.
7. Задати масив з 10 елементів, що містить чоловічі і жіночі імена. Підрахувати кількість букв в щонайкоротшому імені.
8. Задати масив, що складається з 10 чисел. Обчислити суму елементів масиву, які більше заданого числа.
9. Задати масив, що складається з 10 чисел. Обчислити суму елементів масиву, які менше заданого числа.
10. Задати масив, що складається з 10 цілих чисел. Обчислити добуток всіх парних елементів масиву, не менших заданого числа.
11. Задати масив з 10 елементів, що містить назви рослин. Підрахувати, скільки разів у всіх елементах масиву зустрічається літера «ь».
12. Задати масив, що складається з 10 чисел. Визначити номер останнього невід'ємного елемента.
13. Задати масив, що складається з 10 цілих чисел. Знайти найменший з парних елементів масиву.

- 14.Задати масив з 10 елементів, що містить назви міст. Підрахувати кількість елементів масиву, що починаються на приголосну літеру.
- 15.Задати масив з 10 елементів, що містить назви міст. Підрахувати кількість елементів масиву, що починаються на голосну літеру.
- 16.Задати масив, що складається з 10 чисел. Обчислити суму елементів масиву, що мають парні номери.
- 17.Задати масив, що складається з 10 чисел. Обчислити суму елементів масиву, що мають непарні номери.
- 18.Задати масив з 10 елементів, що містить назви тварин. Підрахувати середню довжину елементів масиву.
- 19.Задати масив, що складається з 10 чисел. Підрахувати кількість елементів масиву, які рівні заданому числу.
- 20.Задати масив з 10 елементів, що містить назви фруктів. Підрахувати кількість елементів масиву, що не містять літери «ш».

ЗАВДАННЯ 2:

Створити в Microsoft Visual Studio 2012 форму, що містить задану кількість полів для введення значень елементів числового або символьного масиву, і кнопку, при натисненні на яку виконується обробка масиву згідно з варіантом завдання:

1. Даний рядок, що містить англійський текст. Знайти кількість слів, що починаються з букви «с».
2. Даний рядок. Підрахувати, скільки в ньому букв «а», «е», «я».
3. Даний рядок. Визначити, скільки в ньому символів « , » і пропуск.
4. Даний рядок, що містить текст. Знайти довжину найкоротшого слова і щонайдовшого слова.
5. Даний рядок, що містить текст. Знайти довжину першого слова і останнього слова.
6. Даний рядок символів, серед яких є двокрапка (:). Визначити, скільки символів йому передуює.
7. Даний рядок символів, серед яких є кома (,). Визначити кількість (,) та кількість символів між двома сусідніми (,).
8. Даний рядок, що містить текст, що закінчується крапкою. Вивести на екран слова, що містять менше п'яти букв.
9. Даний рядок, що містить текст, що закінчується крапкою. Вивести на екран слова, що містять чотири та більше літер.
- 10.Даний рядок. Перетворити його, видаливши кожну букву «и» і

повторивши кожну букву «а».

11. Даний рядок. Підрахувати кількість букв «и» в останньому його слові.
12. Даний рядок. Підрахувати, скільки різних букв зустрічається в ньому. Вивести їх на екран.
13. Даний рядок символів, серед яких є та, що відкривається і та дужка, що закривається. Вивести на екран всі символи, розташовані усередині цих дужок.
14. Є рядок, що містить букви латинського алфавіту і цифри. Вивести на екран довжину найбільшої послідовності цифр, що йдуть підряд.
15. Даний рядок. Вказати ті слова, які містять хоч би одну букву «ж».
16. Даний рядок. Знайти ті слова, які починаються і закінчуються однією і тією ж голосною буквою.
17. У рядку замінити всі знаки пропусків двокрапкою (:). Підрахувати кількість замін.
18. Визначити, скільки разів в рядку зустрічається задане слово.
19. У рядку є одна крапка з комою (;). Підрахувати кількість символів до крапки з комою і після неї.
20. Даний рядок. Знайти ті слова, які починаються і закінчуються однією і тією ж приголосною буквою.

Контрольні питання:

1. Дайте визначення масиву.
2. Як можна виконувати відразу і оголошення і ініціалізацію масиву у програмі?
3. Що таке розмірність масиву, які масиви бувають у C# ?
4. Як одержати доступ до певного елемента масиву?
5. Як у програмі організувати введення й вивід масиву?
6. Які основні методи виконує Клас Array?
7. Які операції дозволено над елементами масиву?
8. Від якого базового класу походять всі рядки в C#?
9. Перерахуйте основні методи класу System.String.
10. Особливості використання класу System.Text.StringBuilder.
11. Дайте визначення процедури і функції.
12. Поясніть різницю між процедурами й функціями.
13. Як здійснюється виклик процедури і функції?

14. Дайте визначення групам аргументів методів.

15. Коли краще використовувати процедури, а коли функції?

ЛАБОРАТОРНА РОБОТА №3
Основи роботи у середовищі Microsoft Visual Studio 2012.
Класи, структури та колекції C#

Мета роботи: Розробка додатків на мові C# у середовищі Microsoft Visual Studio .NET 2012. Вивчення і розгляд основних особливостей і базових принципів використання об'єктно–орієнтованого програмування в C#. Розгляд класів, опису їх полів, методів і властивостей, їх відмінності від структур, колекцій.

Постановка завдання: У середовищі Microsoft Visual Studio 2012 виконати Завдання №1, 2, 3 з використанням інструкції до виконання, яка наведена в теоретичній частині цієї лабораторної роботи. Проілюструйте виконання зазначених завдань екранними формами, які будуть включені до звіту.

Теоретичні відомості

Класи

Мова C# повною мірою є об'єктно–орієнтованою. З представленого вище матеріалу видно, що всі дії в C# є зверненнями до ресурсів якихось класів. Іноді для цього необхідно створювати об'єкти класів, іноді (при виклику статичних методів) створення об'єктів не потрібне.

Однією з найважливіших особливостей C# є те, що в ньому не можна створити процедуру або функцію, яка існує окремо, вона обов'язково повинна входити в який–небудь клас.

Клас – це шаблон, який визначає властивості і методи об'єктів, що створюються як екземпляри даного класу.

Основне призначення класу – створювати тип даних, який задає реалізацію деякої абстракції даних відповідно завдання, яке реалізує програмна система. Таким чином, класи мають важливу змістовну начинку, що відповідає за виконання своєї операції.

Визначення класу в C# практично ідентично визначенню такого ж класу в C++. Проте в C# існують додаткові елементи, що значно полегшують практичне використання класів. Розглянемо і засвоїмо основні правила побудови і використання класів, виконуючи послідовно завдання з наведених нижче прикладів.

Приклад 1

Для системи обробки банківських операцій клієнтів необхідно створити клас *Client*, що містить інформацію про ім'я, паспорт і дату народження позичальника, властивості для доступу до цих даних, а також метод, за допомогою якого можливо редагувати інформацію про клієнта.

Визначення класу починається з оголошення його імені, а також завдання його полів і необхідних методів (конструкторів). Визначення класу виглядає таким чином:

```
public class Client
{
private string Name; private string Passport;
private DateTime BirthDate;

public Client()
{
}
}
```

В наведеному кодї визначені поля класу, а також конструктор за замовчуванням (без параметрів). Для класів C# можна визначити будь-яку кількість конструкторів. Якщо в класі не визначений жоден конструктор, використовується конструктор за замовчуванням, який при створенні об'єкту автоматично привласнить всім змінним – членам класу безпечні значення. В даному випадку, конструктор за замовчуванням задається явно.

Зона видимості полів класу відповідно до правил має бути визначена як *закрита* або як *захищена* (у випадку, якщо передбачається спадкоємність його полів). Доступ до полів – членів класу має бути організований або за допомогою методів, або за допомогою властивостей класу. Створимо властивості класу *Client*, який забезпечує читання і запис значень полів класу.

```
public string passport
{
get
{
return Passport;
}
set
{
```

```

    Passport = value;
}
}
public string name
{
    get
    {
        return Name;
    }
    set
    {
        Name = value;
    }
}
public int age
{
    get
    {
        int a;
        a = DateTime.Now.Year - BirthDate.Year; return a;
    }
}
public DateTime birthdate
{
    get
    {
        return BirthDate;
    }
    set
    {
        if (DateTime.Now > value) BirthDate = value;
        else
            throw new Exception("Введена невірна дата народження");
    }
}
}

```

Як видно з даного прикладу, властивість складається з методів set і get.

При цьому властивість повинна містити хоча б один з методів.

Set дозволяє змінювати значення поля класу, get – набувати значення. У метод Set передається значення параметра за допомогою змінної value. Обидва методи можуть містити довільну кількість операторів, що описують алгоритм виконання дій в процесі читання або запису значення в полі класу. В даному прикладі властивості passport і name дозволяють просто дістати доступ до полів класу шляхом читання або встановлення значення відповідних змінних.

Властивість birthdate також призначена для читання і запису значення змінної – члена класу BirthDate. При цьому при читанні значення (операція get) відбувається просто передача значення змінній BirthDate, при спробі ж запису нового значення в цю змінну відбувається перевірка допустимості встановлення значення змінної. В даному випадку перевірка зводиться до порівняння нового значення дати народження з поточною датою. Якщо нове значення дати народження більше або дорівнює поточній даті, генерується виключення, яке не дозволяє записати нове значення в змінну – член класу.

Властивість age застосовується для отримання поточного віку клієнта. Вона призначена тільки для читання значення змінної, тому містить лише метод get. При використанні властивості age відбувається обчислення поточного значення віку клієнта в роках шляхом віднімання року народження від поточного значення року. Використання властивостей аналогічно використанню змінних.

Приклад 2

Створити об'єкт c1 класу Client. Поля даного об'єкта заповнити значеннями з використанням властивостей. Організувати на екран вивід значень полів, для цього також застосовуються властивості класу:

```
Client c1=new Client(); c1.name = "Іванов"; c1.passport = "9002";  
c1.birthdate = new DateTime(1967, 08, 03); Response.Write("Імя=" +  
c1.name+"</br>"); Response.Write("Паспорт=" + c1.passport+"</br>");  
Response.Write("Возраст="+c1.age);
```

Окрім змінних – членів класу і властивостей для доступу до цих змінних практично будь-який клас містить методи, що призначені для опису алгоритмів виконання дій класом. Один з обов'язкових методів класу – конструктор за замовчуванням.

Проте класи в C# можуть містити довільну кількість конструкторів, які призначені для ініціалізації об'єкта даного класу.

Створимо конструктор з параметрами, що дозволяє ініціалізацію об'єкта

класу, вводячи значення полів цього об'єкту як параметри конструктора. Зробити це можна таким чином:

```
public Client(string ClientName, string ClientPassport, DateTime
ClientBirthDate)
{
    name = ClientName; passport = ClientPassport; birthdate =
ClientBirthDate;
}
```

Видно, що конструктор містить три параметри. В тілі конструктора відбувається запис переданих, як параметри значень, у відповідні поля класу за допомогою властивостей даного класу. У випадку з датою народження це дозволяє не дублювати процедуру перевірки введеної дати, а скористатися алгоритмом, реалізованим у властивості birthdate. Створимо також метод, що дозволяє змінити значення полів об'єкта класу Client:

```
public void EditClient(string ClientName, string ClientPassport,DateTime
ClientBirthDate)
{
    Name = ClientName; Passport = ClientPassport; birthdate =
ClientBirthDate;
}
```

Як видно з прикладу, початковий код цього методу практично повністю ідентичний конструктору з параметрами з різницею в імені, а також в типі значення, яке повертається. Звичайно, в даному випадку можна було б обійтися і використанням властивостей для зміни значень полів класу, проте, іноді буває корисно, щоб такого роду зміни були реалізовані в рамках одного методу, тим більше, якщо алгоритм змін є нестандартним.

Тепер, з використанням конструктора з параметрами, можна створити і відразу ж ініціалізувати об'єкт класу Client:

```
Client c1=new Client("Іванов","9002",new DateTime(1967,08,03));
```

Структури

У багатьох відношеннях, структури можна розглядати як особливий різновид класів. Для структур можна визначати конструктори, реалізовувати інтерфейси. Проте для структур в C# не існує базового класу, тому всі структури є похідними від типу ValueType.

Простий приклад структури можна представити так:

```

public struct Employee
{
    public string name; public string type; public int deptID;
}

```

Використання структури можливе таким чином. Спочатку її необхідно створити. В момент створення структури для неї виділяється пам'ять в області стека. Надалі до елементів структури можливе звернення шляхом вказівки імені структури і імені елемента, що розділені крапкою:

```

Employee Alex; Alex.name = "Alex"; Alex.type = "manager"; Alex.deptID = 2310;

```

У реальній системі для зручнішого використання структур доцільно визначити конструктор або декілька конструкторів. При цьому необхідно пам'ятати, що в структурах неможливо перевизначити конструктор за замовчуванням (без параметрів). Всі визначені в структурі конструктори повинні приймати параметри. Нижче представлений приклад конструктора структури.

```

public Employee(int DEPTID, string Name, string EmpType)
{
    deptID = DEPTID;
    type = EmpType; name = Name;
}

```

Для виклику конструктора структури необхідно використовувати ключове слово `new`. Використання конструктора виглядає таким чином:

```

Employee Nick=new Employee(278,"Nick","worker");

```

Аналогічним чином стає можливим створення і використання методів структур.

Колекції

Раніше був розглянутий клас `Array`, в якому реалізований цілий набір операцій над елементами масиву, таких як сортування, клонування, перерахування і розстановка елементів в зворотному порядку. Проте при роботі з масивами виникає і цілий ряд інших завдань. Вирішенням цих завдань і займаються колекції, що дозволяють організовувати елементи спеціальним чином і проводити згодом над ними певний набір операцій. Всі визначені в .NET Framework колекції розташовані в просторі імен `System.Collections`. В табл. 3.1 представлені колекції цього простору імен, які найчастіше використовуються.

Колекції простору імен System.Collections

Колекція	Призначення колекції
ArrayList	Масив об'єктів, що динамічно змінює свій розмір.
Hashtable	Набір взаємозв'язаних ключів і значень, заснованих на хеш-кодів ключа.
Queue	Стандартна черга, реалізована за принципом "першим увійшов, першим вийшов" (FIFO – First In First Out).
SortedList	Представляє клас, елементами якого можуть бути пари "ключ-значення", відсортовані за значенням ключа, які надають доступ до елементів по їх порядковому номеру (індексу).
Stack	Черга, реалізована за принципом "першим увійшов, останнім вийшов" (LIFO – Last In First Out).

У просторі імен System.Collections.Specialized розташовані класи, які призначені для використання в спеціальних випадках. Так, клас StringCollection представляє набір рядків. Класом StringDictionary є набір строкових значень, що складаються з пар "ключ-значення".

Розглянемо застосування колекцій на прикладі класу ArrayList. Для цього скористаємося розглянутим раніше класом Client. Дуже часто виникає ситуація, при якій необхідно буває поміщати об'єкти в список або масив для їх подальшої обробки. Звичайно, при цьому може бути достатньо тих можливостей, які надають стандартні масиви C#, розглянуті раніше. Проте колекції, володіють ширшими можливостями, отже, їх застосування переважне в порівнянні з масивами.

Приклад 3

Для обробки даних про клієнтів створити клас Clients, який призначений для зберігання списку клієнтів і його обробки. При цьому необхідно врахувати, що клас Clients повинен забезпечувати можливість додавання, видалення і нумерації елементів (клієнтів).

Для реалізації цієї функціональності необхідно використовувати клас ArrayList, як вкладений в клас Clients. Таким чином, необхідно визначити в даному класі елемент, що дозволяє вести список клієнтів, а також реалізувати набір відкритих методів, які передаватимуть виклики на виконання різних дій внутрішньому класу, похідному від ArrayList. Приклад початкового коду класу Clients приведений нижче:

```

public class Clients
{
private ArrayList ClientsList; public Clients()
{
ClientsList=new ArrayList();
}
public int ClientsCount
{
get
{
return ClientsList.Count;
}
}
public void AddClient(Client c)
{
ClientsList.Add(c);
}
public void RemoveClient(int ClientToRemove)
{
ClientsList.RemoveAt(ClientToRemove);
}
public Client GetClient(int CLIENTID)
{
return (Client) ClientsList[CLIENTID];
}
}

```

Використання такого класу представляється дуже простим:

```

Clients cl=new Clients();
cl.AddClient(new Client("Судопов", "9002", new DateTime(1980, 12, 21)));
cl.AddClient(new Client("Петров", "9004", new DateTime(1975, 03, 15)));

```

Тепер стає можливим звернення до будь-якого класу Client, який поміщений в колекцію Clients за допомогою методу GetClient:

```

Client MyClient = cl.GetClient(1); Response.Write(MyClient.name);

```

Результатом роботи цього фрагмента програми буде рядок "Петров".

Проте існує декілька незручностей від використання класу Clients в представленій реалізації. По–перше, характерним зверненням до елемента масиву або списку в мові С# є використання індексаторів, які пишуться в квадратних дужках. Для реалізації цієї можливості в класі Clients необхідно створити індексатор:

```
public Client this[int pos]
{
    get
    {
        return ((Client)ClientsList[pos]);
    }
    set
    {
        ClientsList[pos]= value;
    }
}
```

Від звичайного методу індексатор відрізняється ім'ям (this), а також наявністю квадратних дужок, в яких вказується параметр – номер зазначеного елемента. За допомогою індексатора стає можливим виділення елемента з колекції, а також запис елемента в колекцію під певним номером:

```
Response.Write(cl[1].name);
cl[1].name = "Сидоров"; Response.Write(cl[1].name);
```

Тепер можна організувати обробку елементів масиву в циклі. При роботі з колекціями досить зручною є можливість використання циклу foreach. Для реалізації такої функціональності при роботі з класом Clients необхідно використовувати метод GetEnumerator() інтерфейсу IEnumerable – задати спадкоємність класом Clients інтерфейсу IEnumerable і реалізувати метод, як показано нижче.

```
public class Clients:IEnumerable
{
    public IEnumerator GetEnumerator()
    {
        return ClientsList.GetEnumerator();
    }
}
```


Приклад застосування циклу `foreach` з використанням класу `Clients` показаний нижче.

```
foreach (Client c in cl)
{
    Response.Write("Імя="+c.name+"");
    Response.Write("Паспорт="+c.passport+" "); Response.Write("Возраст=" +
    c.age); Response.Write("</br>");
}
```

Створіть клас `Client` та об'єкт `c1` класу `Client` по аналогії з прикладами 1, 2, 3. Поля даного об'єкта заповнити значеннями з використанням властивостей. Організувати на екран вивід значень полів, для цього також застосовуються властивості класу. Для обробки даних про клієнтів створити клас `Clients`, який призначений для зберігання списку клієнтів і його обробки. При цьому необхідно врахувати, що клас `Clients` повинен забезпечувати можливість додавання, видалення і нумерації елементів (клієнтів). Згідно з варіантом об'єкти початкові умови для створення класів та об'єктів в табл. 3.2.

Таблиця 3.2

Варіанти індивідуальних завдань

№	Властивості класу Client	Поля, які виводяться на екран	Функція обробки полів
1	Прізвище, Ім'я, Дата народження, Сума договору	Прізвище, Сума договору	Обчислити середню суму за договорами клієнтів
2	Індивідуальний код, Прізвище, Дата договору	Прізвище, Дата договору	Визначити клієнта, який має договір із давнішньою датою
3	Логін, Пароль, Номер паспорту	Логін, Пароль, Номер паспорту	Визначити клієнта, який має найкоротший логін
4	Логін, пароль, Дата народження	Логін, Дата народження	Визначити наймолодшого клієнта
5	Прізвище, Ім'я, Дата народження	Прізвище, Ім'я, Дата народження	Визначити найстаршого клієнта
6	Прізвище, Ім'я, Дата реєстрації	Прізвище, Дата реєстрації	Визначити клієнта, який зареєструвався першим

№	Властивості класу Client	Поля, які виводяться на екран	Функція обробки полів
7	Логін, пароль, Дата реєстрації	Логін, Дата реєстрації	Визначити клієнта, який зареєструвався останнім
8	Прізвище, Ім'я, Дата народження, Сума першого договору, Сума останнього договору	Прізвище, Сума першого договору	Визначити клієнта, який має найбільшу суму першого договору
9	Прізвище, Ім'я, Дата народження, Сума першого договору, Сума останнього договору	Прізвище, Сума останнього договору	Визначити клієнта, який має найбільшу суму останнього договору
10	Ім'я, Логін, Дата народження, Дата реєстрації, Кількість договорів	Ім'я, Кількість договорів	Визначити клієнта, який має найбільшу кількість договорів
11	Ім'я, Логін, Дата народження, Кількість договорів	Логін, Дата реєстрації, Кількість договорів	Визначити клієнта, який має найменшу кількість договорів
12	Індивідуальний код, Прізвище, Ім'я, Дата договору	Ім'я, Дата договору	Визначити клієнта, який має договір із датою, максимально наближеною до поточної дати
13	Логін, Пароль, Дата народження, Номер паспорту	Логін, Дата народження, Номер паспорту	Визначити клієнта, який має найдовший логін
14	Ім'я, Логін, Дата народження, Дата реєстрації, Кількість договорів, Сума останнього договору	Ім'я, Дата реєстрації, Кількість договорів	Визначити клієнта, який має парну кількість договорів

№	Властивості класу Client	Поля, які виводяться на екран	Функція обробки полів
15	Ім'я, Логін, Дата народження, Дата реєстрації, Кількість договорів, Сума останнього договору	Ім'я, Дата реєстрації, Кількість договорів, Сума останнього договору	Визначити клієнта, який має непарну кількість договорів
16	Логін, пароль, Дата реєстрації, Поточна заборгованість	Логін, Дата реєстрації, Поточна заборгованість	Визначити клієнта, який має найбільшу заборгованість
17	Ім'я, Логін, Дата народження, Поточна заборгованість	Ім'я, Поточна заборгованість	Визначити клієнтів, які не мають заборгованості
18	Прізвище, Дата реєстрації, Кількість договорів, Поточна заборгованість	Прізвище, Дата реєстрації, Поточна заборгованість	Визначити клієнта, який має найменшу заборгованість
19	Ім'я, Прізвище, Номер паспорту, Кількість договорів, Поточна заборгованість	Ім'я, Прізвище, Номер паспорту, Кількість договорів, Поточна заборгованість	Визначити клієнтів, які мають заборгованості
20	Логін, Пароль, Номер паспорту, Кількість договорів, Поточна заборгованість	Логін, Пароль, Номер паспорту, Кількість договорів, Поточна заборгованість	Визначити клієнтів, які не мають договорів

Контрольні питання:

1. Які типи даних описуються за допомогою класів C# ?
2. Сформулюйте правила передачі параметрів в процедури і функції.
3. Що таке клас? Основне призначення класів C#?

4. Як відбувається визначення класу? Призначення полів і методів класу.
5. Що таке структура? Надайте приклад використання структур у C#.
6. Які колекції розташовані в просторі імен System.Collections?
7. Вкажіть різницю між методом та індексатором класу. Для чого використовують індексатори класів?

ЛАБОРАТОРНА РОБОТА №4

Робота з базами даних MS SQL Server в Microsoft Visual Studio 2012

Мета роботи: побудувати Web–додаток, який ефективно взаємодіє з базою даних в середовищі Microsoft Visual Studio .NET 2012.

Постановка завдання: в середовищі Microsoft Visual Studio 2012 виконати Завдання №1, 2 з використанням інструкції до виконання, яка наведена в теоретичній частині цієї лабораторної роботи. Проілюструйте виконання зазначених завдань екранними формами, які будуть включені до звіту.

Теоретичні відомості

Діяльність більшості Web–додатків зосереджена на відображенні та модифікації даних. Ці завдання здаються достатньо прямолінійними, але за останнє десятиліття способи використання даних постійно мінялися. Розробники перейшли від простих клієнтських застосувань з локальними базами даних до розподільних систем, які засновані на централізованих базах даних і виділених серверах. В той же час розвивалися технології доступу до даних. Сьогодні понад 95% всіх інформаційних систем так чи інакше використовують бази даних.

Для того, щоб додаток .NET міг здійснювати взаємодію з джерелом даних, необхідно встановити з'єднання з ним. Найбільш типовим сценарієм роботи Web–додатку є наступний:

- встановлюється з'єднання, відкривається підключення до бази даних;
- виконується один або декілька запитів, що здійснюють внесення змін до наборів даних, а також вибірки даних з БД;
- здійснюється відключення від джерела даних. При цьому користувач працює з набором даних, проглядаючи його, виконуючи фільтрацію, вносячи зміни і так далі;
- при необхідності перенесення змін з набору даних в БД, а також при необхідності розглядання змін, які внесені в БД іншими користувачами;
- здійснюється підключення до джерела даних, виконуються необхідні дії, після чого проводиться відключення від БД.

Приклад 1

Створити базу даних безпосередньо в середовищі Microsoft Visual Studio 2012 за інструкцію, яка наведена нижче.

1) В меню оберіть команду Файл/Создать/ Веб-сайт.(File/New Web Site). У діалоговому вікні, що відкрилося, задайте шаблон: Пустой Веб-сайт ASP.NET, Language: Visual C# і шлях до каталогу, де буде збережений проєкт (рис. 4.1).

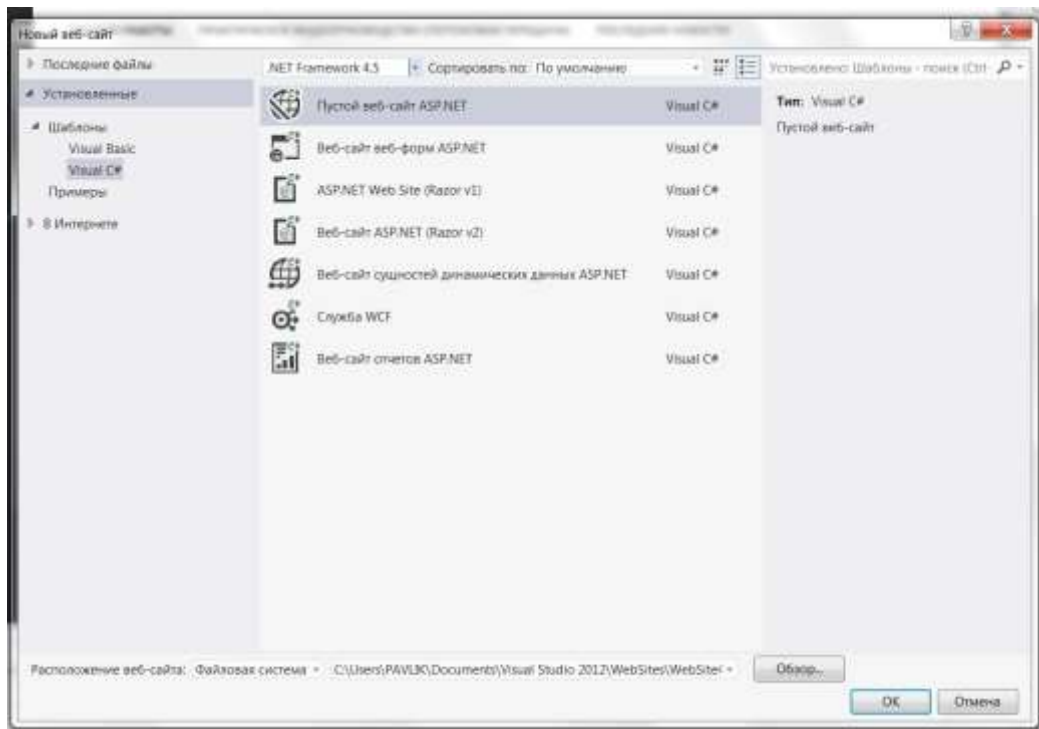


Рис. 4.1 – Створення Web-додатка

2) В меню оберіть команду Веб-сайт/Добавить новый элемент (Website/Add New Item). В діалоговому вікні, що відкрилося, оберіть шаблон База данных SQL Server і задайте ім'я майбутньої БД, наприклад, Test.mdf. При цьому з'являється нове джерело даних у вікні Server Explorer (рис.4.2).

3) У вікні Обзор серверов (Server Explorer) у Test.mdf вибираємо Таблицы (Tables), натискаємо праву кнопку миші і з меню, що з'явилося, вибираємо пункт Добавить новую таблицу (Add New Table). З'являється конструктор таблиць, де задаються імена полів таблиць, типи даних і обмеження цілісності. Створимо наступні таблиці, що містять інформацію про студентів учбового закладу(рис. 4.3).

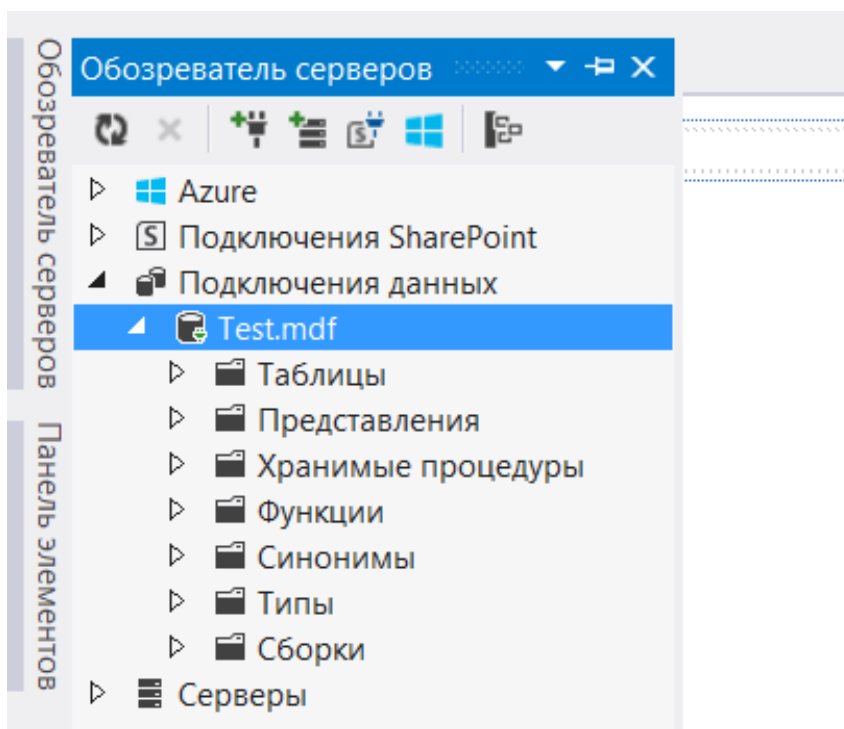


Рис. 4.2 – Створення бази даних

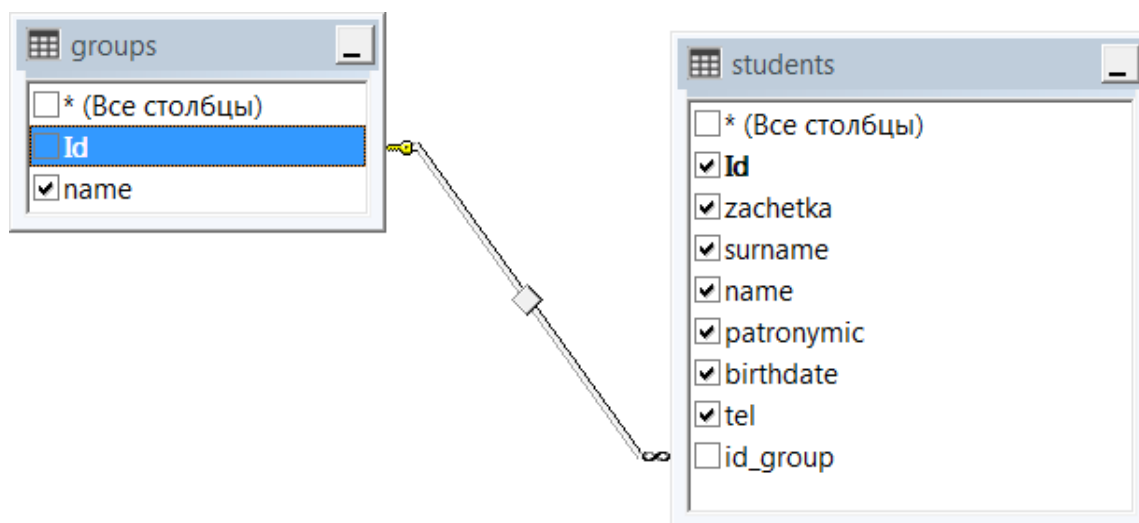


Рис.4.3 – Таблиці БД, що містять інформацію про студентів учбового закладу

Спочатку виконаємо створення таблиці groups шляхом заповнення стовпців Імя (Column Name), Тип даних (Data Type) і Доступные значения ноль (Allow Nulls) необхідними значеннями імен полів, типів даних і наявності порожніх значень відповідно. Для завдання первинного ключа в рядку, що відповідає полю id, клацаємо правою кнопкою миші і вибираємо Set Primary Key (рис.4.4).

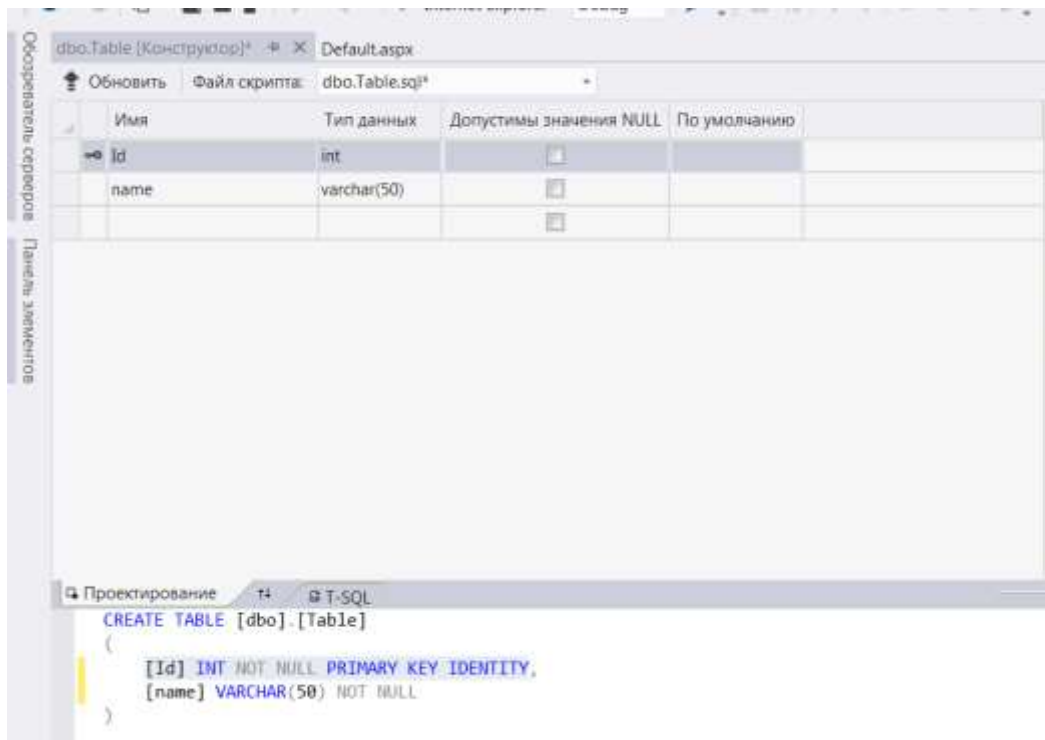


Рис.4.4 – Таблица groups БД, що містить інформацію про групи студентів учбового закладу

Для того, щоб значення первинного ключа формувалися автоматично шляхом збільшення попереднього значення на 1, необхідно, знаходячись на рядку, що відповідає полю id, розкрити в Свойства (Column Properties) пункт Спецификация идентификатора (Identity specification) і додати властивостям Идентификатор (Is Identity), Начальное значение (Identity Increment) і Шаг приращения (Identity Seed) значення Yes, 1, і 1 відповідно (рис.4.5).

Після цього зберегти таблицю groups, для цього необхідно назвати таблицю "groups" шляхом оновлення першого рядка в області скриптів, як показано в наступному прикладі:

CREATE TABLE [dbo].[groups]

У лівому верхньому кутку конструктора таблиць натисніть кнопку Оновити (Update), як показано на рис.4.6.

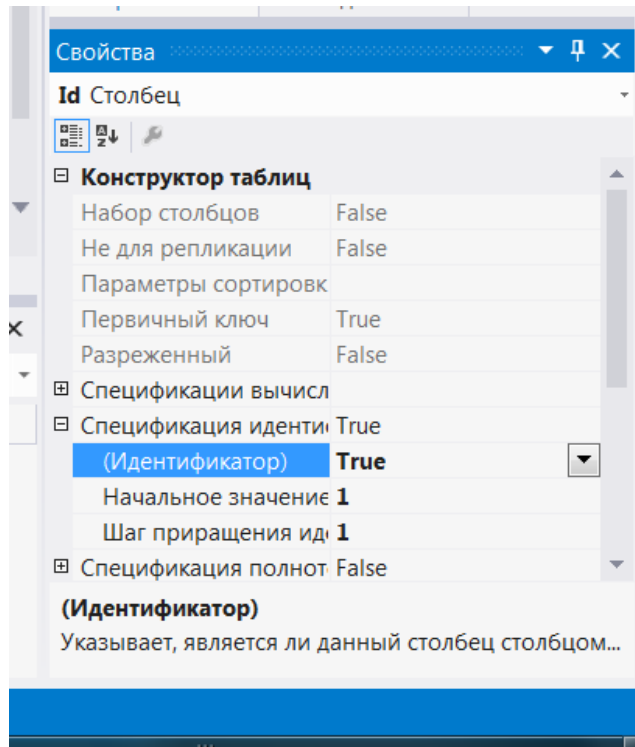


Рис.4.5 – Встановлення властивостей таблиці groups

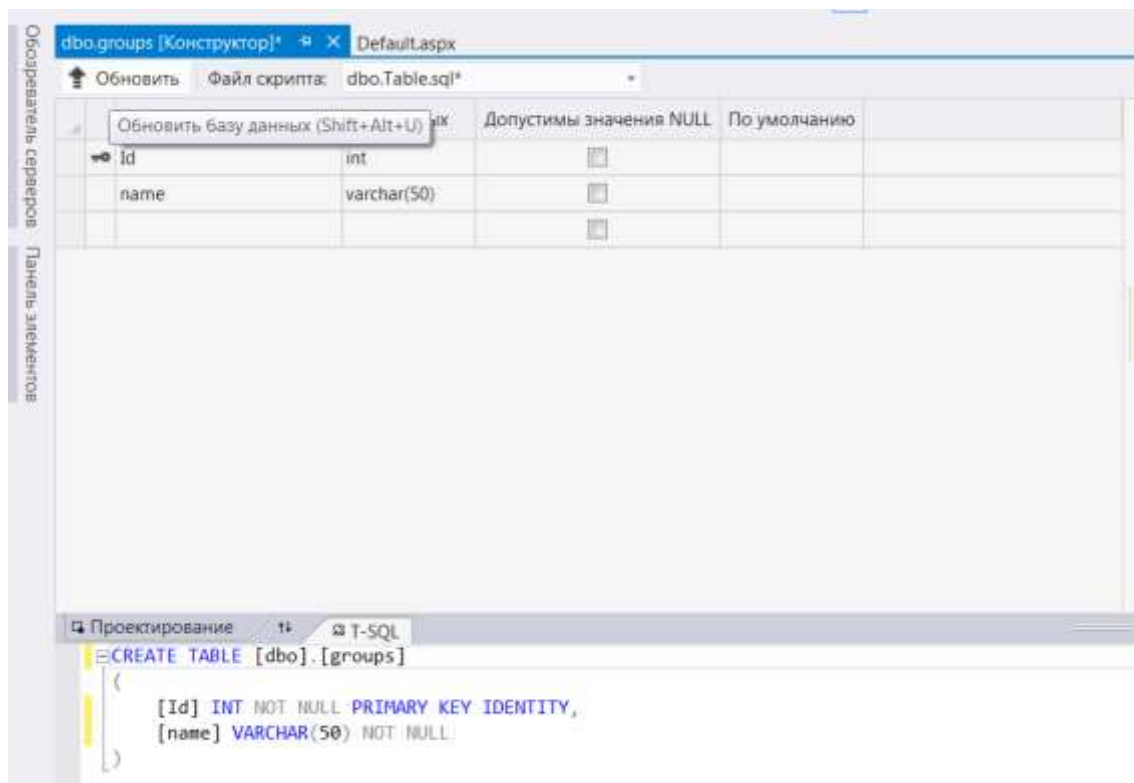


Рис. 4.6 – Зберігання таблиці groups у БД

У діалоговому вікні Предварительный просмотр обновления базы данных натисніть кнопку Обновить базу данных. Зміни зберігаються в файл локальної бази даних Test.mdf. Аналогічно створюємо таблицю students.(рис.4.7).

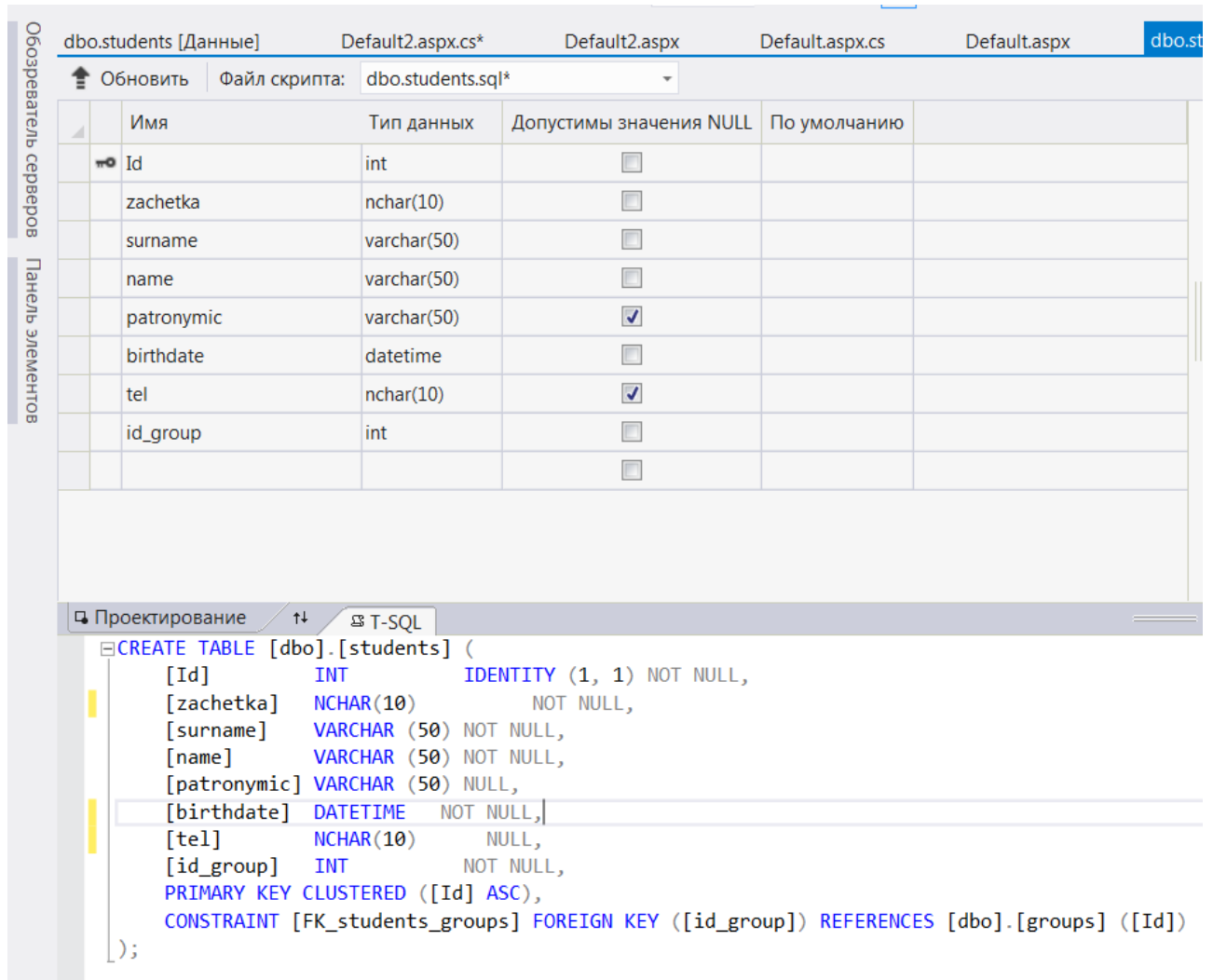


Рис. 4.7 – Зберігання таблиці students у БД

4) Створюємо зв'язок між таблицями. Для створення зовнішнього ключа необхідно в області контексту конструктора з таблицею students в правій частині сітки відкрити контекстне меню елемента Внешние ключи і вибрати команду Добавить новый внешний ключ, як показано на рис.4.8. В текстовому полі, що з'явилося, замініть ToTable на groups (рис.4.9).

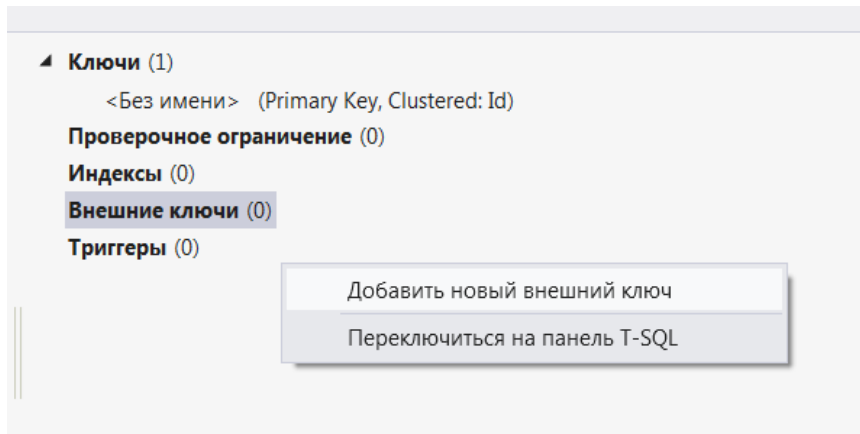


Рис. 4.8 – Створення зовнішнього ключа таблиці students

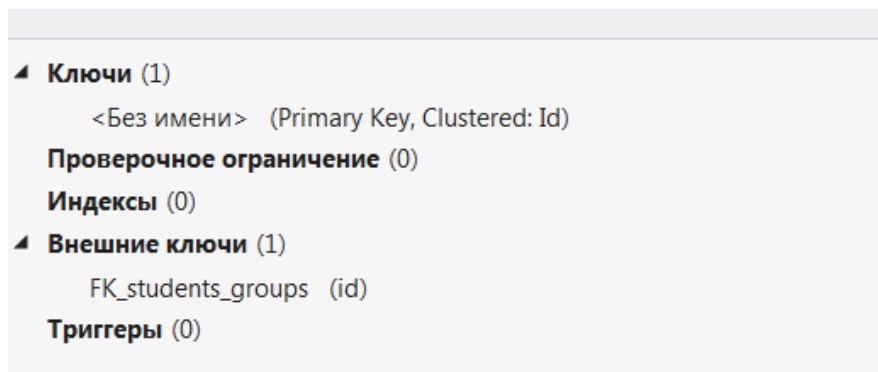


Рис. 4.9 – Зовнішній ключ таблиці students

В області скриптів необхідно поновити останній рядок наступним чином:

CONSTRAINT [FK_students_groups] FOREIGN KEY ([id_group]) REFERENCES [dbo].[groups] ([Id])

В лівому верхньому кутку конструктора таблиць натисніть кнопку **Обновить**. В діалоговому вікні **Предварительный просмотр обновления базы данных** натисніть кнопку **Обновить базу данных**. Зміни зберігаються у файл локальної бази даних.

5) Для заповнення таблиць даними необхідно у вікні **Server Explorer** клацнути правою кнопкою миші на імені таблиці і в меню, що з'явилося, вибрати пункт **Показать таблицу данных (Show Table Data)**. У вікні, що відкрилося, можна вводити дані в поля таблиці.

Приклад 2

Створити додаток для роботи з базою даних у середовищі Visual Studio 2012 за інструкцією, яка наведена нижче.

1) Якщо БД створюється разом із створенням проєкту, то даний крок можна пропустити. Інакше у вікні Server Explorer вибираємо елемент Data Connections (підключення до даних), клацаємо правою кнопкою миші і з меню вибираємо пункт Add Connection. У вікні, що з'явилося, вибираємо раніше створену БД за допомогою кнопки Browse, або вписуємо шлях до неї.

2) Спочатку створимо сторінку для роботи з даними таблиці groups. Для цього відкриваємо Default.aspx у режимі дизайну. Перетягуємо з Server Explorer таблицю groups на форму. При цьому повинні відобразитися 2 елементи: GridView і SqlDataSource. Слід звернути увагу на маленьку кнопку з направленою вправо стрілкою в правому верхньому кутку елементу GridView. Клацання на ній викликає панель редагування, що дозволяє форматувати елемент GridView і задавати деякі його властивості. Активуємо можливість редагування і видалення даних в таблиці шляхом визначення пунктів Enable Editing і Enable Deleting.

3) Компілюємо і запускаємо додаток. Він є повністю працездатним і дозволяє редагувати і видаляти інформацію в таблиці.

4) Додамо на форму елементи управління, які дозволять додавати дані в таблицю groups. На форму поміщаємо елемент управління Label і його властивості Text надаємо значення New group name. Ідентифікатор залишаємо без зміни – Label1. На форму поміщаємо елемент управління TextBox і його ідентифікатор залишаємо без зміни – TextBox1. На форму поміщаємо елемент управління Button і його властивості Text надаємо значення Insert. Ідентифікатор залишаємо без зміни – Button1.

5) Двічі клацаємо на кнопці і в редакторі коду вписуємо наступні фрагменти:

- для роботи з даними MS SQL Server необхідно додати простір імен *System.Data.SqlClient*:

```
using System.Data.SqlClient;
```

- в обробнику натискання кнопки пишемо:

```
protected void Button1_Click(object sender, EventArgs e)
```

```
{
```

```
{
```

```
// зчитуємо назву групи в змінну name
```

```
String name = TextBox1.Text;
```

```

// значення назви групи обов'язково повинно бути введено
if (name == "") return;
// запит на додавання даних
String query = "INSERT INTO groups (name) VALUES ('" + name + "')";
//виконання запиту на додавання даних
SqlDataSource1.InsertCommand = query; SqlDataSource1.Insert();
//очищення поля вводу
TextBox1.Text = "";
}
}

```

6) Компілюємо і запускаємо додаток. Тепер він дозволяє не тільки редагувати і видаляти інформацію в таблиці, але і додавати нові дані.

7) Створення сторінки для роботи з даними таблиці students проводиться аналогічно. Відкриваємо Default.aspx у режимі дизайну (або створюємо ще одну Web-форму: вибираємо меню File/New File, потім в діалоговому вікні, що з'явилося, вибираємо шаблон Web Form і задаємо його ім'я). Перетягуємо з Server Explorer таблицю students на форму. При цьому повинні відобразитися елементи GridView і SqlDataSource. Активуємо можливість редагування і видалення даних в таблиці шляхом визначення пунктів Enable Editing і Enable Deleting.

8) Додаємо елементи управління, які дозволять додавати дані в таблицю students. На форму поміщаємо 7 елементів управління Label, які відповідають таким полям таблиці students, як zchetka, surname, name, patronymic, birthdate, tel і id_group, і їх властивостям Text надаємо відповідні значення (наприклад, New group або New telephone number). Ідентифікатори можна залишити без зміни – Label1...Label7. На форму поміщаємо 6 елементів управління TextBox, які відповідають таким полям таблиці students, як zchetka, surname, name, patronymic, birthdate і tel. Їх ідентифікатори залишаємо без зміни – TextBox1...TextBox6.

9) Для відображення можливих значень поля id_group необхідно додати на форму елемент управління DropDownList – список, що розкривається, який містить значення внесені в таблицю groups. Також необхідно перетягнути на форму з Server Explorer таблицю groups і властивості Visible елементу управління GridView надати значення False. Змінимо ідентифікатор елементу управління SqlDataSource із заданого за замовчуванням SqlDataSource2 на SqlGroups, а для відповідного елементу управління GridView значення

властивості DataSourceID міняємо з SqlDataSource2 на SqlGroups.

При відображенні на формі елементу управління DropDownList з'являється панель редагування, за допомогою якої можна вибрати джерело даних для цього елементу управління. Вибираємо на панелі редагування пункт Choose Data Source. Оскільки дані для заповнення списку, що розкривається, містяться в таблиці groups, у вікні майстра налаштування джерела даних, що з'явилося, в полі Select a data source вибираємо SqlGroups, в полі Select a data field to display in the DropDownList вписуємо name (значення якого поля таблиці groups відображатимуться), а в полі Select a data field for the value of DropDownList вписуємо id (значення якого поля таблиці groups вноситимуться до поля id_group таблиці students).

10) На форму поміщаємо елемент управління Button і його властивості Text надаємо значення Insert. Ідентифікатор залишаємо без зміни – Button1.

11) Двічі клацаємо на кнопці і в редакторі коду вписуємо наступні фрагменти:

- для роботи з даними MS SQL Server необхідно додати простір імен *System.Data.SqlClient*:

```
using System.Data.SqlClient;
```

- в обробнику натиснення кнопки пишемо:

```
protected void Button1_Click(object sender, EventArgs e)
```

```
{
```

```
//зчитуємо дані про номер залікової, прізвище, ім'я, по батькові,
```

```
// дату народження, адресу, номер телефону студента
```

```
//в змінні z, s, n, p, b1, a, t відповідно
```

```
String z = TextBox1.Text;
```

```
String s = TextBox2.Text;
```

```
String n = TextBox3.Text;
```

```
String p = TextBox4.Text;
```

```
String b1 = TextBox5.Text;
```

```
String t = TextBox6.Text;
```

```
//зчитуємо дані про групу із списку в змінну g
```

```
int g = Convert.ToInt16(DropDownList1.SelectedValue);
```

```
// перевіряємо наявність введенні всіх обов'язкових даних
```

```
if (z == "" || s == "" || n == "" || b1 == "") return;
```

```
// конвертуємо дату народження в тип DateTime
```

```
DateTime b = Convert.ToDateTime(b1);
```

```

// запит на додавання даних
String query = "INSERT INTO students
(zachetka,surname,name,patronymic,birthdate,tel,id_group) VALUES ('" + z +
"',"' + s + "',"' + n + "',"' + p + "',"' + b + "',"' + t + "'," + g + ")";
// виконання запиту на додавання даних
SqlDataSource1.InsertCommand = query; SqlDataSource1.Insert();
//очищення полів вводу
TextBox1.Text = "";
TextBox2.Text = "";
TextBox3.Text = "";
TextBox4.Text = "";
TextBox5.Text = "";
TextBox6.Text = "";

}

```

12) Компілюємо і запускаємо додаток.

Варіанти індивідуальних завдань

Створити базу даних та додаток для роботи з нею, використовуючи дані табл. 3.2 лабораторної роботи №3.

Контрольні питання:

1. Описати серверні елементи управління, які були використані в лабораторній роботі.
2. Яка інформація відображається у вікні оглядача серверів?
3. Як створюється зв'язок між таблицями БД ?
4. Призначення, властивості, події веб-елементу управління GridView.
5. Призначення, властивості, події веб-елементу управління SqlDataSource.

ЛАБОРАТОРНА РОБОТА №5

Вивчення серверних WEB- елементів управління

Мета роботи: Вивчити класи елементів управління ASP .NET і виконати порівняння з HTML, вивчити порядок роботи з дизайнером форм, розглянути елементи управління – кнопки, зображення, календар.

Постановка завдання: у середовищі Microsoft Visual Studio 2012 виконати завдання 5.1 – 5.6 з використанням інструкції до виконання, яка наведена в теоретичній частині лабораторної роботи. Проілюструйте виконання зазначених завдань екранними формами з вказаними серверними елементами управління. Дизайн Web-форми і властивості серверних елементів управління мають бути індивідуальні для кожного студента.

Теоретичні відомості

Одне з найважливіших завдань web-обробника – отримання і обробка даних, які вводить користувач. Інформація посилається серверу через форму. Форма містить елементи управління, які дозволяють різними способами вводити інформацію.

Форми застосовуються в більшості сайтів. Наприклад, якщо користувач пише лист в web-інтерфейсі, з'являється форма з текстовими полями, що відповідають темі і тексту листа. Натисненням на кнопку можна додати файл і остаточно відправити лист кнопкою Send. Форма HTML містить теги, такі як текстове поле, список, перемикачі (radiobuttons) і прапорці (checkbox), кнопки.

Форми ASP.NET відрізняються від звичайних форм наявністю властивості `runat="server"`. Вони обробляються ASP.NET на сервері. Форми є одним з полів сторінки. На сторінці знаходяться елементи управління, які мають бути розташовані всередині форми. ASP.NET дозволяє запам'ятовувати стан елементів управління, тобто текст, який був введений в текстове поле, або вибраний перемикач, передати його на сервер і назад на сторінку після її оновлення:

```
<form ID="FormVote" runat="server"></form>
```

Всі форми обробляються методом POST. Атрибут Action можна задавати, але не обов'язково. За замовчуванням це поточна сторінка. У елементів управління ASP.NET теж є властивість `runat="server"`. Другий

обов'язковий атрибут – це його ідентифікатор або ID. Він потрібний, щоб звертатися до елемента в програмі, тобто це ім'я структурного елемента сторінки. ASP.NET пропонує безліч серверних елементів управління, які діляться на кілька категорій.

Серверні елементи управління HTML

Це класи, в яких містяться стандартні HTML-елементи. За винятком атрибута `runat="server"` оголошення серверних елементів управління HTML нічим не відрізняється від оголошення інших елементів управління. Двома найбільш яскравими представниками серверних елементів управління є `HtmlAnchor` (представляє дескриптор `<a>`) і `HtmlSelect` (представляє дескриптор `<select>`).

Однак в серверний елемент управління може бути перетворений будь-якій дескриптор. Якщо відповідного класу немає, ASP.NET буде просто використовувати клас `HtmlGenericControl`. Щоб перетворити звичайний елемент HTML в серверний елемент управління, потрібно всього лише додати до дескриптора цього елемента атрибут `runat="server"`.

Web-елементи управління

Ці класи дублюють функції базових HTML-елементів, але мають більш узгоджений і значущий набір властивостей і методів, які спрощують їх оголошення і доступ до них. Як приклади можна назвати елементи управління `HyperLink`, `ListBox` і `Button`. Більш того, кілька інших типів елементів управління ASP.NET (такі як багатофункціональні елементи управління і елементи управління перевіркою достовірності) часто вважаються особливими типами веб-елементів управління. В Visual Studio базові елементи управління розташовані на вкладці `Standard` (Стандартні) у вікні `Toolbox` (Панель інструментів).

Багатофункціональні елементи управління

Ці вдосконалені елементи управління можуть генерувати великий обсяг HTML-розмітки і навіть клієнтський JavaScript-код для створення інтерфейсу. Як приклади можна назвати елементи управління `Calendar`, `AdRotator` і `TreeView`. В Visual Studio чимало багатофункціональних елементів управління, які доступні на вкладці `Standard` (Стандартні) у вікні `Toolbox`.

Елементи управління перевіркою достовірності

Цей набір елементів управління дозволяє швидко перевіряти достовірність елемента управління на предмет дотримання декількох стандартних або призначених для користувача правил.

Наприклад, можна вказати, що введення не може бути порожнім, що це повинно бути число, що воно не повинно перевищувати певне значення та ін.

Якщо перевірка достовірності не проходить, необхідно запобігти обробки сторінки або дозволити цим елементам управління відображати повідомлення про помилки на сторінці. В Visual Studio ці елементи управління можна знайти на вкладці Validation (Перевірка достовірності) у вікні Toolbox.

Елементи управління даними

Ці елементи управління включають складні сітки і списки, що призначені для відображення великих обсягів даних, з підтримкою додаткових властивостей на зразок створення шаблонів, редагування, сортування і розбиття на сторінки.

Цей набір також містить елементи управління джерелом даних, що дозволяють прив'язуватися до різних джерел даних декларативно, без написання додаткового коду.

Елементи управління навігацією

Ці елементи управління призначені для відображення карт сайту і дозволяють переміщатися з однієї сторінки на іншу.

Елементи управління входом в систему

Ці елементи управління підтримують аутентифікацію за допомогою форм, реалізують модель ASP.NET для аутентифікації користувачів по базі даних і відстеження їх стану.

Замість написання власних інтерфейсів, для роботи з аутентифікацією за допомогою форм, можна застосовувати ці елементи управління для роботи з сторінками входу в систему, відновлення паролів і майстрів створення нових користувачів.

Елементи управління Web Parts

Цей набір елементів управління підтримує Web Parts – модель ASP NET для побудови компонентних веб-порталів, що легко конфігуруються.

Елементи управління ASP.NET AJAX

Ці елементи управління дозволяють використовувати в веб-сторінках прийоми Ajax без написання клієнтського коду. Сторінки в стилі Ajax можуть бути більш швидкодіючими, оскільки виключають проходження стандартного циклу зворотного відправлення і оновлення сторінки.

Елементи управління ASP.NET Dynamic Data

Ці елементи управління підтримують компонент ASP.NET Dynamic Data, який дозволяє створювати гнучкі веб-сайти за рахунок побудови гнучких шаблонів, а не написання коду.

Всі серверні елементи управління успадковані від базового класу управління з простору імен System.Web.UI. Це вірно при використанні серверних елементів управління HTML, застосуванні Web-елементів управління або створенні власних спеціальних елементів управління. Це також відноситься до класу Page, від якого відбуваються всі форми.

На рисунку 5.1 нижче показані основні гілки цього ланцюжка успадкування.

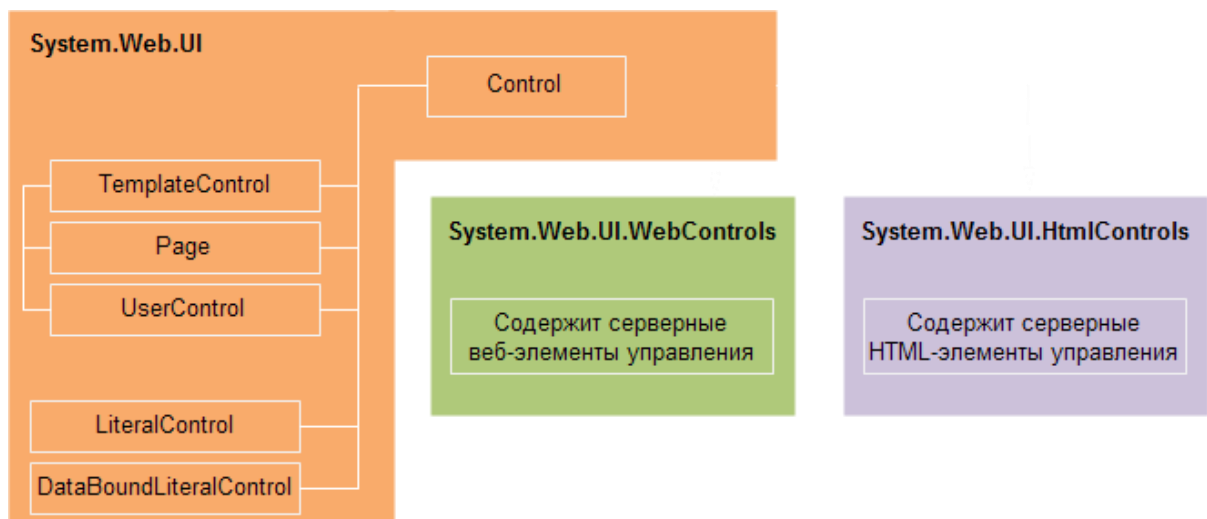


Рис. 5.1 – Ієрархія серверних елементів управління

Елементи управління HTML є спадкоємцями класу System.Web.UI.HtmlControls. Вони безпосередньо відображаються у вигляді елементів розмітки HTML. Їх відображення не залежить від типу браузера. Властивості таких елементів повністю відповідають атрибутам тегів HTML.

Порівняємо звичайний тег:

```
<input id="Reset1" type="reset" value="reset" />
```

з елементом управління HTML:

```
<input id="Reset1" runat="server" type="reset" value="reset" />
```

Різниця полягає тільки в наявності атрибуту `runat="server"`. Але він дає колосальну різницю. Теги сервер відображає як `є`, а елементом управління можна маніпулювати в кодї. Тільки у другому випадку в функції–методї сторїнки можна написати:

```
Reset1.Value = "АСП";
```

або

```
this.Reset1.Value = "АСП";
```

Отже, `Reset1` стає одним з членів класу сторїнки.

Цї класи використовують, якщо необхідно отримати визначенї теги HTML або якщо потрібно конвертувати старї сторїнки `asp`. Елементи управління HTML можна розміщувати на одній сторїнцї разом з Web–серверними елементами.

Web–сервернї елементи могутнїшї, тому що вони прив'язанї не до розмітки, а до функціональності, яку потрібно забезпечити. Багато елементів не мають аналогів в HTML, наприклад, календар. Відображення таких елементів повністю контролюється ASP .NET. Перехоплюючи подїї `PreRender`, `Init`, `Load`, можна втручатися в цей процес. Оголошення серверного елемента управління починаються з блоку:

```
<asp:mun> і закінчуються </asp:mun> Наприклад:
```

```
<asp:Label ID="Label1" runat="server" Text="Hello World"></asp:Label>
```

Можливо також закрити оголошення тегом `</>`, якщо всерединї блоку немає тексту:

```
<asp:Label ID="Label1" Runat="server" Text="Hello World" />
```

Властивостї цих елементів строго типїзуються, на відмїну від HTML–елементів.

У таблицї 5.1 приведенї WEB–елементи управління, якї мають пару серед тегів HTML. Взагалї їх значно бїльше. Деякї елементи генерують не тїльки HTML–код, але і JavaScript. В ASP .NET було додано безлїч нових складних елементів управління, наприклад, `MultiView`, `TreeView`, `Wizard`, `GridView`.

Сервер не обов'язково генерує ті ж самі теги HTML для серверних елементів управління. Все залежить від типу браузера, який використовує клієнт. Всі серверні елементи управління знаходяться в просторі імен System.Web.UI.Control і успадковуються від класу System.Web.UI.WebControls.

Таблиця 5.1

Таблиця відповідності деяких серверних елементів управління тегам HTML

Елемент управління ASP.NET	Відповідний тег HTML	Призначення
<asp:Label>		Відобразити текст
<asp:ListBox>	<Select>	Список вибору
<asp:DropDownList>	<Select>	Список
<asp:TextBox>	<Input Type="Text"> <Input Type="Password"> <Textarea>	Рядок редагування Поле редагування
<asp:HiddenField>	<Input Type="Hidden">	Невидиме поле
<asp:RadioButton>		
<asp:RadioButtonList>	<Input Type="Radio">	Перемикач, список перемикачів
<asp:CheckBox>		
<asp:CheckBoxList>	<Input Type="CheckBox">	Прапорець, список прапорців
<asp:Button>	<Input Type="button"> <Input Type="submit">	Командна кнопка
<asp:Image>		Зображення
<asp:ImageButton>	<input type="image">	Кнопка-зображення
<asp:Table>	<Table>	Таблиця
<asp:Panel>	<Div>	Контейнер
<asp:BulletedList>	,	Маркований список
<asp:HyperLink>	<A Href>	Гіперпосилання

Всі існуючі класи можна переглянути за допомогою Class Browser. У всіх інтегрованих середовищах розробки є можливість додавати елементу

управління за допомогою простого перетягання мишею.

Серверні елементи управління надають в розпорядження програміста властивості, методи і події. При їх допомозі можна абстрагуватися від деталей HTML-коду і працювати із сторінкою і її елементами як з об'єктами.

Завдання для виконання

ЗАВДАННЯ 5.1

Створіть Web-форму, яка виводить персональне вітання для відвідувача сайту. Від відвідувача сайту потрібно ввести своє ім'я (рис. 5.2). Використовуйте елементи управління TextBox, Label, Button.

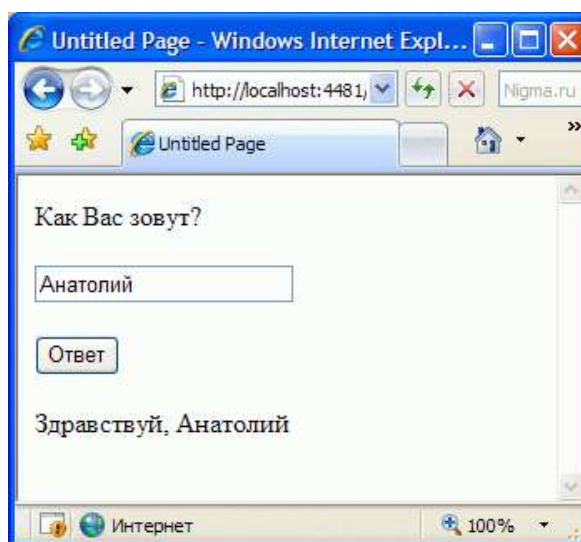


Рис.5.2 – Приклад виконання завдання 5.1

Запишіть відомості про елементи управління, які використали в наведену таблицю:

№	Тип елемента	Назва	Значення
1	TextBox		
2	Label		
3	Button		

Приведіть фрагмент коду Web-сторінки, за допомогою якого задаються елементи управління.

Приведіть текст функції, за допомогою якої виводиться вітання.

ЗАВДАННЯ 5.2

Створіть Web-форму з використанням елементів управління DropDownList і Label, яка пропонує вибрати із списку, що розкривається, стать (чоловік, жінка, воно) відвідувача і виводить повідомлення (рис. 5.3).

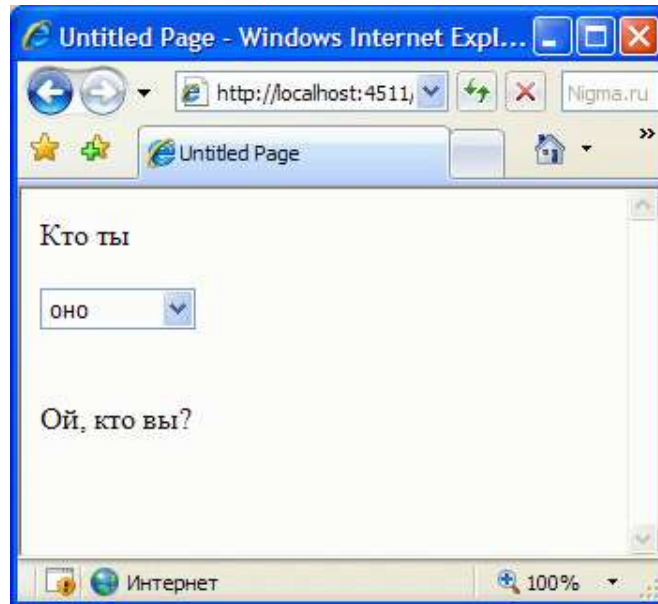


Рис.5.3 Приклад виконання завдання 5.2

Для написання функції обробки вибраного значення в елементі DropDownList пропонується використовувати оператори If або Switch. Запишіть код функції обробки вибраного значення, яка реалізована в елементі DropDownList.

ЗАВДАННЯ 5.3

Створіть Web-форму з використанням елементів управління TextBox, Button і Label, яка пропонує користувачу ввести своє ім'я і по закінченню імені намагається вгадати його стать. При виконанні роботи врахувати, що повинні вводиться повні імена (рис. 5.4).

Чоловічі імена як правило мають закінчення «й», «р», «н». Жіночі імена як правило мають закінчення «а», «я». Для отримання останньої букви імені використовуйте функції роботи з рядками. Запишіть код функції обробки вибраного значення, що реалізована, в полі TextBox.

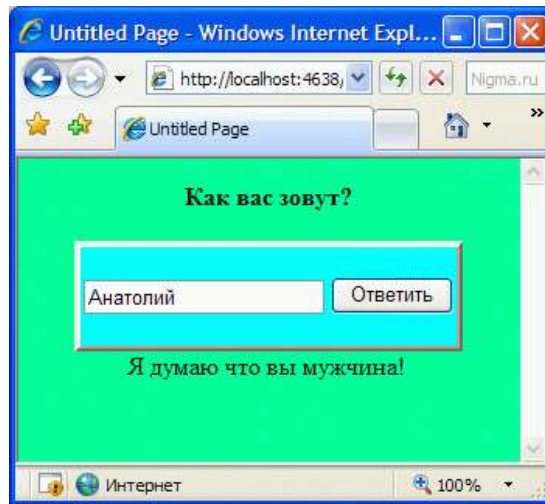


Рис.5.4 – Приклад виконання завдання 5.3

ЗАВДАННЯ 5.4

Створіть Web-форму з використанням елементів управління Textbox, Button і Label, яка пропонує користувачу ввести своє ім'я і вік, а потім підраховує кількість прожитих днів. При виконанні роботи вважати, що:

- у році 365 днів;
- високосні роки не враховувати;
- враховувати тільки число повних прожитих років.

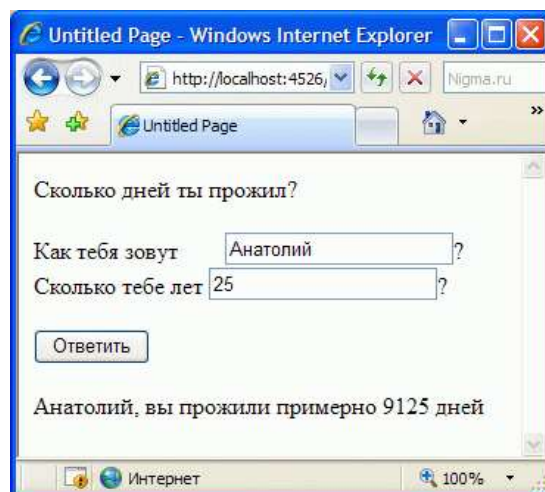


Рис.5.5 – Приклад виконання завдання 5.4

Для перекладу значення строкової змінної (тип string), що визначає число років в числове уявлення (тип int) рекомендується використовувати

функцію Convert.ToInt16. Запишіть код функції обробки кількості прожитих років, яка реалізована в полі Textbox.

ЗАВДАННЯ 5.5

Створіть Web-форму з використанням елементу управління CheckBoxList і Label, яка виводить слова "Привіт" або "Hello" або обидва слова разом залежно від того, напроти якої мови встановлений прапорець (рис. 5.6):

- встановлене Russian – вивести "Привет";
- встановлене English – вивести "Hello";
- встановлене Russian і English – вивести "Hello Привет".

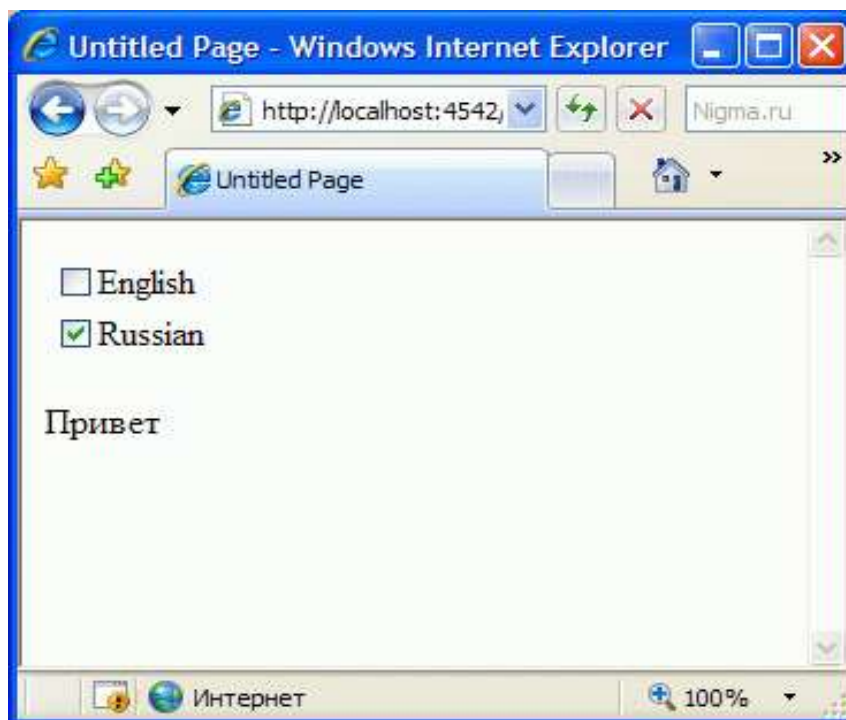


Рис.5.6 – Приклад виконання завдання 5.5

Для перевірки установки прапорця скористайтеся властивістю CheckBoxList1.Items[j].Selected, що набуває значень true при встановленому прапорці, і false – інакше. Запишіть код функції обробки встановлених прапорців елементу управління CheckBoxList.

ЗАВДАННЯ 5.6

Створіть Web-форму з використанням елементів управління

DropDownList і Image, яка виводить зображення відповідне вибраному знаку зодіаку (рис. 5.7).

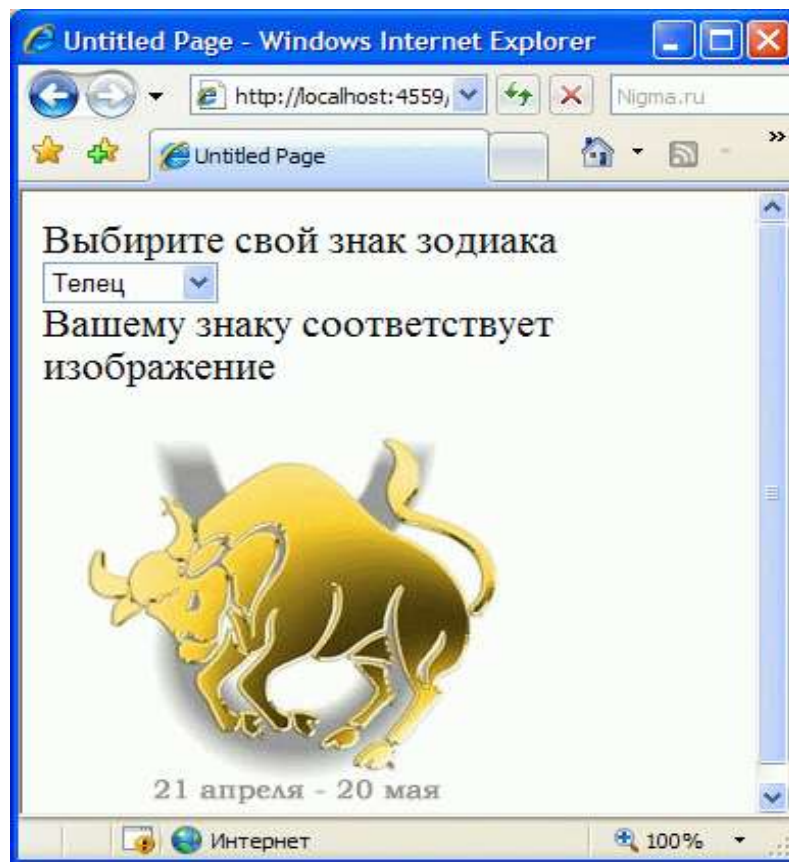


Рис.5.7 – Приклад виконання завдання 5.6

Для перевірки вибраного значення в елементі DropDownList рекомендується використовувати властивість DropDownList1.SelectedItem, а для здійснення подальших дій рекомендується використовувати оператор switch. Запишіть код функції обробки вибраного значення і виведення зображення, що реалізована в елементі DropDownList.

Контрольні питання:

1. Описати типи серверних елементів управління.
2. Пояснити відмінність Web-елементів управління від HTML-елементів управління.
3. Як оголошується веб-елемент управління в .aspx файлі?
4. Визначити поняття властивості та події елементів управління,

привести приклади.

5. Призначення, властивості і події веб-елемента управління Label.
6. Призначення, властивості, події веб-елемента управління TextBox.
7. Призначення, властивості, події веб-елемента управління Button.
8. Призначення, властивості, події веб-елемента управління CheckBoxList.
9. Призначення, властивості, події веб-елемента управління DropDownList.
10. Призначення, властивості, події веб-елемента управління Image.

ЛАБОРАТОРНА РОБОТА №6

Створення дизайну сторінок та системи навігації

веб-додатка в Microsoft Visual Studio 2015

Мета роботи: вивчити особливості побудови дизайну сторінок веб-додатка та отримати практичні навички створення:

- майстер-сторінки та контент-сторінки;
- таблиць стилів;
- зв'язку таблиць стилів з веб-сторінками;
- тем, додавання в них стилів, застосування тем до веб-сторінок;
- системи навігації сторінками веб-додатка.

Постановка завдання: у середовищі Microsoft Visual Studio 2015 виконати завдання 6.1 – 6.5 за інструкцією до виконання, яка наведена в теоретичній частині лабораторної роботи. Проілюструйте виконання зазначених завдань екранними формами, які будуть включені до звіту.

Теоретичні відомості

Створення і використання майстра сторінок

Майстер-сторінки служить шаблоном відображення інших сторінок. Для цього на майстер-сторінки виділяються області, які не підлягають зміні, і області, де буде відображатися інформація пов'язаних сторінок (сторінок контенту).

Контент-сторінка – це будь-яка сторінка, яка використовує майстер-сторінку. Кожен раз, коли відвідувач запитує контент-сторінку, ASP.NET завантажує майстер-сторінку, виробляє злиття з контент-сторінкою і посилає об'єднаний результат користувачеві. Злиття майстер-сторінки і контент-сторінки без перезавантаження (так званий засіб «на льоту») має два важливих наслідки:

- відвідувач завжди отримує поточні версії шаблону і його змісту;
- отримана після злиття сторінка має всі можливості звичайної ASP.NET сторінки.

Наприклад, майстер-сторінка, як і контент-сторінка може містити будь-які елементи управління, або фрагменти коду. Оптимальний варіант – розробка майстер-сторінки на етапі планування сайту.

Щоб створити сайт, який використовує майстер–сторінки потрібно виконати наступні дії:

- створіть новий сайт;
- видалити будь–які веб–сторінки, які містять сайт, наприклад, видаліть сторінку default.aspx, яка створюється за замовчуванням;
- у вікні Solution Explorer клацніть правою кнопкою миші по папці сайту і виберіть Add New Item з контекстного меню.

Коли діалогове вікно Add New Item з'явиться, виконайте наступні дії, щоб отримати майстер–сторінку (рис. 6.1):

- виберіть зі всіх елементів списку Master Page;
- залиште назву сторінки за замовчуванням MasterPage.master;
- виберіть мову програмування (за замовчуванням C #);
- поставте галочку навпроти Place code in separate file;
- натисніть Add, щоб створити майстер–сторінку.

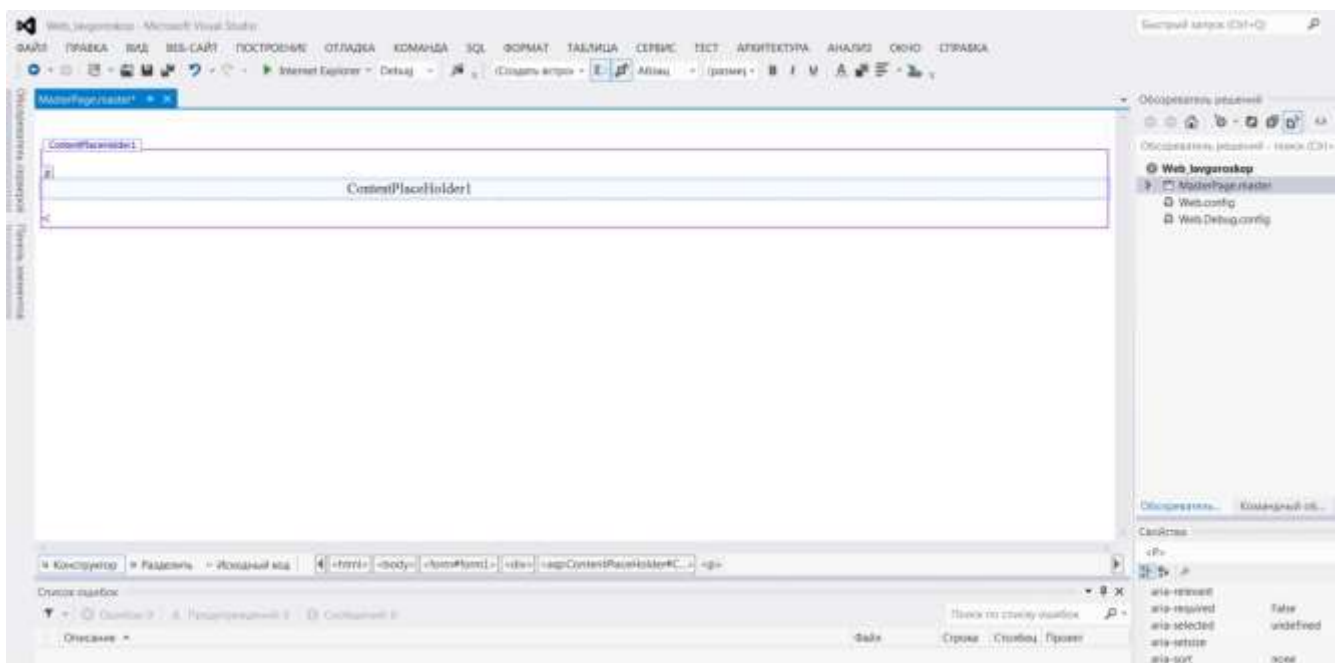


Рис. 6.1 – Нова майстер–сторінка, яка містить один елемент управління ContentPlaceHolder

Створена майстер–сторінка, доступна для редагування за винятком одного елемента управління ContentPlaceHolder, який створюється на ній за замовчуванням. Цей елемент резервує простір для вмісту контент–сторінки

(рис. 6.1). Редагування майстер–сторінки таке ж, як і редагування звичайної ASP–сторінки. Тому до неї можна застосувати всі елементи управління, які були розглянуті раніше. На рис.6.2 представлена відредагована майстер–сторінка.

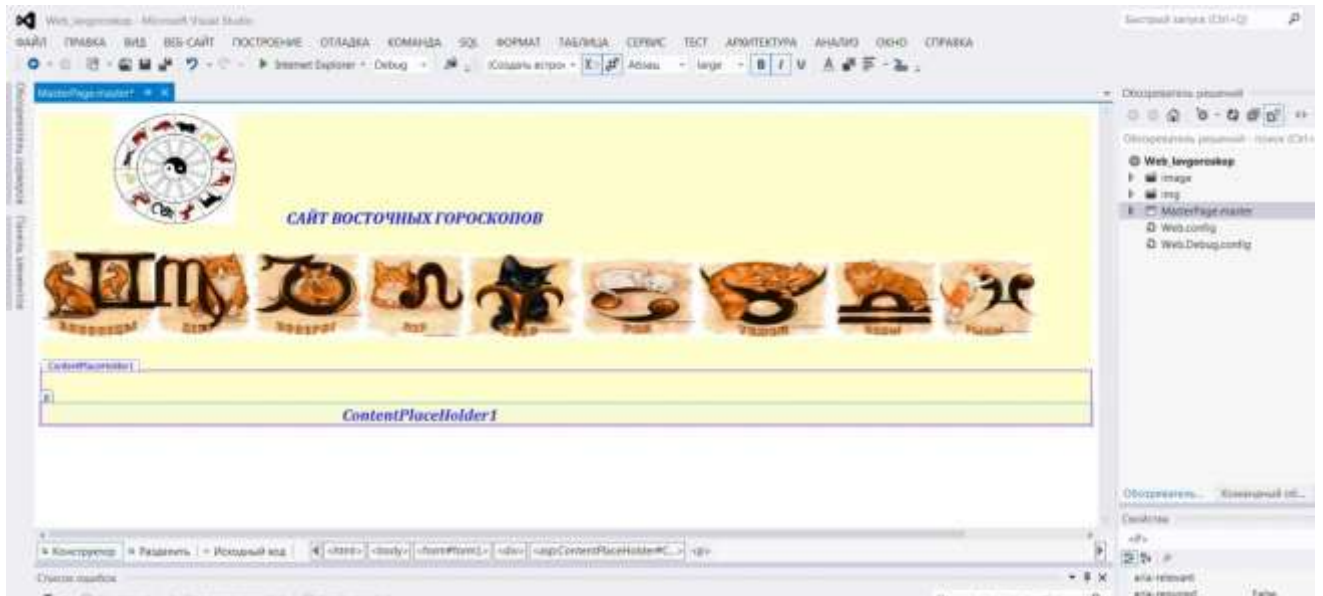


Рис. 6.2 – Створена та відредагована майстер–сторінка

Найзручніший спосіб використання майстер–сторінки полягає в тому, щоб прив'язати її до нової сторінки під час її створення, що може бути виконано наступним чином:

- відкрийте сайт, який містить майстер–сторінку;
- у вікні Solution Explorer клацніть правою кнопкою миші по папці, де буде розміщена нова контент–сторінка;
- виберіть Add New Item з контекстного меню.

Коли діалогове вікно Add New Item з'явиться, виконайте наступні дії (рис.6.3):

- виберіть шаблон документа: Web Form;
- поставте галочку навпроти Select Master Page;
- клацніть на кнопку Add для того, що з'явилося діалогове вікно Master Page;
- виберіть майстер–сторінку.

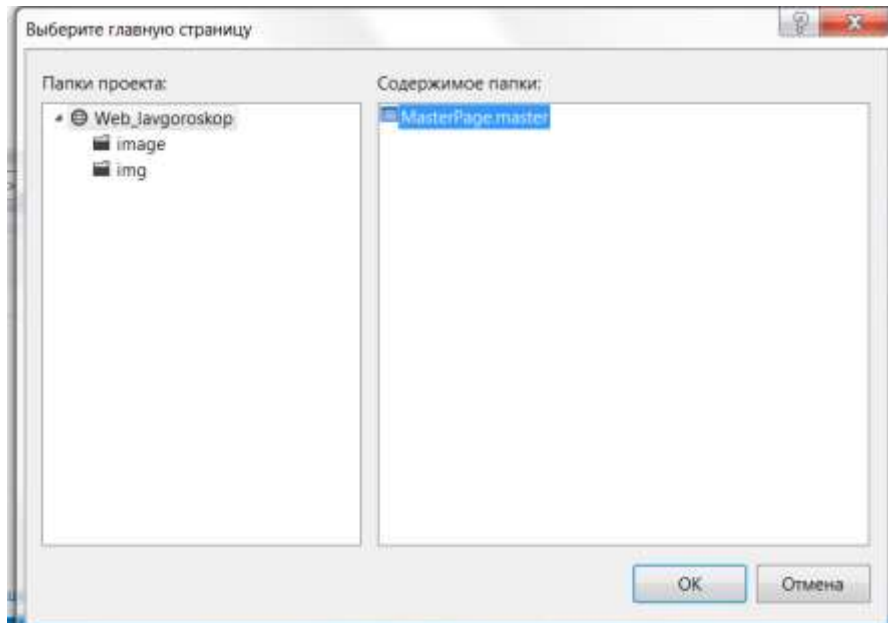


Рис. 6.3 – Вибір майстер-сторінки

В режимі конструктора частини контент-сторінки, які додалися з майстер-сторінки будуть недоступні для редагування. Щоб змінити щось в них необхідно відкрити і відредагувати майстер-сторінку (рис.6.4).

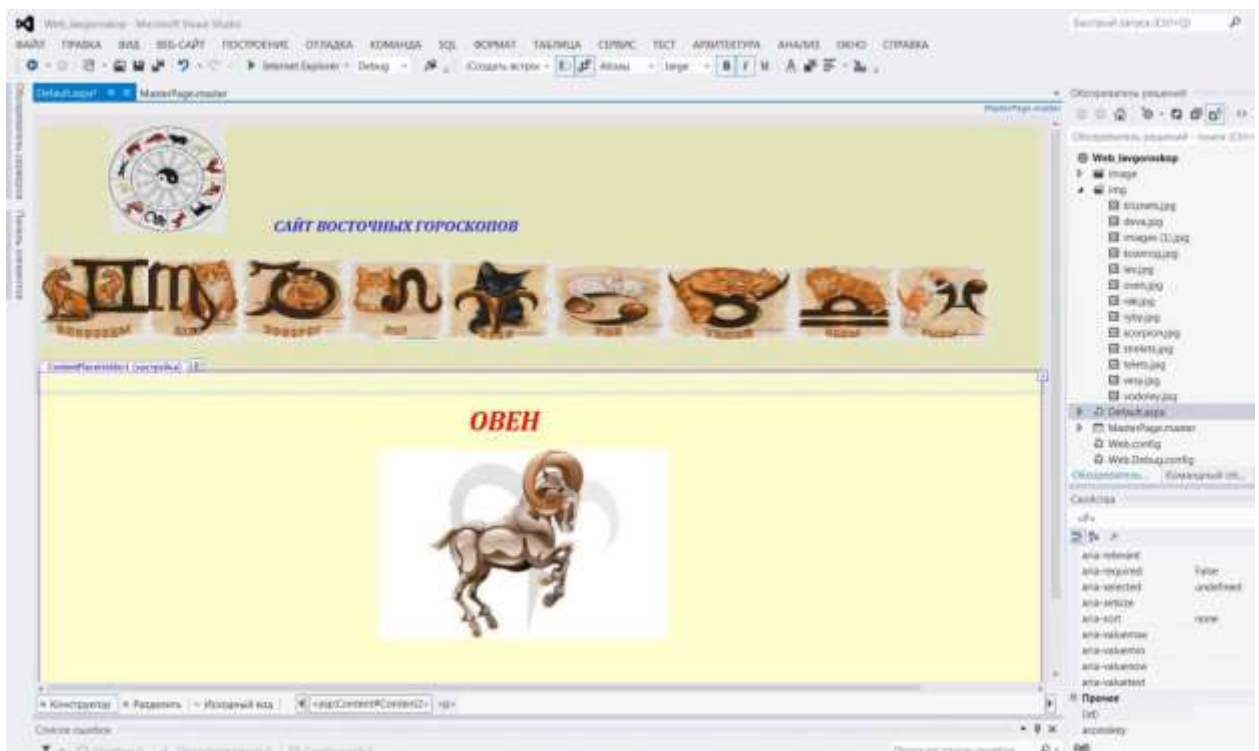


Рис. 6.4. – Контент-сторінка з шаблоном

На рис.6.4 представлена контент–сторінка, яка використовує приєднану майстер–сторінку. Контент–сторінка відображає деякий власний вміст. Щоб створити таку сторінку, виконайте наступні дії:

- створіть ASP–сторінку, в прикладі вона називається Бук.aspx;
- на етапі створення підключіть майстер–сторінку (MasterPage.master);
- задайте вміст сторінки всередині елемента ContentPlaceHolder, текст можна надрукувати або вставити його як зображення;
- збережіть сторінку.

Клацніть правою кнопкою миші по сторінці в Solution Explorer і виберіть View In Browser з контекстного меню для перегляду сторінки в браузері.

Контент–сторінки, які використовують майстер–сторінки спочатку містять дуже невеликий вихідний текст, наприклад такий:

```
<%@ Page Language="C#" MasterPageFile="~/MasterPage.master"
AutoEventWireup="true"
CodeFile="овен.aspx.cs" Inherits="овен" Title="Untitled Page"
%>
<asp:Content ID="Content1"
ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
</asp:Content>
```

Директива @Page визначає мову програмування для будь–якого коду, що додано до сторінці, назву файлу майстер–сторінки і заголовки сторінки.

Тег <asp: Content> визначає ім'я елемента управління ContentPlaceHolder на головній сторінці, і атрибут Runat = "Server". Дані теги дозволяють зрозуміти принцип роботи майстер–сторінки. Майстер–сторінка містить один або більше елементів управління ContentPlaceHolder, кожен з яких має власне індивідуальне ім'я.

Коли контент–сторінка використовує майстер–сторінку, вона повинна містити один або декілька тегів <asp: Content> для кожного елемента управління ContentPlaceHolder в майстер–сторінці. Коли відвідувач запитує контент–сторінку, ASP.NET зливає її зміст з кожним елементом управління ContentPlaceHolder в майстер–сторінці, потім показує результати.

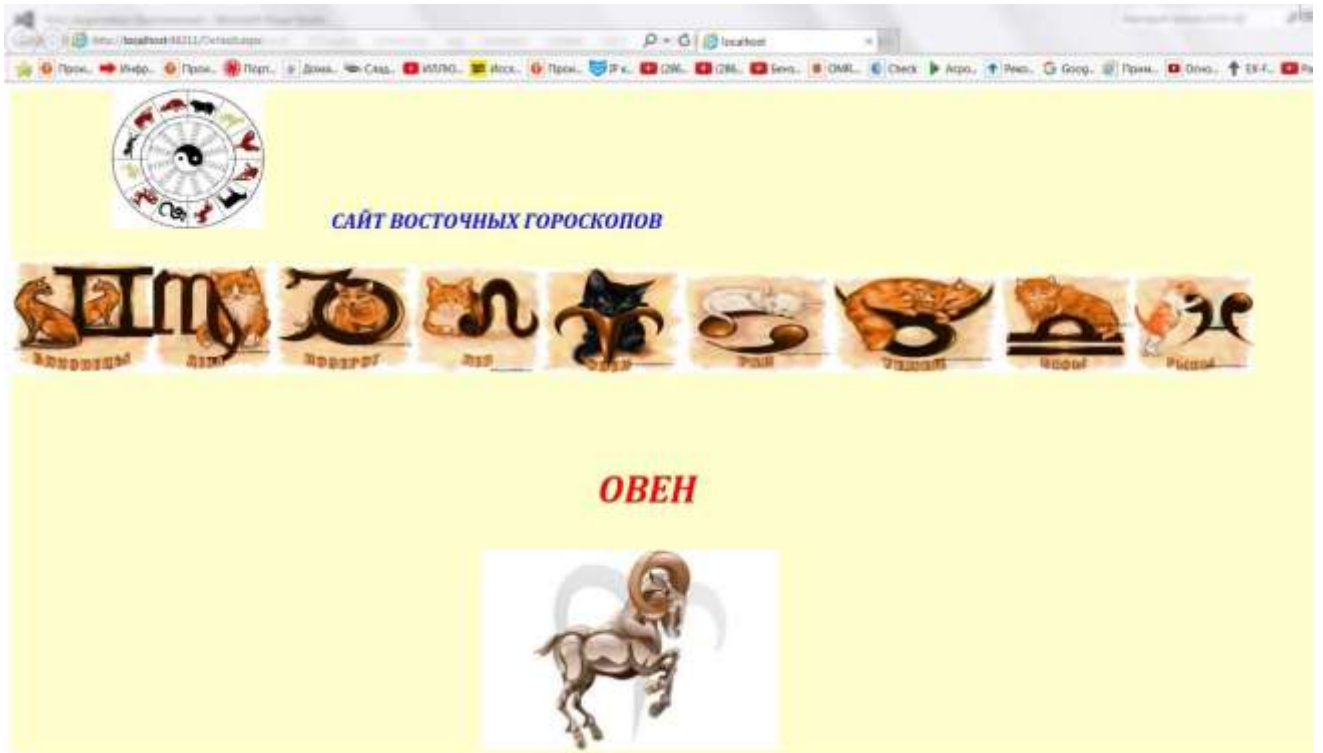


Рис.6.5 – Відображення контент–сторінки та майстер–сторінки в браузері

Слід зазначити, що контент–сторінка, яка використовує майстер–сторінку, не містить ніяких тегів типу `<html>` `</html>`, розділу `<head>`, тега `<body>` `</body>` і тега `<form>` `</form>`. Всі вони запозичуються з майстер–сторінки. Майстер–сторінки представляють собою шаблони веб–сторінок. Відвідувачі не переглядають їх безпосередньо. Замість цього вони переглядають контент–сторінки, що прикріплені до них. Це робить сайт більш однорідним і простішим в процесі роботи з ним.

Використання CSS

Щоб визначити стиль для певного елемента сторінки, спочатку виділіть його в режимі конструктора і потім зробіть наступне:

- виберіть пункт Новий стиль (New Style) з меню Format або
- виберіть ознаку Style у вікні Properties;
- клацніть на кнопку, яка з'явилася в тому ж рядку.

Після цього Visual Studio відобразить діалогове вікно Style Builder, яке

показано на рис. 6.6.

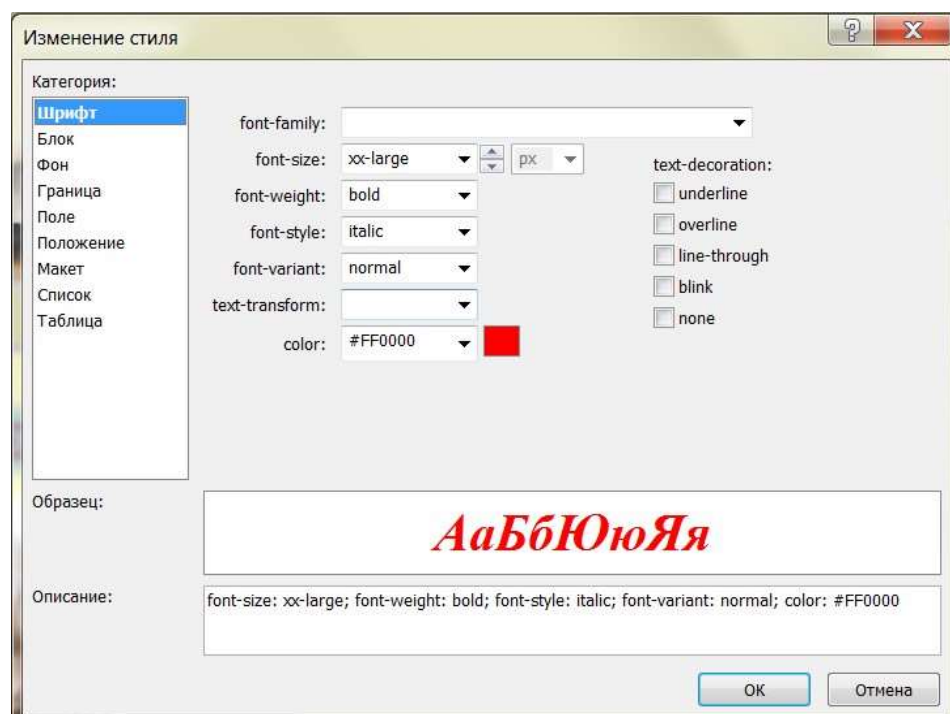


Рис. 6.6 – Вікно діалогу, яке застосовує CSS–властивості до елементів сторінки або до правил таблиці стилів

Властивості CSS згруповані у вісім категорій, які розташовані по лівому краю діалогового вікна. Права частина діалогового вікна змінюється в залежності від обраної категорії. Більшість параметрів CSS можна визначити інтуїтивно. Повний перелік параметрів наведено в вбудованій системі допомоги або в офіційній специфікації CSS на сайті <http://www.w3.org/Style/CSS>.

Застосування властивостей CSS до індивідуальних елементів сторінки забезпечує більшу гнучкість в оформленні, але не більшу однорідність. Чим більша кількість сторінок містить сайт, тим важче завдання збереження шрифтів і кольорів. Але є можливість створити загальнодоступний файл таблиці стилів, який вирішить цю проблему. Процедура створення нового файлу таблиці стилів полягає в наступному.

Виберіть New File з меню File або клацніть правою кнопкою миші по папці в Solution Explorer, і виберіть пункт Add New Item. Коли діалогове вікно Add New Item з'явиться, виберіть шаблон Style Sheet, за потреби змініть назву файлу, яке дається за замовчуванням, і потім клацніть кнопкою Add.

У файлі таблиці стилів містяться правила, кожне правило складається з назви, списку властивостей CSS і значень. Передбачено два способи додати нове правило стилю до файлу таблиці стилів:

- надрукувати назву правила, і пару вигнутих фігурних дужок ({});
- вибрати назву правила в створеному файлі CCS, натиснути праву кнопку миші і вибрати пункт Побудувати стиль, що призведе до появи діалогового вікна Зміна стилю, яке показано на рис.6.7.

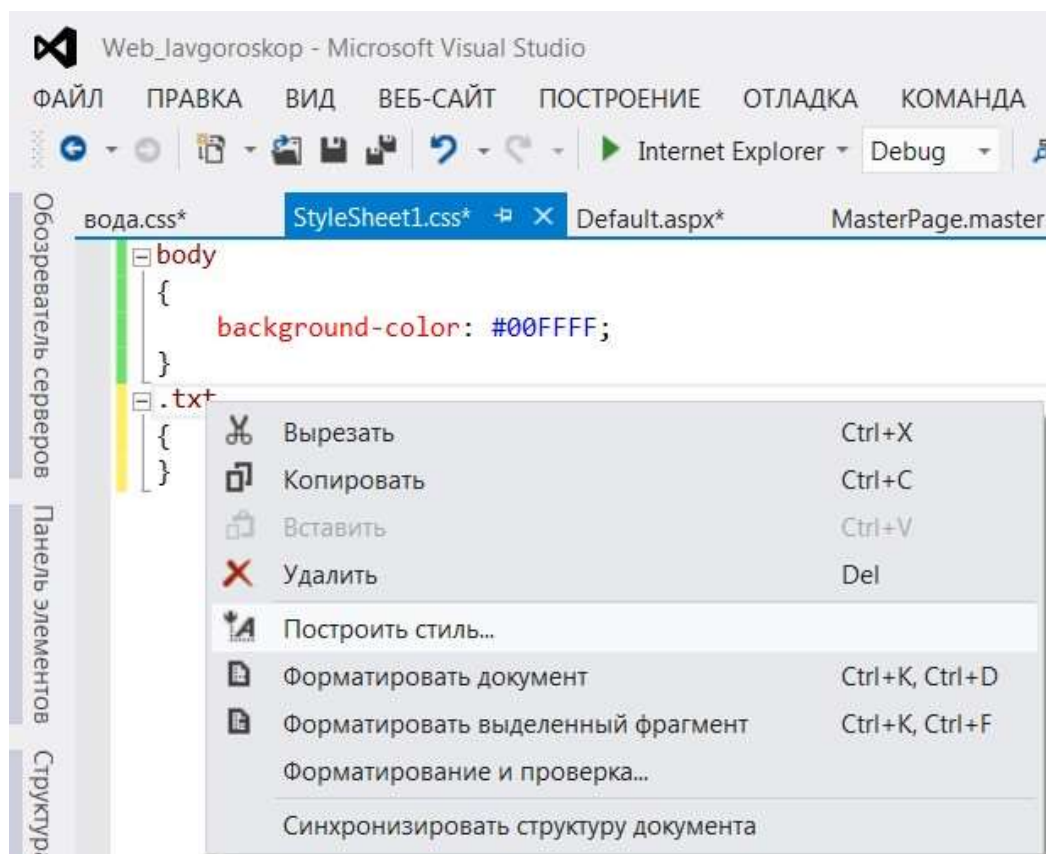


Рис. 6.7 – Вікно діалогу створення нових правил стилю

Після створення правила стилю можна додати до нього властивості наступними способами:

- ввести їх самостійно або за допомогою IntelliSense;
- натиснути праву кнопку миші по стилю і вибрати Зміна стилю з контекстного меню.

На рис.6.8 представлено відображення файлу таблиці стилів. Зліва відображається вікно Структура CSS, яке полегшує знаходження і вибір стилів, що оголошені у файлі.

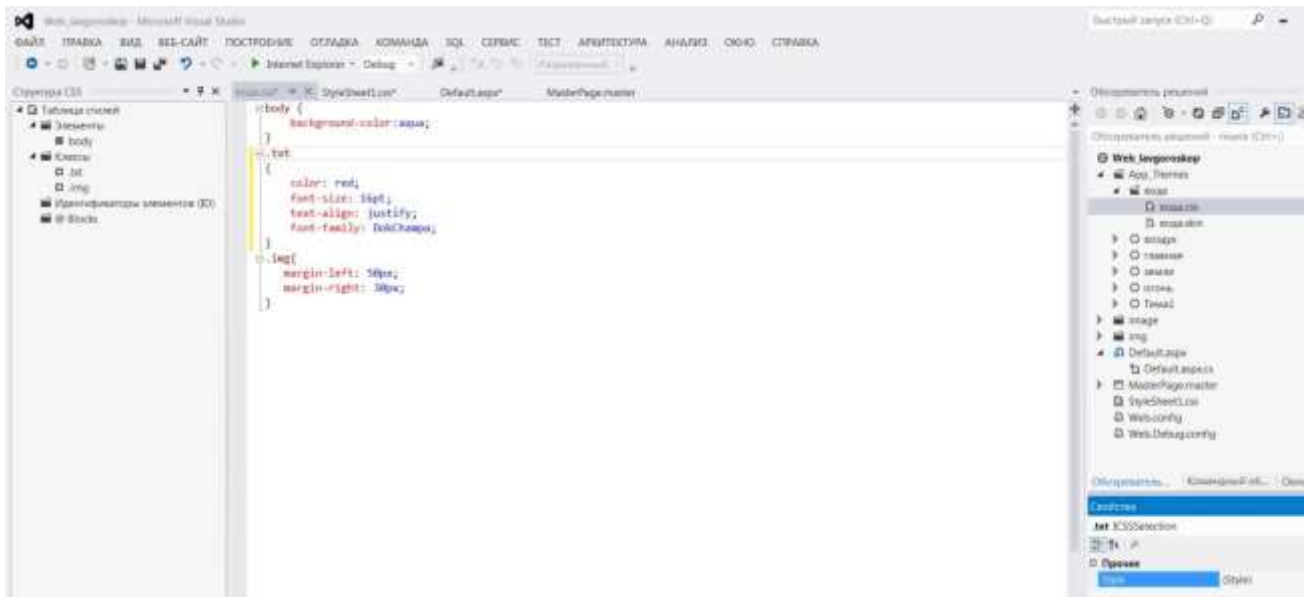


Рис 6.8 – Таблица стилів

Щоб використовувати створений файл стилів потрібно зв'язати його з веб–сторінкою. Для цього необхідно відкрити веб–сторінку, яка буде використовувати файл таблиці стилів і потім виконайте одну з дій:

- відкрити сторінку в режимі Design, потім захопіть файл таблиці стилів з вікна Solution Explorer і помістіть його сторінку, що редагується;
- виберіть пункт DOCUMENT у вікні Properties, виберіть властивість StyleSheet, клацніть на кнопку, що з'явилася в рядку.

Після появи вікна діалогу виберіть файл таблиці стилів та натисніть ОК. Як тільки файл таблиці стилів буде пов'язаний зі сторінкою, правила будуть застосовані на сторінку і вступлять в силу негайно. Використання таблиці стилів робить оформлення сайту однорідним та більш прозорим для подальшого редагування.

Використання тем

Як і таблиця стилів, тема не має графічного представлення, тому робота з нею ведеться в редакторі тексту. Щоб створити нову тему необхідно виконати наступні дії:

- відкрити веб–сайт, для якого буде створена тема;

- клацнути правою кнопкою миші по папці сайту та вибрати Add Folder з контекстного меню, а потім вибрати папку Теми з підменю; якщо сайт ще не містить папку App_Themes, то VS створить її. VS завжди створює підпапку з назвою Theme1, яку можна перейменувати за потреби;
- перейменувати підпапку Theme1. Назва, яку отримає папка, стане назвою теми.

Якщо папка теми залишиться порожньою, то змін в дизайні не буде. Щоб задіяти тему, спочатку необхідно додати .skin або .css файли. (рис.6.9).

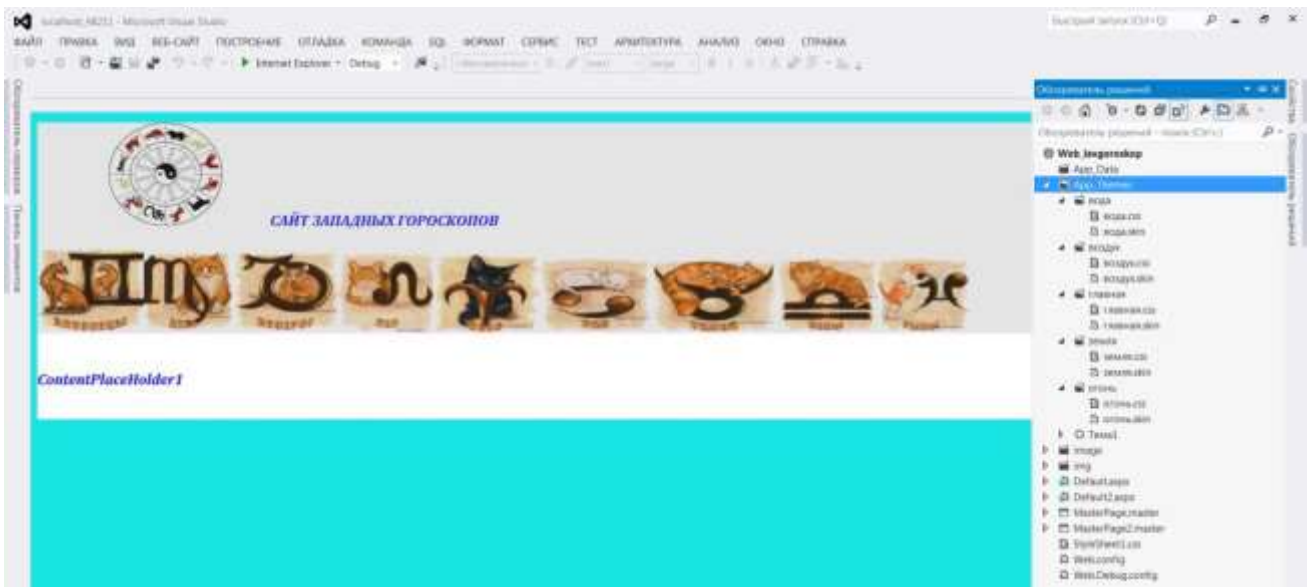


Рис. 6.9 – Створення чотирьох стилів для сайту

У ASP.NET skin-файл визначає візуальні властивості для одного або більше видів елементів управління, що розташовані на сторінках сайту. Підпапка теми може містити як один, так і кілька skin-файлів. При розміщенні в ній кілька файлів ASP.NET буде читати їх всі, як ніби це один файл.

Для створення skin-файлів потрібно виконати наступні дії:

- клацнути правою кнопкою миші по папці потрібної теми та виберіть Add New Item з контекстного меню;
- після появи вікна діалогу Add New Item виберіть шаблон Файл обкладинки (Skin File), за потреби змінити ім'я файлу за замовчуванням та клацнути кнопкою Add.

VS відобразить новий skin-файл в редакторі коду. Для кожного елемента управління, до якого буде застосована тема додайте наступні рядки:

```
<asp:RadioButtonList runat = "server"/>  
<asp:DropDownList runat = "server"/>  
<asp:TextBox runat = "server"/>
```

Ці теги нагадують ті, що з'являються при додаванні відповідного елемента управління до веб-сторінки. Найпростіший спосіб створити skin-файл скопіювати потрібні теги з веб-сторінки і відредагувати їх. У межах кожного тегу додати параметри елементів управління і задати їх значення.

У разі необхідності можна відкрити веб-сторінку в режимі дизайнера, змінити відповідні параметри, а потім перейти в режим редагування коду і скопіювати отриманий код з веб-сторінки в skin-файл. Після цього необхідно зберегти skin-файл шляхом вибору пункту Save з меню File. Слід враховувати, що skin-файл може тільки змінити відображення елементів веб-сторінки, які мають візуальні властивості, і не може керувати ними.

Після створення папок з темами файли таблиці стилів можна зберігати не окремо, а в підпапках відповідних тем. Тоді теми автоматично будуть пов'язувати збережені в своїх підпапках таблиці стилів з будь-якими сторінками, які використовують дану тему.

У прикладі є тема "земля" і тема "вода", які розташовуються у відповідних папках: папка App_Themes / вода, містить файл вода.css, папка App_Themes / земля, містить файл земля.css Результат застосування теми "земля" показаний на рис. 6.10, а теми "вода" – на рис. 6.11.



Рис. 6.10 – Сторінка, що використовує тему «земля»



Рис. 6.11 – Сторінка, що використовує тему «вода»

Теми можливо застосовувати до індивідуальних веб–сторінок або до всього сайту. Щоб застосувати тему до одної веб–сторінки, необхідно виконати наступні дії:

- відкрити сторінку в VS;
- вибрати DOCUMENT у вікні Properties;
- знайти властивість Theme і вибрати потрібну тему зі списку.

Щоб визначити тему, яка звертається до всього веб–сайту, необхідно змінити файл конфігурації сайту (web.config):

- відкрити файл web.config в VS;
- знайти пару тегів <system.web> і </system.web>; при відсутності цих тегів їх необхідно створити;
- додати тег, що задає тему:


```
<system.web>
  <pages theme = "земля"/>
</system.web>
```

На рис. 6.10 і рис. 6.12 представлена одна сторінка сайту, вона має різний вигляд тому, що в другому випадку в skin–файл теми "земля" був доданий рядок наступного виду:

```
<asp:Image runat="server" ImageUrl="~/img/мелец01.jpg"
  BorderColor="Red" BorderStyle="Dotted" BorderWidth="2px" />.
```

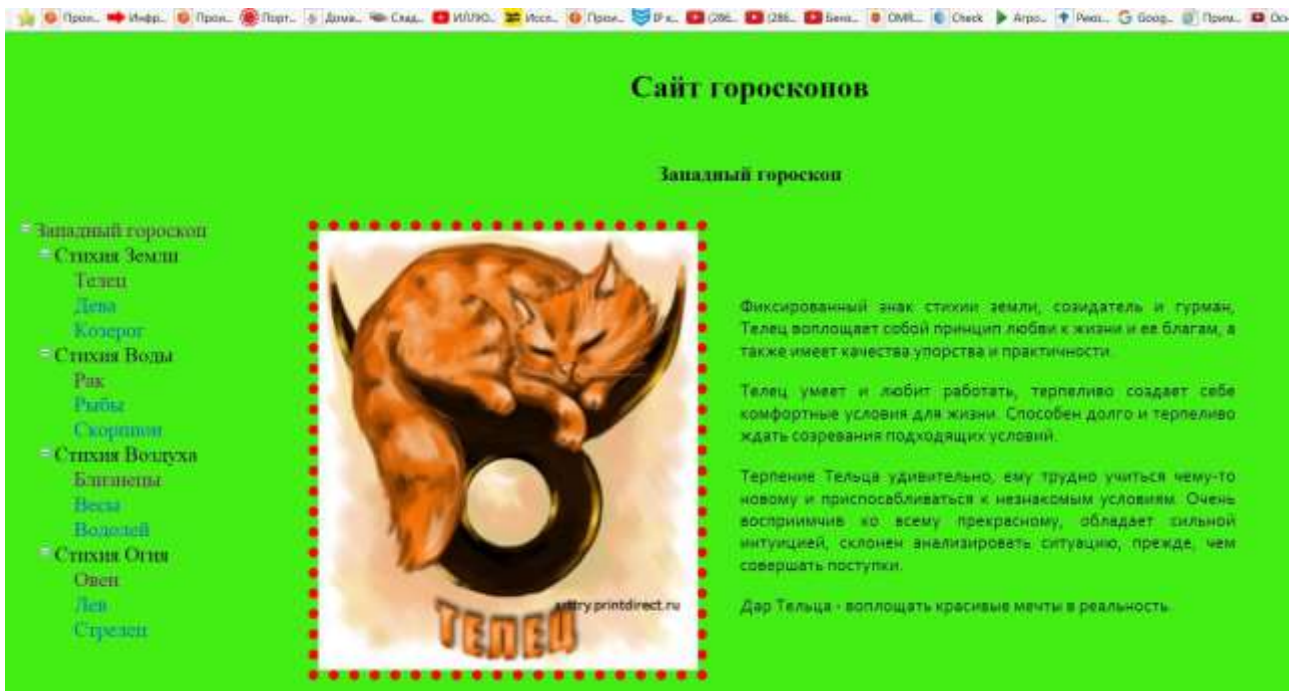


Рис. 6.12 – Сторінка, що використовує тему «земля», на якій змінено рисунок

Приклад файлу земля.skin:

```
<asp:Label runat="server" CssClass="txt" />
<asp:Image runat="server" ImageUrl="~/image/телец.jpg"
SkinID="телец" CssClass="img"/>
<asp:Image runat="server" ImageUrl="~/image/дева.jpg" SkinID="дева"
CssClass="img"/>
<asp:Image runat="server" ImageUrl="~/image/козерог.jpg"
SkinID="козерог" CssClass="img"/>
```

Створення карти сайту

Кожен веб-сайт потребує системи навігації, яка дозволяє відвідувачам отримувати швидкий доступ до інформації, яку вони шукають. При розробці великого сайту важливо зберегти ясність і зрозумілість системи навігації, оскільки з часом сайт буде змінюватися і оновлюватися. З цим завданням допоможуть впоратися такі чотири компоненти:

- файл Web.sitemap, який задає логічну структуру сайту;
- елемент управління Menu, який читає файл Web.sitemap і відображає структуру сайту у вигляді меню, що розкривається;

- елемент управління TreeView, який читає файл Web.sitemap і відображає структуру у вигляді деревовидного списку;
- елемент управління SiteMapDataSource, який пов'язує файл карти сайту (Web.sitemap) і елементи управління для її відображення (TreeView або Menu).

Спільне застосування цих компонент дозволяє швидко і просто організувати систему навігації на сайті і підтримувати її в постійному актуальному стані.

Побудова системи навігації сайту найпростіше почати з створення файлу карти сайту, для цього необхідно зробити наступне:

- відкрити сайт і клацнути правою кнопкою миші по папці сайту;
- в меню вибрати пункт Add New Item; коли діалогове вікно Add New Item з'явиться, вибрати шаблон Карта сайту (Site Map);
- задати своє ім'я файлу або залишити Web.sitemap та натиснути кнопку Add.

За замовчуванням створюється файл Web.sitemap в папці сайту. Файл карти сайту являє собою XML-файл, тому Visual Studio відображає його в текстовому режимі. Щойно створений файл Web.sitemap виглядає наступним чином:

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File- 1.0"
>
  <siteMapNode url="" title=""      description="">
  <siteMapNode url="" title=""      description="" />
  <siteMapNode url="" title=""      description="" />
</siteMapNode>
</siteMap>
```

Формат цього файлу досить простий. Тег <? Xml> в першому рядку визначає версію XML і його міняти не потрібно. Тег <siteMap> у другому рядку і його закінчення </ siteMap> у сьомому рядку зазначає початок і кінець XML-документа. Кожен тег <siteMap> повинен містити один і тільки один тег верхнього рівня <siteMapNode>. Він зазвичай визначає стартову сторінку в карті сайту. У прикладі це тег <siteMapNode> в третьому рядку, а його

закінчення `</ siteMapNode>` в шостому рядку. Між тегами верхнього рівня `<siteMapNode>` і `</ siteMapNode>` можна додавати потрібну кількість додаткових тегів `<SiteMapNode>`. Кожен з них буде рівнем в меню або деревовидному списку. Ці теги розташовані в рядках чотири і п'ять. В межах тегів другого рівня `<siteMapNode>` можна додавати стільки тегів `<siteMapNode>` третього рівня та ін. Наприклад, для сайту "Гороскоп" запропонована наступна схема навігації (рис. 6.13), щоб її застосувати потрібно в файлі `Web.sitemap` створити структуру, представлену на рис. 6.14.

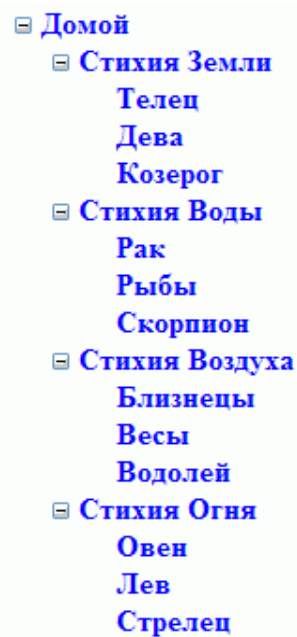


Рис. 6.13 – Карта сайту "Гороскоп"

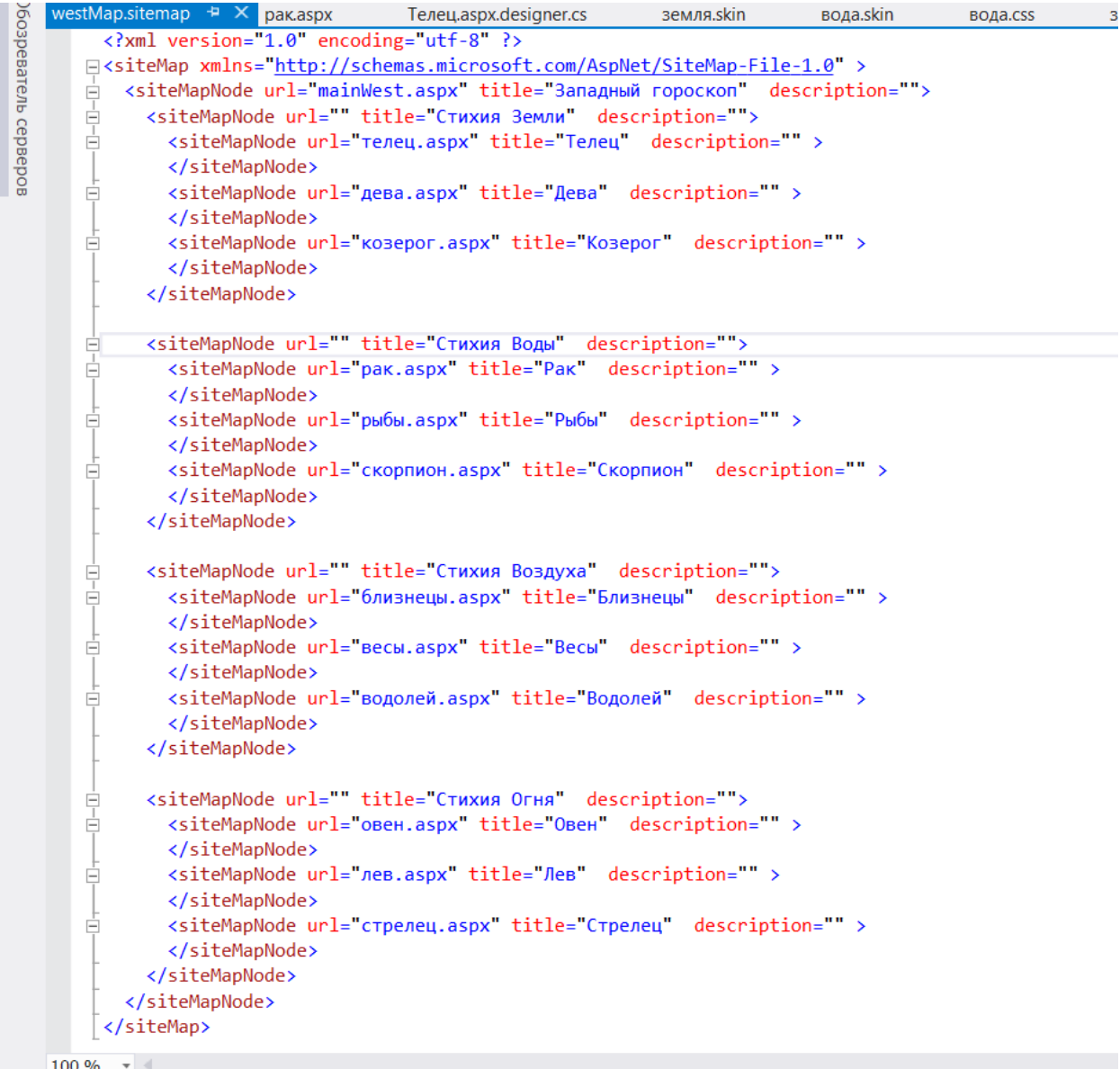
Тег `<siteMapNode>` має три ознаки, які можна змінювати (рис.6.14):

1) URL – абсолютне або відносне посилання на сторінку, куди буде здійснений перехід при виборі посилання в меню або деревовидному списку. Значення URL повинно бути унікальне, інакше з'явиться повідомлення про помилку. Щоб створити `<siteMapNode>`, який не пов'язаний зі сторінкою, необхідно залишити пробіл в якості значення URL. Це корисно для пунктів меню, які представляють групу сторінок, але самі ні на які сторінки не посилаються.

2) TITLE – назва посилання, яке з'явиться як видимий текст в меню, або деревовидному списку.

3) DESCRIPTION – більш довга назва посилання, яке з'явиться як текст

підказки, коли покажчик миші буде перебувати за межами. Ця ознака є додатковою і заповнювати її не обов'язково.



```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode url="mainWest.aspx" title="Западный гороскоп" description="" >
    <siteMapNode url="" title="Стихия Земли" description="" >
      <siteMapNode url="телец.aspx" title="Телец" description="" >
      </siteMapNode>
      <siteMapNode url="дева.aspx" title="Дева" description="" >
      </siteMapNode>
      <siteMapNode url="козерог.aspx" title="Козерог" description="" >
      </siteMapNode>
    </siteMapNode>

    <siteMapNode url="" title="Стихия Воды" description="" >
      <siteMapNode url="пак.aspx" title="Пак" description="" >
      </siteMapNode>
      <siteMapNode url="рыбы.aspx" title="Рыбы" description="" >
      </siteMapNode>
      <siteMapNode url="скорпион.aspx" title="Скорпион" description="" >
      </siteMapNode>
    </siteMapNode>

    <siteMapNode url="" title="Стихия Воздуха" description="" >
      <siteMapNode url="близнецы.aspx" title="Близнецы" description="" >
      </siteMapNode>
      <siteMapNode url="весы.aspx" title="Весы" description="" >
      </siteMapNode>
      <siteMapNode url="водолей.aspx" title="Водолей" description="" >
      </siteMapNode>
    </siteMapNode>

    <siteMapNode url="" title="Стихия Огня" description="" >
      <siteMapNode url="овен.aspx" title="Овен" description="" >
      </siteMapNode>
      <siteMapNode url="лев.aspx" title="Лев" description="" >
      </siteMapNode>
      <siteMapNode url="стрелец.aspx" title="Стрелец" description="" >
      </siteMapNode>
    </siteMapNode>
  </siteMapNode>
</siteMap>
```

Рис.6.14 – Код, який описує карту сайту "Гороскоп"

Описану в файлі Web.sitemap карту сайту можна використовувати для створення меню:

- відкрити майстер-сторінку або будь-яку іншу сторінку, яка буде відображати меню;
- в панелі Toolbox відкрити групу Data і перетягнути елемент SiteMapDataSource на відкриту сторінку; його можна помістити де

завгодно, тому що в будь-якому випадку при перегляді сторінки цей елемент відобразитися не буде; за замовчуванням для цього елемента призначено ім'я SiteMapDataSource1, ця назва з'явиться у вигляді напису на самому елементі і буде використана далі;

- в панелі Toolbox відкрити групу Navigation і перетягнути елемент Menu на відкриту сторінку в потрібне місце; Visual Studio 2015 відобразить меню так, як показано на рис. 6.15.
- відкрити список Choose Data Source і вибрати назву SiteMapDataSource1, яка було додано на кроці 2;
- зберегти сторінку та переглянути її (рис. 6.16).

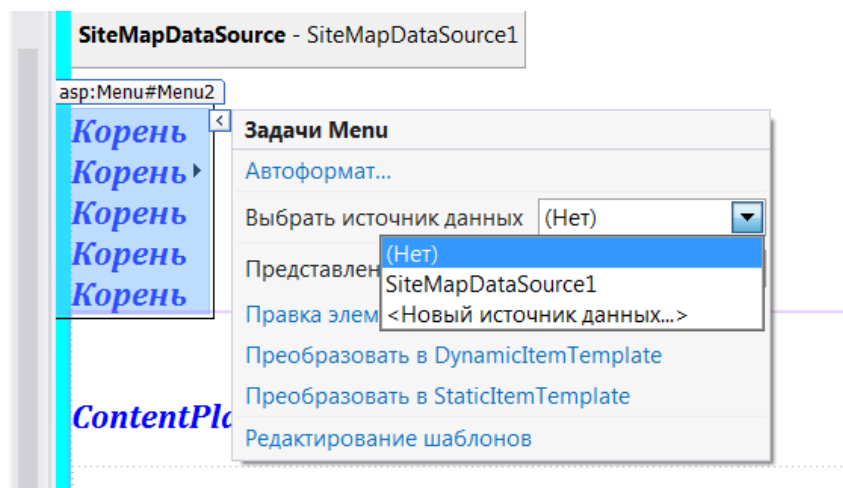


Рис. 6.15 – Прив'язка меню до карти сайту



Рис. 6.16 – Приклад меню

Щоб налаштувати конфігурацію меню, виділіть його, а потім визначте потрібні значення у вікні Properties. При встановленні властивості Orientation можна вибрати між значеннями Horizontal і Vertical:

1) *Horizontal*. Горизонтальні пункти меню Top-level з'являться зліва направо. Результат – система меню, що розкривається.

2) *Vertical*. Вертикальні пункти меню Top-level з'являться зверху вниз. Результат – система меню, що випадає.

Файл карти сайту можна також використовувати для створення навігації аналогічної Windows Explorer. Для цього потрібно скористатися елементом TreeView з групи Navigation. Тоді відвідувачі сайту зможуть використовувати значки плюс (+) і мінус (-), щоб розгорнути або згорнути будь-які групи елементів меню.

Щоб створити меню-список потрібно виконати наступні дії:

- відкрити майстер-сторінку або будь-яку іншу сторінку, яка буде відображати меню-список;
- в панелі Toolbox відкрити групу Data і перетягнути елемент SiteMapDataSource на відкриту сторінку; за замовчуванням для цього елемента буде призначено ім'я SiteMapDataSource1, ця назва з'явиться у вигляді напису на самому елементі;
- в панелі Toolbox відкрити групу Navigation і перетягнути елемент TreeView на відкриту сторінку в потрібне місце;
- Visual Studio 2015 відобразить меню, в якому потрібно відкрити список Choose Data Source і вибрати назву SiteMapDataSource1, яка була додана на кроці 2;
- зберегти сторінку і переглянути її (рис. 6.17).

Файл карти сайту можна використовувати для створення панелі, яка показує шлях від головної сторінки до поточної сторінки за допомогою елемента управління SiteMapPath. На рис. 6.18 елемент управління SiteMapPath розташовано внизу сторінки. Можливо додати елемент SiteMapPath до будь-якої .aspx веб-сторінці або майстер-сторінки. Для цього достатньо в панелі Toolbox відкрити групу Navigation і перетягнути елемент SiteMapPath на потрібну сторінку, яка відкрита в режимі конструктора.



Рис.6.17 – Приклад меню–список

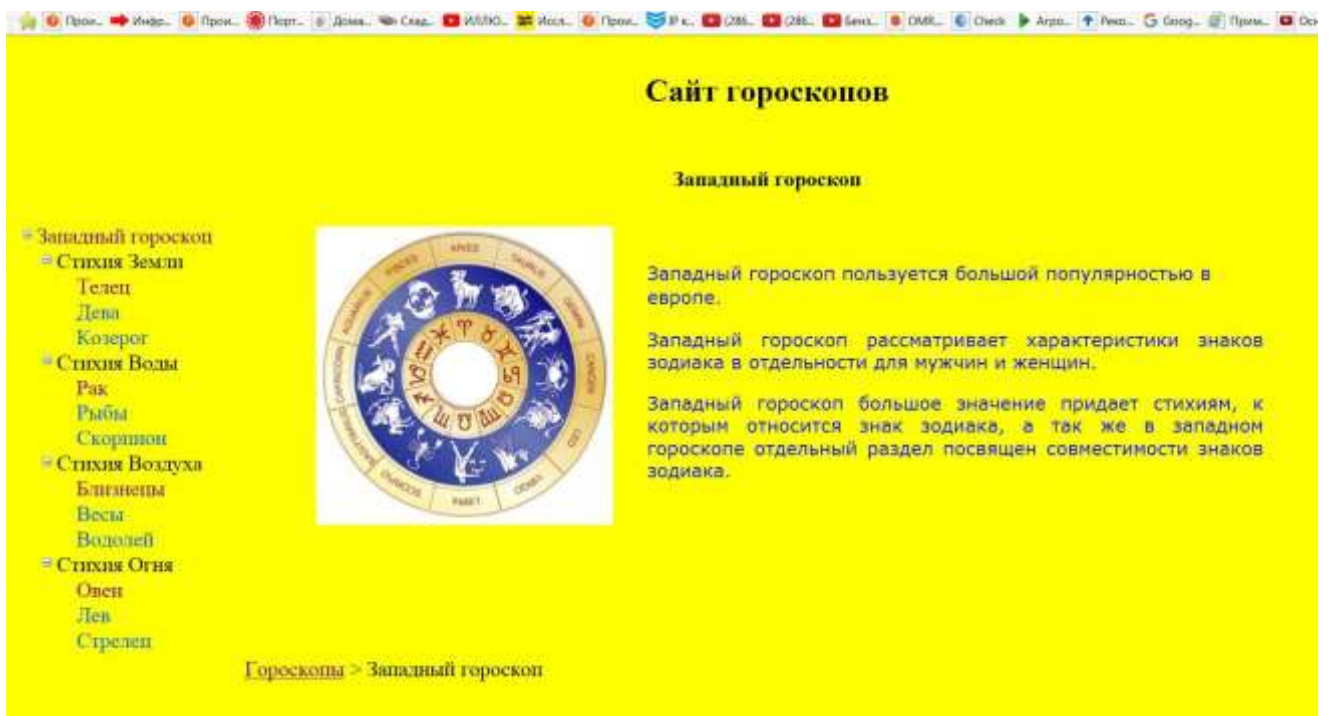


Рис. 6.18 – Приклад використання SiteMapPath

На відміну від елементів управління Menu і TreeView, елемент управління SiteMapPath не отримує дані від SiteMapDataSource, а завжди використовує файл карти сайту за замовчуванням Web.sitemap. Це може служити серйозною підставою, щоб не змінювати ім'я файлу карти сайту дане за замовчуванням.

Таким чином, було розглянуто чотири елементи управління, які допомагають організовувати гіперпосилання в межах сайту. Перший, SiteMapDataSource, забезпечує доступ до ієрархічних списків зв'язків, який закодований у файлі web.sitemap. Цей елемент не видно в режимі перегляду і він забезпечує даними елементи управління Menu і TreeView. Елемент управління Menu відображає, в залежності від налаштувань, меню, яке розкривається або випадає, а TreeView відображає ті ж самі дані у вигляді ієрархічного списку. Четвертий елемент управління, SiteMapPath, знаходить поточну сторінку в файлі web.sitemap і потім відображає шлях від поточної сторінки до домашньої сторінки сайту.

Завдання для виконання

ЗАВДАННЯ 6.1

Створити сайт, який містить гороскоп. Існує два види гороскопів: східний і західний. Західний гороскоп заснований на 12 знаках зодіаку, які змінюють один одного протягом року і підрозділяються на чотири групи, так звані стихії: вода, земля, вогонь і повітря. Китайський гороскоп, також заснований на 12 знаках зодіаку, але змінюють вони один одного протягом 12 років. Стихій в китайському гороскопі всього п'ять: залізо, вода, дерево, вогонь, земля і вони також змінюють один одного через рік. Таким чином, в китайському гороскопі повний цикл здійснюється за 60 років.

Необхідно створити дві майстер-сторінки, одну для відображення інформації західного гороскопу, другу для відображення інформації східного гороскопу. При розробці майстер-сторінок врахувати наступні рекомендації:

1. Кожна майстер сторінка повинна містити заголовок "Сайт гороскопів".
2. Вгорі кожної майстер сторінки повинен бути відповідний підзаголовок: "Східний гороскоп" або "Західний гороскоп".
3. Внизу майстер сторінок повинен стояти знак копірайту розробника, наприклад: © Іванов Іван Іванович, ОДЕКУ, Одеса, 2021 р. E-mail ivanov@gmail.com.

4. Зліва від елементу управління ContentPlaceholder має бути місце для подальшого розміщення карти сайту
5. Необхідно передбачити посилання переходу на домашню сторінку для тих користувачів, хто прийшов на сайт через пошукову систему.
6. На кожній майстер–сторінці повинен бути тематичний малюнок, що відображає сутність західного і східного гороскопів.
7. На кожній майстер сторінці повинен бути фрагмент тексту, що описує характеристики західного і східного гороскопів.

ЗАВДАННЯ 6.2

Створіть шість тем, які будуть використовуватися для оформлення гороскопів (табл. 6.1).

Таблиця 6.1

Відповідність стихій та гороскопів

Стихія	Західний гороскоп	Східний гороскоп
Вода	Присутній	Присутній
Земля	Присутній	Присутній
Вогонь	Присутній	Присутній
Повітря	Присутній	
Дерево		Присутній
Залізо		Присутній

При створенні теми використовуйте CSS–файли і skin–файли. В CSS–файлі для кожної теми виберіть різні параметри шрифту (колір, розмір, начертання), колір фона задайте у відповідності до стихії. У skin–файлі задайте зображення для кожної стихії.

ЗАВДАННЯ 6.3

Зробіть три файли карти сайту. У кожному з файлів розмістіть посилання на наступні веб–сторінки:

Загальна карта сайту

західний гороскоп

східний гороскоп

інформація про розробників
сторінка заповнення анкети
сторінка отримання прогнозу

Східний гороскоп

Щур

Бик

Тигр

Кролик

Дракон

Змія

Кінь

Вівця

Мавпа

Півень

Собака

Свиня

Західний гороскоп

стихія землі

Тілець

Діва

Козеріг

стихія води

Рак

Риби

Скорпіон

стихія повітря

Близнюки

Ваги

Водолій

Стихія вогню

Овен

Лев

Стрілець

В кожному файлі карти сайту заповніть поля DESCRIPTION розгорнутим описом посилань. Загальну карту сайту використовуйте для заповнення елемента управління Menu на домашній сторінці сайту. Файл карти сайту "Східний гороскоп" використовуйте для заповнення елемента управління TreeView, який розмістіть на майстер-сторінці "Східний гороскоп". Файл карти сайту "Західний гороскоп" використовуйте для заповнення елемента управління TreeView, який розташуйте на майстер-сторінці "Західний гороскоп". На обох майстер-сторінках замініть посилання на домашню сторінку елементом управління SiteMapPath.

ЗАВДАННЯ 6.4

Створити базу даних "Зодіак" з наступними таблицями:

- 1) Анкета – особисті дані про користувача сайту.
- 2) Опис знаків – коротка характеристика знака зодіаку.
- 3) Прогноз – прогноз для кожного знака зодіаку на конкретну дату.

ЗАВДАННЯ 6.5

Створіть форми для кожної таблиці бази даних (табл. 6.2).

Таблиця 6.2

Форми для таблиць бази даних

Таблиця	Призначення форм	
	DetailsView	GridView
Анкета	Додавання анкети користувачем	Форма відсутня
Опис знаків	Перегляд докладної характеристики того чи іншого знака	Перегляд коротких характеристик по кожному знаку
Прогноз	Перегляд прогнозу на конкретну дату	Перегляд прогнозів за певний період

Контрольні питання:

1. Як створити майстер-сторінку? Для чого вона використовується?
2. Як створити контент-сторінку? Для чого вона використовується?
3. В чому полягає різниця між майстер- та контент-сторінкою?

4. Для чого використовується елемент управління ContentPlaceHolder?
5. Як задати стиль для певного елемента сторінки?
6. Опишіть вміст файлу з таблицями стилів.
7. Вкажіть способи додавання властивостей в правила стилю.
8. Як зв'язати створений файл стилів з веб-сторінкою?
9. Що таке «тема»? Для чого використовується?
10. Яке призначення skin-файлів? В чому різниця між css-файлами та skin-файлами?
11. Як створити карту сайту?
12. Як використовувати карту сайту для створення меню?

ЛІТЕРАТУРА

Основна

1. Гнатовська Г.А. Технологія створення WEB–застосувань: конспект лекцій / Одеса: ОДЕКУ, 2016. 139 с.
2. Методичні вказівки до виконання лабораторних робіт з дисципліни «Технології створення WEB–застосувань» /Укладачі: Волощук Л.А., Гнатовська Г.А. Одеса: ОДЕКУ, 2016. 79 с.

Додаткова

1. Мак–Дональд М. Microsoft ASP.NET 4 с прикладами на С# 2010 для професіоналов. – М. : ООО "И.Д. Вильямс", 2011. 1424 с.
2. Сандереон Стивен. ASP.NET MVC Framework с прикладами на С# для професіоналов. – М. : ООО "И.Д. Вильямс", 2010. 560 с
3. Троелсен Э. Язык программирования С# 2010 и платформа .NET 4.0. – М. : ООО "И.Д. Вильямс", 2011. 1392 с.
4. Microsoft ASP.NET 2.0. Базовый курс. Мастер–класс. – СПб.: Питер, 2007. 688 с.

Інформаційні ресурси

1. Бельчусов А. Академия Microsoft: Разработка интерактивных сайтов с помощью Microsoft Visual Web: <https://intuit.ru/studies/courses/1142/252/info>.
2. Сайт ASP.NET. URL: <http://www.asp.net/>.
3. Сайт компанії Microsoft. URL: www.microsoft.com.
4. Довідкове керівництво по ASP.NET. URL: <http://msdn.microsoft.com/ruru/asp.net/default.aspx>.
5. Довідкове керівництво по MySQL. URL: www.mysql.ru/docs/man/.
6. Довідник по Web–мовам. URL: www.spravkaweb.ru/.

Електронна бібліотека ОДЕКУ www.library-odeku.16mb.com