СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки
Кафедра програмування та математики

ПОЯСНЮВАЛЬНА ЗАПИСКА
до кваліфікаційної випускної роботи

освітній ступінь бакалавр
спеціальність 121 „Інженерія програмного забезпечення”
(шифр і назва спеціальності)
спеціалізація „Інженерія програмного забезпечення”

на тему „ Реалізація веб-ресурсу надання послуг для розміщення нформації ”

Виконав: студент групи ІПЗ-16д _____ Д.Ю. Мирошниченко
(підпис) (ініціали і прізвище)

Керівник _____ Ю.Г. Ковальов
(підпис) (ініціали і прізвище)

Завідувач кафедри _____ В.О. Лифар
(підпис) (ініціали і прізвище)

Рецензент_____

Сєвєродонецьк – 2020

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки
Кафедра програмування та математики

Освітній ступінь бакалавр
спеціальність 121 „Інженерія програмного забезпечення”
(шифр і назва спеціальності)
спеціалізація „Інженерія програмного забезпечення”
(назва спеціалізації)

ЗАТВЕРДЖУЮ
Завідувач кафедри ПМ,
д.т.н., доцент
_____ Лифар В.О.
“____” _____2020 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ ВИПУСКНУ РОБОТУ СТУДЕНТУ

Мирошниченко Данило Юрійович
(прізвище, ім’я, по батькові)

1.Тема роботи Реалізація веб-ресурсу надання послуг для розміщення нформації.

Керівник роботи к.ф-м.н. Ковальов Ю.Г. ,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджений наказом університету від “___”_____20__року №___

2. Строк подання студентом роботи 20 травня 2020 р.

3. Вихідні дані до роботи Об'єктом даної роботи є процес розробки сучасного сайту

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Вступ. Аналітичний огляд, з висвітленням наступних питань: елементи уравління, основи веб-розробки, клієнт і сервер. Основна частина, в якої висвітлити: збір вимог та проектування. Висновки. Перелік використаних джерел.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслеників)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання__30 березеня 2020 року. _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання кваліфікаційної випускної роботи	Строк виконання етапів	Примітка
1	Одержання завдання на виконання роботи	30.03.20	
2	Укладання і погодження з керівником плану і етапів виконання роботи	06.04.20	
3	Узагальнення даних літературних джерел, укладання розділу «Аналіз предметної галузі»	13.04.20	
4	Аналіз шляхів виконання завдання. Вибір і погодження керівником оптимального шляху	20.04.20	
5	Укладання та тестування програмного продукту	27.04.20	
6	Укладання, оформлення та погодження пояснювальної записки з керівником	04.05.20	
7	Здача готової пояснювальної записки на кафедру	12.05.20	
8	Укладання доповіді і презентації	30.05.20	

Студент _____ Д.Ю. Мирошниченко _____
(підпис) (ініціали і прізвище)

Керівник роботи _____ Ю.Г. Ковальов_____
(підпис) (ініціали і прізвище)

ЛИСТ ПОГОДЖЕННЯ І ОЦІНЮВАННЯ

дипломної роботи студента гр. ІПЗ-16д Мирошниченко Д.Ю.

Науковий керівник

Доцент, ф-м.т.н. _____ Ковальов Ю.Г.

Оцінка наукового керівника: _____

Рецензент _____

ПІБ, місто роботи, посада

Оцінка рецензента: _____

Кінцева оцінка за результатами захисту:

_____

Голова ЕК

_____ Лифар В.О.

підпис

РЕФЕРАТ

Робота містить: 40 сторінок основного тексту, 25 сторінок додатків, 9 рисунків, 3 таблиці, 9 використаних джерел.

Метою випускної кваліфікаційної роботи є завдання виявити найбільш зручне та доступне середовище розробки WEB-сайта.

Було проаналізовано багато конструкторів сайтів, їх принцип роботи, перелік наданих можливостей і кращі рішення взаємодії з користувачами..

В результаті виконаної роботи, було реалізовано розробку WEB-сайта

Система реалізована відповідно всім вимогам технічного завдання. Зроблено детальний опис процесу розробки системи, а також, продемонстрована робота готового WEB-сайта.

ЗМІСТ

ВСТУП

Актуальність досліджень: Навчившись створювати цілісні веб-сторінки, програміст починає замислюватися про загальну картину - тобто про групуванні великого числа веб-сторінок в єдиний логічно пов'язаний веб-сайт.

Однак програміст стикається також з такими завданнями, як забезпечення логічної зв'язності всіх сторінок і спрощення навігації по веб-сайту. У цій частині ми розглянемо питання, які стають актуальними, коли ви перестаєте думати про окремих сторінках і переходите до планування веб-додатки в цілому.

В даний час, у зв'язку з глобальним розвитком мережі Інтернет, в програмуванні все більш різко виділятися окрема його галузь - web-програмування. Спочатку, воно не могло навіть зрівнятися за своєю складністю з іншими областями «програміста ремесла», не «дотягуючись" не тільки до системного, але навіть і до прикладного програмування. Йдеться, звичайно, про програмування сценаріїв для інтернет сайтів, або, Web-програмуванні.

Під час стрімкого прогресу просто красиво оформлений текст і картинки на веб-сайті вже нікого не здивують. Вимоги до сайтів, змінилися - тепер для успішного представництва компанії в Інтернет необхідно надати своїм відвідувачам різні можливості: зворотний зв'язок, форум, голосування, інтернет-магазин, різні web-тести, пошук по сайту, лічильник відвідувань і багато іншого. За допомогою звичайного html цього не досягти, адже html - це мова гіпертекстової розмітки - інструмент для створення гіперпосилань, вставки зображень, таблиць та ін. За допомогою мови html легко і швидко можна зробити форму для відправки будь-якого запиту. Така форма буде мати всі необхідні атрибути: і поле для введення тексту, і кнопку відправки.

Однак, при натисканні на таку кнопку в більшості випадків не відбудеться зовсім нічого - адже не було наведено сценарій дій, які слід виконати, щоб отримати результат. Тому створення інтерактивних компонентів - це завдання для web-програмування.

Веб-програмування здійснюється за допомогою спеціальних програмних засобів - скриптів. Ці програмні засоби поділяються на два основних види: серверні і клієнтські. Серверні скрипти виконуються на стороні сервера, тобто того комп'ютера, на якому розміщений сайт. Вони виконуються ще до завантаження сторінок сайту на комп'ютер користувача. У свою чергу, клієнтські скрипти виконуються на комп'ютері клієнта вже після завантаження сторінки з сервера і не вимагають її додаткового перезавантаження.

Мови веб - програмування, на яких виконуються і ті, і інші скрипти різні. Деякі з мов використовуються тільки для створення серверних скриптів, інші - тільки для клієнтських, а багато мов - для тих і інших.

Методи дослідження: Для вирішення поставлених задач застосовані:

• загальнонаукові методи: теоретичного пошуку; концептуально-порівняльного аналізу, визначення теоретичних і прикладних аспектів дослідження, визначення структури і змісту підготовки;

• емпіричні методи: дослідження предметної області, дослідження об’єкту, що вивчається у штучно створених для нього умовах, порівняння об’єкту, що досліджується з аналогом;

Об'єктом дослідження: є процес розробки сайту.

Завдання дослідження: а) скласти інформаційну модель;

б) розкрити Web-технології розробки WEB-сайта

в) створити розробку WEB-сайту

Мета дослідження: Виявити найбільш зручне та доступне середовище розробки WEB-сайта.

РОЗДІЛ 1 АНАЛІТИЧНИЙ ОГЛЯД

Веб-сайт (англ. Website, від web - Павутина и site - "місце") - це одна сторінка або сукупність веб-сторінок, доступних в Інтернеті через протоколи HTTP / HTTPS. Веб-сайт - це місце в інтернеті, відзначається своєю адресою (URL), має свого власника и складається з веб-сторінок, Які сприймаються як єдине ціле.

Сукупність всіх загальнодоступних веб-сайтів є Всесвітня Павутина. Веб-сайти - це збірники документів, відомих як веб-сторінки (або Сторінки для стислості). Смороду містять Деяк інформацію: зображення, текст, відео-, аудіо матеріалі та інше. Головна сторінка веб-сайту називається домашньою, з неї можна перейти на всі інші Сторінки. Смороду пов'язані гіперпосиланням, Які виділені особливо видом шрифту або представлені за допомогою збережений. Сторінки веб-сайту об'єднані загальною корінною адресою, а також э темою, логічною структурою, оформленням або авторством. Сторінки веб-сайтів - це файли з текстом, розміченім на мові HTML або XHTML. ЦІ файли, будучи завантаження відвідувачем на его комп'ютер, обробляються програмою-оглядач, звання браузером (наприклад, Internet Explorer, Firefox, Google Chrome) и виводяться на засіб відображення користувача (монітор, екран КПК, принтер або синтезатор мови). Мова HTML / XHTML дозволяє форматувати текст, розрізняти в ньому функціональні елементи, створювати гіпертекстові посилання (гіперпосілання) и вставляти в сторінку, що відображається, зображення, звукозаписи и інші.

Мультимедійні елементи. Відображення сторінки можна змінити. Додавання в неї таблиці стилів на мові CSS або сценаріїв на мові JavaScript.

Виготовлення сайтів як працюючих цілісних інформаційних ресурсів є Складення процес, що залучає працю різних спеціальностей.

1.1 ASP.NET Web Forms

ASP.NET Web Forms є основою веб - додатків і одна з декількох моделей програмування, підтримуваних Microsoft ASP.NET технології. Web Forms додатки можуть бути написані на будь-якій мові програмування, який підтримує Common Language Runtime, такі як C # або Visual Basic. Основні будівельні блоки Web Forms сторінок серверні елементи управління, які багаторазово використовуються компоненти, відповідальні за надання HTML - розмітку і реагувати на події. Методика називається вид стан використовується для зберігатися станом серверних елементів управління між нормально особами без HTTP запитів.

Веб - форми була включена в початковий .NET Framework версії 1.0 у 2002 році (див .NET Framework версії історії і ASP.NET історію версій), як перша модель програмування, доступних в ASP.NET. На відміну від нових компонентів ASP.NET, Web Forms не підтримує ASP.NET Ядра.

Веб - сторінки ASP.NET, відомі офіційно як веб - форм, є основними будівельними блоками для розробки додатків в ASP.NET. Існують дві основні методики для веб - форм, формат веб - додатків і формат веб - сайту. Веб - додатки повинні бути скомпільовані перед розгортанням, в той час як веб - сайти структур дозволяє користувачеві копіювати файли безпосередньо на сервер без попереднього компіляції. Веб - форми містяться в файлах з розширенням «.aspx»; ці файли зазвичай містять статичну (X) HTML - розмітку або компонент розмітку. Компонент розмітки може включати в себе веб - елементи управління на стороні сервера і призначених для користувача елементів управління, які були визначені в рамках або веб - сторінки. Наприклад, компонент текстового поля може бути визначений на сторінці, як <asp: textbox id = ' myid 'runat =' server '>, яка надається в поле введення HTML. Крім того, динамічний код, який виконується на сервері, можуть бути

розміщені на сторінці в блоці <% - dynamic code -%>, який схожий на інші технології веб - розробки, таких як PHP, JSP і ASP. З ASP.NET Framework 2.0, Microsoft представила новий код-за моделлю, яка дозволяє статичний текст залишається на сторінці .aspx, в той час як динамічний код залишається в .aspx.vb або .aspx.cs або .aspx.fs файл (в залежно від мову програмування, що використовується). Microsoft рекомендує справу з динамічним кодом програми, використовуючи модель коду позаду, який поміщає цей код в окремому файлі або в спеціально відведеному тегом. Код через файли зазвичай мають імена, як «MyPage.aspx.cs» або «MyPage.aspx.vb», а файл підкачки MyPage. aspx (те ж ім'я, як файл підкачки (ASPX), але з кінцевим розширенням позначивши сторінку мову). Ця практика є автоматичним в Visual Studio і інших Іди, хоча користувач може змінити сторінку коду позаду. Крім того, в форматі веб - додатки, то pagename.aspx.cs частковий клас, який пов'язаний з файлом pagename.designer.cs.

Файл дизайнер представляє собою файл, який автоматично зі сторінки ASPX і дозволяє програмісту довідкових компонентів на сторінці ASPX зі сторінки CS без оголошення їх вручну, як це було необхідно в версіях ASP.NET до версії 2. При використанні цього стиль програмування, розробник пише код, щоб реагувати на різні події, такі як сторінки завантажується, або контрольний бути натиснуто, а не процедурна покроковим документ. але з кінцевим розширенням позначивши сторінку мову). Ця практика є автоматичним в Visual Studio і інших Іди, хоча користувач може змінити сторінку коду позаду. Крім того, в форматі веб - додатки, то pagename.aspx.cs частковий клас, який пов'язаний з файлом pagename.designer.cs.

Фоновим кодом в ASP.NET в моделі знаменує собою відхід від класичного ASP в тому, що він заохочує розробників створювати додатки з поділом подання та змісту на увазі. У теорії, це дозволить веб - дизайнер, наприклад, зосередити увагу на дизайн розмітки з меншим потенціалом для

порушення програмного коду, який управляє його. Це схоже на поділ контролера від уявлення в модель-уявлення-контролер (MVC) рамки.

ASP.NET додатки розміщуються на веб - сервері, а доступ до них за допомогою особи без HTTP протоколу. Таким чином, якщо додаток використовує динамічну взаємодію, воно повинно здійснювати державне управління за своїм власним. ASP.NET надає різні функції для державного управління. Концептуально, Microsoft розглядає «стан» в GUI стані. Проблеми можуть виникнути, якщо програма має відстежувати стан «дані»; наприклад, кінцевий автомат, який може знаходитися в перехідному стані між запитами (ледачою оцінкою) або займає багато часу для ініціалізації. Державне управління в сторінках ASP.NET з аутентифікацією може зробити веб - зішкріб важко або неможливо.

Базовий набір елементів управління ASP.NET великий і значний. Він охоплює елементи управління, які включають в себе базові HTML-дескриптори, і елементи управління, що надають розвинену високорівневу модель, такі як Calendar, TreeView і елементи управління даними. Звичайно, навіть найкращий набір елементів управління не в змозі задовольнити потреби всіх розробників. Рано чи пізно доведеться зайнятися створенням власних компонентів для користувача інтерфейсу.

Платформа .NET допускає два способи включення власних елементів управління в структуру веб-форм. Можна розробити такі компоненти:

1.2 Користувальницькі елементи управління

Призначений для користувача елемент управління - невелика частина сторінки, яка може містити статичний HTML-код і елементи управління веб-сервера. Перевага для користувача елементів управління полягає в тому, що після того як якийсь елемент створений, його можна повторно

використовувати в декількох сторінках одного веб-додатки. У нього навіть можна додавати власні властивості, події і методи.

1.3 Спеціальні серверні елементи управління

Спеціальні серверні елементи управління охоплюють класи, які програмно генерують власну HTML-розмітку. На відміну від призначених для користувача елементів управління (подібно сторінкам веб-форм оголошуються як простий текстовий файл), серверні елементи управління завжди попередньо компілюються в DLL-збірки. Залежно від способу формування коду серверного елемента управління, візуалізація вмісту може виконуватися з нуля, за рахунок успадкування зовнішнього вигляду і поведінки існуючого веб-елемента управління і розширення його функціональності або шляхом побудови інтерфейсу за рахунок створення екземплярів і конфігурації групи складових елементів управління.

Досліджуємо першу можливість - застосування призначених для користувача елементів управління. Призначені для користувача елементи управління служать прекрасним засобом стандартизації повторюваного вмісту у всіх сторінках веб-сайту. Наприклад, припустимо, що користувачам потрібно надати однаковий спосіб введення адресної інформації на декількох різних сторінках. Для вирішення цього завдання можна було б створити окремий елемент управління адресою, який об'єднує текстові поля і кілька пов'язаних з ними функцій перевірки. Потім цей елемент управління адреси можна було вставляти в будь-яку веб-форму і програмно працювати з ним як з єдиним об'єктом.

Призначені для користувача елементи управління добре підходять також для побудови і багаторазового використання колонтитулів сайтів і навігаційних засобів.

1.4 Основи призначених для користувача елементів управління

Файли для користувача елементів управління (.ascx) подібні файлів веб-форм ASP.NET (.aspx). Як і веб-форми, призначені для користувача елементи управління складаються з інтерфейсу користувача з дескрипторами елементів управління (файл .ascx) і можуть застосовувати вбудовані сценарії або файли відокремленого коду (.cs). Призначені для користувача елементи управління можуть містити практично всі компоненти, властиві веб-сторінці, включаючи статичне HTML-вміст і елементи управління ASP.NET, і отримують ті ж події, що і об'єкт Page (на зразок Load і PreRender), і за допомогою властивостей (таких як Application , Session, Request і Response) відображають той же набір об'єктів, властивих ASP.NET.

Нижче перераховані основні відмінності між призначеними для користувача елементами управління і веб-сторінками:

Призначені для користувача елементи управління починаються з директиви Control, а не Page.

Призначені для користувача елементи управління використовують розширення .ascx, а не .aspx, а їх файли відокремленого коду успадковуються від класу System.Web.UI.UserControl. Фактично класи UserControl і Page успадковані від одного і того ж класу TemplateControl, що обумовлює спільність багатьох їхніх методів і властивостей.

Призначені для користувача елементи управління не можуть запитуватися безпосередньо клієнтським браузером. (При будь-якій подібній спробі ASP.NET буде виводити узагальнене повідомлення про помилку "that file type is not served" ("не обслуговуються тип файлу"). Натомість призначені для користувача елементи управління впроваджуються в інші веб-сторінки.

1.5 Створення простого користувача елементу управління

Щоб створити елемент управління в Visual Studio, виберіть команду меню Website -> Add New Item (Веб-сайт -> Додати новий елемент), а потім вкажіть шаблон Web User Control (для користувача веб-елемент керування).

Іноді при розробці призначеного для користувача елементу управління найпростіше спочатку помістити його на веб-сторінку, протестувати її, а потім перетворити в призначений для користувача елемент управління. Навіть при відмові від цього підходу, в кінцевому рахунку, може виявитися, що частина інтерфейсу користувача потрібно витягти з сторінки і задіяти в декількох місцях.

В цілому цей процес зводиться до вирізання та вставляння. Однак при цьому слід приділити увагу кількох моментів:

Видаліть всі дескриптори <html>, <head>, <body> і <form>. Вони повинні бути присутніми на сторінці тільки в одному екземплярі, тому не можуть бути додані в призначені для користувача елементи управління (які можуть багаторазово зустрічатися в одній сторінці). Видаліть також дескриптор типу документа (DOCTYPE).

При наявності директиви Page замініть її директивою Control і видаліть всі атрибути, які не підтримуються директивою Control, такі як AspCompat, Buffer, ClientTarget, CodePage, Culture, EnableSessionState, EnableViewStateMac, ErrorPage, LCID, ResponseEncoding, Trace, TraceMode і Transaction.

Якщо модель відокремленого коду не використовується, переконайтеся, що ім'я класу включено в директиву Control за допомогою атрибута ClassName. Завдяки цьому веб-сторінка, яка використовує даний елемент управління, може бути строго типізований, що дозволить їй

отримувати доступ до властивостей і методів, доданим в елемент управління. У разі застосування моделі відокремленого коду клас відокремленого коду потрібно змінити так, щоб він дістався у спадок від класу UserControl, а не Page.

Змініть розширення файлу з .aspx на .ascx.

РОЗДІЛ 2 ПІДГОТОВКА ДО РЕАЛІЗАЦІЇ

Якщо веб-сайт містить досить багато сторінок, можливо, буде потрібно якась система навігації, яка дозволить користувачеві переходити від однієї сторінки до іншої. Як було сказано раніше, для визначення шаблону сайту, що включає панель навігації, можна застосовувати майстер-сторінки. Але заповнювати цю панель навігації доведеться самостійно.

Очевидно, що для реалізації практично будь-якої системи навігації можна використовувати набір елементів управління ASP.NET, але всю важку роботу все ж доведеться виконувати вручну. На щастя, ASP.NET пропонує новий набір функціональних засобів навігації, які суттєво спрощують стоїть завдання.

У кращих традиціях засобів ASP.NET навігація в ASP.NET є гнучкою, конфігурується і підключається. Вона складається з трьох компонентів:

Спосіб визначення структури навігації по веб-сайту. Цей компонент є XML-карту сайту, яка (за замовчуванням) зберігається в файлі.

Зручний спосіб синтаксичного аналізу файлу карти сайту і перетворення його інформацією в відповідну об'єктну модель. Ці завдання виконуються елементом управління SiteMapDataSource і компонентом XmlSiteMapProvider.

Спосіб використання інформації карти сайту для відображення поточної позиції користувача і надання йому можливості без праці переходити з одного місця в інше. Ця частина реалізується за допомогою елементів управління, пов'язаних з елементом управління SiteMapDataSource; до них відносяться навігаційні ланцюжки ("хлібні крихти"), списки, меню і дерева.

Кожна з цих складових може налаштовуватися і розширюватися незалежно від інших. Наприклад, якщо потрібно змінити зовнішній вигляд елементів управління навігації, потрібно просто зв'язати різні елементи управління з елементом управління SiteMapDataSource. З іншого боку, якщо необхідно прочитати інформацію карти сайту в іншому форматі або з іншого місця, знадобиться змінити постачальника карти сайту.

Взаємозв'язок між цими компонентами показана на малюнку нижче:

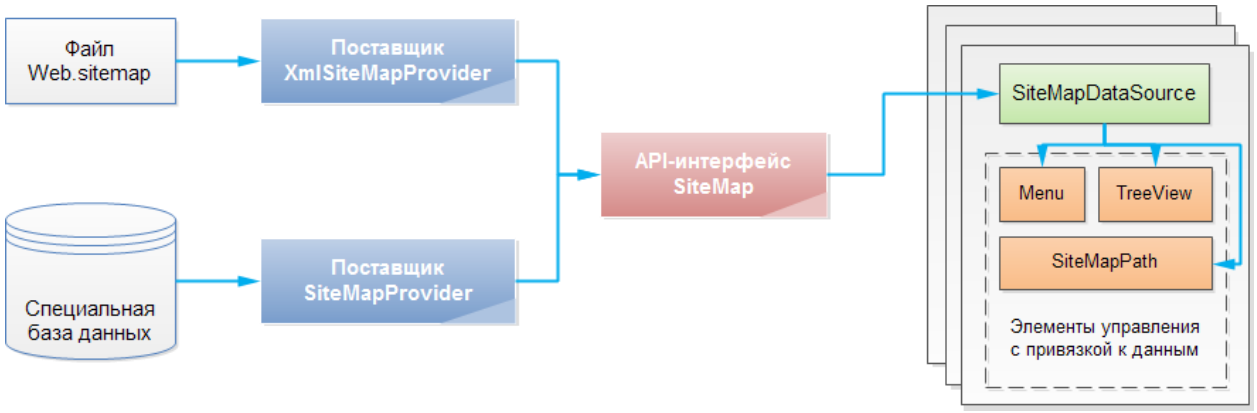

Рис. 2.1 - Взаємозв'язок компонентів

2.1 Визначення карти сайту

Відправною точкою в системі навігації сайту, побудованої на основі карти, є постачальник карти сайту. ASP.NET поширюється з єдиним постачальником карти сайту XmlSiteMapProvider, який може витягувати інформацію про карту сайту з XML - файла. Якщо потрібно отримати карту сайту з іншого розташування або в спеціальному форматі, доведеться створити власний постачальник карти сайту - ця тема розглядається в наступній статті.

Постачальник XmlSiteMapProvider шукає файл Web.sitemap в корені віртуального каталогу. Подібно до всіх постачальникам карт сайту, його

завдання полягає в отриманні даних карти сайту і створенні відповідного об'єкта SiteMap. Потім цей об'єкт SiteMap може бути зроблений доступним іншим елементам управління за допомогою SiteMapDataSource.

Щоб перевірити його в дії, створіть файл Web.sitemap і визначте структуру веб-сайту за допомогою елементів <siteMap> і <siteMapNode>. Щоб додати карту сайту в Visual Studio, виберіть пункт меню Website -> Add New item, вкажіть шаблон Site Map і клацніть на кнопці Add.

Базова структура файлу карти сайту, має такий вигляд:

```
<? Xml version = "1.0" encoding = "utf-8"?>

<SiteMap xmlns = "http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">

 <SiteMapNode>

 <SiteMapNode ...> </ siteMapNode>

 <SiteMapNode ...> </ siteMapNode>

 </ SiteMapNode>

</ SiteMap>
```

Допустима карта сайту повинна починатися з кореневого вузла <siteMap>, за яким слідує елемент <siteMapNode>, який представляє домашню сторінку за замовчуванням. В кореневий елемент <siteMapNode> можна вкладати будь-яке потрібне кількість рівнів інших елементів <siteMapNode>.

Кожен вузол карти сайту повинен містити заголовок, опис і URL-адресу, як показано в наступному прикладі:

```
<SiteMapNode url = "~ / default.aspx" title = "Головна" description = "Головна сторінка сайту">
```

У цьому прикладі для вказівки URL-адреси використаний синтаксис відносного шляху ~ /, який вказує на кореневу папку веб-додатки. Цей стиль не обов'язковий, але настійно рекомендується, оскільки він гарантує правильну інтерпретацію посилань карти сайту незалежно від поточної папки.

Тепер елемент <siteMapNode> можна використовувати для створення карти сайту. Єдине додаткове обмеження полягає в тому, що не можна створювати два вузла карти сайту, які мають однаковий URL-адресу.

У самій системі навігації обмеження на дублювання URL-адрес відсутня. Просто це обмеження накладається постачальником карти сайту XmlSiteMapProvider, оскільки він застосовує URL-адресу в якості унікального ключа. Якщо створити власний постачальник карти сайту або скористатися стороннім постачальником, можна дозволити дубльовані URL-адреси і зажадати окремого вказівки інформації про ключі. Однак дотримання правила про те, що кожен сайт повинен починатися з одного кореневого вузла, є обов'язковим, оскільки воно реалізовано в базовому класі SiteMapProvider. (Як незабаром буде показано, як і раніше можна застосовувати різні варіанти відображення дерева карти сайту, однак починати завжди потрібно з одного домашнього вузла.)

Нижче наведено приклад карти сайту:

<? Xml version = "1.0" encoding = "utf-8"?>

<SiteMap xmlns = "http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">

<SiteMapNode url = "~ / default.aspx" title = "Головна" description = "Головна сторінка сайту">

<SiteMapNode title = "Товари" description = "Наші товари" url = "~ / Products.aspx">

<SiteMapNode title = "Комплектуючі ПК" description = "Процесори, відеокарти, жорсткі диски"

url = "~ / Hardware.aspx" />

<SiteMapNode title = "Програми" description = "Visual Studio, Expression Blend, Windows Server"

url = "~ / Software.aspx" />

</ SiteMapNode>

<SiteMapNode title = "Служби" description = "Наші служби тех. Підтримки, навчання та ін."

url = "~ / Services.aspx">

<SiteMapNode title = "Навчання" description = "Ми навчимо використовувати вас наші програмні продукти"

url = "~ / Training.aspx" />

<SiteMapNode title = "Консультації" description = "Задавайте ваші питання"

url = "~ / Consulting.aspx" />

<SiteMapNode title = "Тих. Підтримка" description = "Допомагаємо 24/7"

url = "~ / Support.aspx" />

</ SiteMapNode>

</ SiteMapNode>

</ SiteMap>

У цьому прикладі всі вузли містять URL-адреси - тобто на них можна клацати (в результаті чого користувачі переходять до певних сторінок). Однак якщо ці вузли повинні служити просто категоріями для організації

інших посилань, атрибут url можна опустити. Вузол як і раніше буде перебувати серед прив'язаних елементів управління, але не буде відображатися у вигляді посилання.

2.2 Прив'язка до карти сайту

Як тільки файл Web.sitemap визначено, його можна використовувати в сторінці. Тут якраз дуже зручно застосовувати майстер-сторінки, щоб елементи управління навігацією можна було визначати як частини шаблону і повторно їх використовувати в кожній сторінці. Нижче показано, як в майстер-сторінці SiteTemplate.master визначити базову структуру, яка поміщає елементи управління навігації зліва і створює об'єкт SiteMapDataSource, що надає навігаційну інформацію іншим елементам управління:

```
<% @ Master Language = "C #" AutoEventWireup = "true" CodeFile =
"SiteTemplate.master.cs"

 Inherits = "SiteTemplate"%>

<! DOCTYPE html>

<Html xmlns = "http://www.w3.org/1999/xhtml">

<Head runat = "server">

 <Title> </ title>

</ Head>

<Body>

 <Form id = "form1" runat = "server">

 <Div>
```

```
<Table>
<Tr>
<Td style = "width: 226px; vertical-align: top;">
<! - Тут будуть розміщуватися елементи навігації ->
</ Td>
<Td style = "vertical-align: top; color: green; padding-left: 20px">
<Asp: ContentPlaceHolder ID = "ContentPlaceHolder1" runat = "server" />
</ Td>
</ Tr>
</ Table>
</ Div>
<Asp: SiteMapDataSource ID = "SiteMapDataSource1" runat = "server" />
</ Form>
</ Body>
</ Html>
```

Тепер можна створити дочірню сторінку з простим статичним вмістом:

```
<% @ Page Language = "C #" AutoEventWireup = "true" CodeFile =
"Default.aspx.cs"
 Inherits = "_ Default" Title = "Головна"%>
<Asp: Content ID = "Content1" runat = "server" ContentPlaceHolderID =
"ContentPlaceHolder1">
 <H1> Головна </ h1>
```

</ Asp: Content>

Єдине, що залишилося зробити - це вибрати елементи керування для відображення даних карти сайту. В якості вирішення, придатного на всі випадки життя, можна скористатися елементом управління TreeView. Можна додати TreeView і прив'язати його до елементу управління SiteMapDataSource на майстер-сторінці за допомогою DataSourceID, як показано нижче:

<! - В майстер-сторінці SiteTemplate.master ->

...

<Td style = "width: 226px; vertical-align: top;">

<Asp: TreeView ID = "TreeView1" runat = "server" DataSourceID = "SiteMapDataSource1" />

</ Td>

...

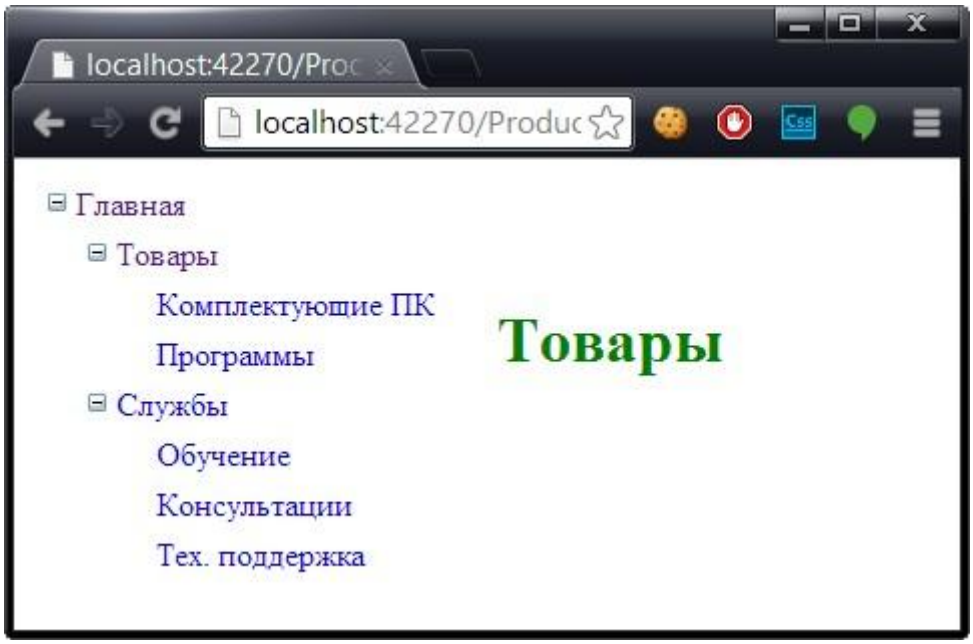

Рис. 2.2 - Приклад зовнішнього вигляду сайту

Або ж скористатися елементом управління Menu:

<! - В майстер-сторінці SiteTemplate.master ->

...

<Td style = "width: 226px; vertical-align: top;">

<asp: Menu ID = "Menu1" runat = "server" DataSourceID = "SiteMapDataSource1" />

</ Td>

...

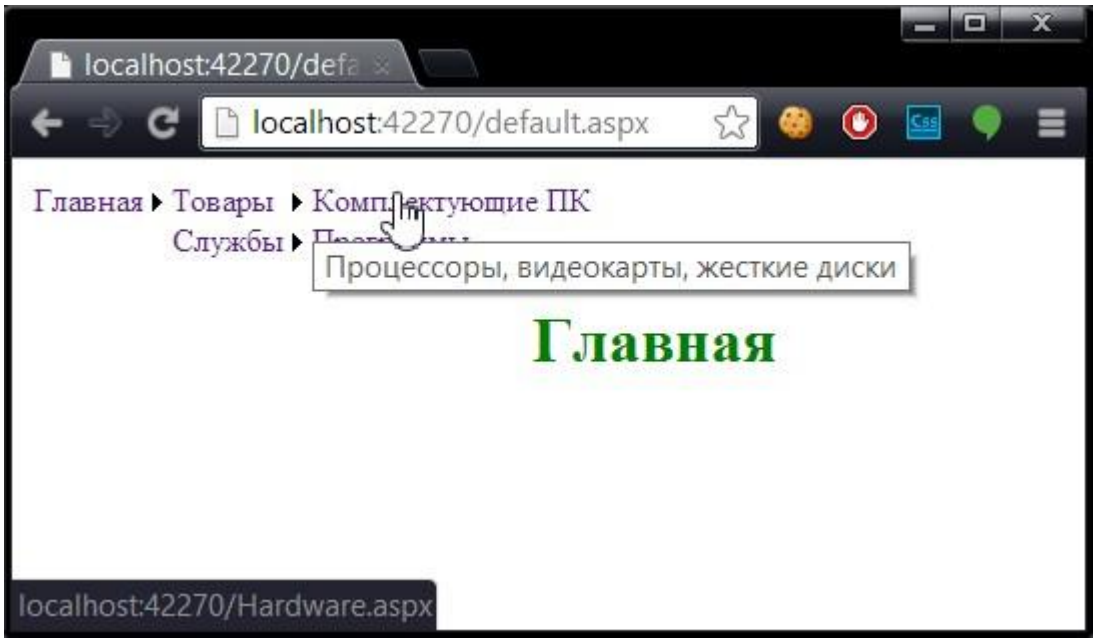

Рис. 2.3 - клікабельне меню сайту

Для настройки зовнішнього вигляду елементів управління і обробки карти сайту можна зробити ще дуже багато. Ці більш складні питання розглядаються в наступних розділах.

2.3 Навігаційні ланцюжка

Насправді ASP.NET визначає три елементи управління навігацією: TreeView, Menu і SiteMapPath. Елемент SiteMapPath забезпечує навігаційну ланцюжок ("хлібні крихти" - breadcrumb) - тобто елемент управління показує поточний стан користувача і дозволяє йому з допомогою посилань переходити вгору по ієрархії на більш високий рівень. На малюнку нижче показаний елемент управління SiteMapPath, коли користувач знаходиться на сторінці Hardware.aspx. За допомогою елемента управління SiteMapPath користувач може повернутися на сторінку Products.aspx або Home.aspx:

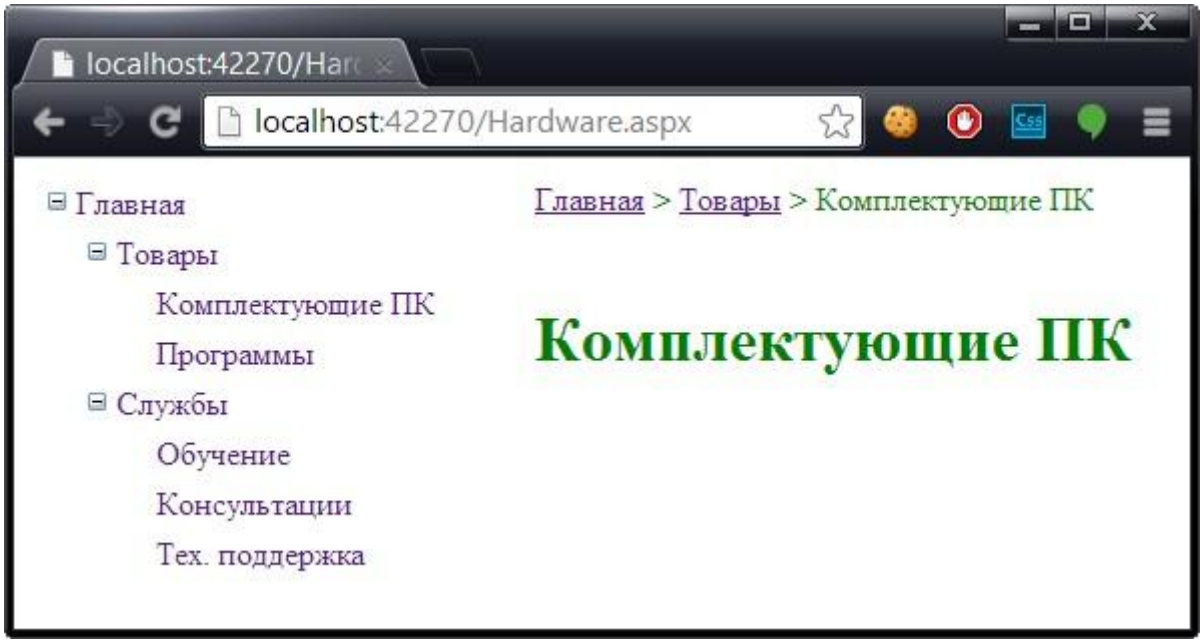

Рис. 2.4 - Шлях сторінки

Елемент управління SiteMapPath володіє невеликим, але дуже важливою відмінністю від інших елементів управління навігації, таких як TreeView і Menu. Елемент SiteMapPath працює безпосередньо з моделлю навігації ASP.NET - іншими словами, йому не потрібно отримувати свої дані через елемент управління SiteMapDataSource. В результаті елемент управління SiteMapPath можна використовувати в сторінках, які не мають елемента управління SiteMapDataSource, а зміна властивостей SiteMapDataSource не чинитиме вплив на SiteMapPath.

Елемент управління SiteMapPath визначається наступним чином:

<Asp: SiteMapPath ID = "SiteMapPath1" runat = "server" />

Як правило, елемент управління SiteMapPath поміщають на майстер-сторінці, щоб він міг відображатися на всіх сторінках вмісту.

Елемент управління SiteMapPath корисний як для миттєвого отримання уявлення про поточну позиції, так і для переміщення вгору по ієрархії. Однак його завжди потрібно використовувати спільно з іншими елементами управління навігацією, які дозволяють користувачеві переміщатися вниз по ієрархії карти сайту.

Елемент управління SiteMapPath можна також налаштовувати самим докладним чином. Деякі з найбільш часто конфігуруються властивостей перераховані в таблиці нижче:

властивість	**опис**
ShowToolTips	Якщо це властивість встановити в false, текст описи й не буде з'являтися при затримці покажчика миші над частиною шляху карти сайту
ParentLevelsDisplayed	Визначає максимальну кількість одночасно відображуваних батьківських рівнів. За замовчуванням ця властивість має значення -1, яке означає, що будуть показані всі рівні
RenderCurrentNodeAsLink	Якщо це властивість встановити в true, частина сторінки, яка вказує поточну сторінку,

	перетворюється в обирану посилання. За замовчуванням ця властивість має значення false, тому що користувач вже знаходиться на поточній сторінці
PathDirection	На вибір доступні два варіанти: RootToCurrent (за замовчуванням) і CurrentToRoot (порядок рівнів в дорозі змінюється на протилежний)
PathSeparator	Показує символи, які будуть поміщені між усіма рівнями в дорозі. За замовчуванням ця властивість встановлено в>. Іншим часто використовуваним роздільником шляху є двокрапка, на моєму сайті це (---).

Для досягнення ще більшого контролю елемент управління SiteMapPath можна конфігурувати за допомогою стилів або навіть перевизначити елементи керування і HTML-вміст за допомогою шаблонів:

стиль	**шаблон**	**застосування**
PathSeparatorStyle	*PathSeparatorTemplate*	Роздільник між усіма вузлами
NodeStyle	*NodeTemplate*	Застосовується до всіх частин шляху за винятком кореневого і поточного

		вузла
CurrentNodeStyle	*CurrentNodeTemplate*	Застосовується до вузла, який представляє поточну сторінку
RootNodeStyle	*RootNodeTemplate*	Застосовується до вузла, який представляє кореневої вузол. Якщо кореневих вузлом є поточний вузол, використовуються стилі або шаблон поточного вузла

Таблиця 2.2 – Шаблони

Наприклад, в наступному елементі управління SiteMapPath використовується зображення стрілки як роздільник і фіксована рядок напівжирного тексту для кореневого вузла. Фінальна частина шляху, яка являє собою поточну сторінку, виділяється курсивом:

<Asp: SiteMapPath ID = "SiteMapPath2" runat = "server">

<PathSeparatorTemplate>

<Asp: Image ID = "Image1" runat = "server" ImageUrl = "~ / arrowright.gif"

GenerateEmptyAlternateText = "true" />

</ PathSeparatorTemplate>

<RootNodeTemplate>

<B style = "border-radius: 4px; border: 2px solid # 5cadff; padding: 5px; color: # 5cadff">

Головна </ b>

</ RootNodeTemplate>

<CurrentNodeTemplate>

<Em>

<Asp: Label ID = "Label1" runat = "server"

Text = '<% # Eval ("title")%>' />

</ Em>

</ CurrentNodeTemplate>

</ Asp: SiteMapPath>

Зверніть увагу на те, як CurrentNodeTemplate використовує вираз прив'язки даних для зв'язування з властивістю title поточного вузла. Аналогічним чином можна також отримати атрибути url і description, які оголошуються в файлі карти сайту.

2.4 Відображення частини карти сайту

У прикладах, розглянутих до цього моменту, елементи управління сторінки в точності відтворювали структуру файлу карти сайту. Однак це підходить не завжди. Наприклад, відображення великої карти сайту може відвернути користувача від тієї частини веб-сайту, яку він переглядає в даний момент часу. Крім того, карта сайту може мати настільки багато рівнів, що все дерево не поміщається повністю на веб-сторінці.

У цій ситуації можна обмежити загальний обсяг інформації і показувати тільки частина карти сайту. У наступних розділах розглядаються різні технології, які можна використовувати.

Пропуск кореневого вузла

Як правило, дерево карти сайту починається з єдиного кореневого вузла. Часто такий спосіб побудови карти сайту виявляється непридатним. Він веде до додаткового рівнем вкладення в структурі карти сайту, в результаті чого вона займає більше місця, і до появи посилання верхнього рівня, яка може виявитися досить марною.

Те, як вузол "Головна" відображається в попередньому прикладі, може бути не зручним. Щоб виправити цю ситуацію, властивість SiteMapDataSource.ShowStartingNode може бути встановлено в false. Якщо ж все-таки бажано, щоб елемент "Головна" відображався, змініть файл карти сайту так, щоб він визначав цей вузол в першій групі сторінок (безпосередньо перед Products). Реальний кореневої вузол не буде доступний широкому, тому URL-адресу йому не потрібно.

Нижче представлений приклад зміненої карти сайту:

<? Xml version = "1.0" encoding = "utf-8"?>

<SiteMap xmlns = "http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">

<SiteMapNode title = "Root" description = "Root">

<SiteMapNode url = "~ / default.aspx" title = "Головна" description = "Головна сторінка сайту" />

<SiteMapNode title = "Товари" description = "Наші товари" url = "~ / Products.aspx">

...

</ SiteMapNode>

<SiteMapNode title = "Служби" description = "Наші служби тех. Підтримки, навчання та ін."

url = "~ / Services.aspx">

...

</ SiteMapNode>

</ SiteMapNode>

</ SiteMap>

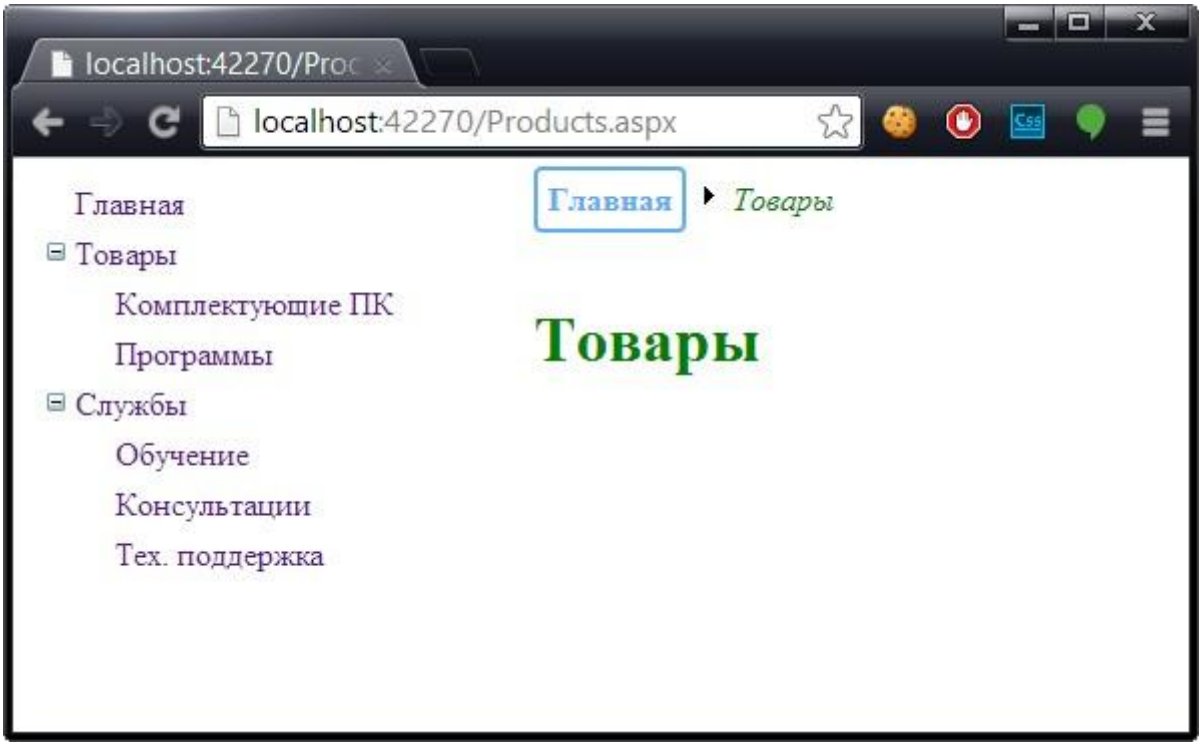

Рис. 2.5 - Навігація

2.5 Вибір поточного вузла в якості початкового

У попередньому прикладі було показано, як можна припустити кореневої вузол. Ще одна наявна можливість - відображення тільки частини

всієї карти сайту, починаючи з поточного вузла. Наприклад, елемент управління, такий як TreeView, можна використовувати для відображення всіх елементів ієрархії, починаючи з поточного вузла. Якщо користувач захоче перейти на рівень вгору, він може використовувати інший елемент керування (наприклад, SiteMapPath).

Для реалізації цього дизайну досить встановити властивість SiteMapDataSource.StartFromCurrentNode в true.

Елемент управління SiteMapPath як і раніше буде показувати всю ієрархію, тому що він не використовує елемент керування SiteMapDataSource. (Таким чином, користувач може клацнути на засланні в елементі управління SiteMapPath, щоб перейти на сторінку вищого рівня.) Однак інші пов'язані елементи управління на кшталт TreeView будуть відображати тільки ті сторінки, які знаходяться нижче поточної, даючи можливість користувачеві переміщатися вниз по ієрархії.

Як і раніше можна використовувати властивість ShowStartingNode, але тепер воно визначає, чи буде відображатися поточний вузол, оскільки він є початковою точкою для дерева навігації. Приклад ситуації, коли властивості StartFromCurrentNode і ShowStartingNode встановлені в true, наведено на малюнку нижче. Поточної є сторінка Products.aspx. Елемент управління SiteMapPath відображає сторінки верхнього рівня, а елемент управління TreeView - вузли, що знаходяться нижче Products.aspx (Hardware.aspx і Software.aspx).

Щоб ця технологія могла працювати, платформа ASP.NET повинна вміти знаходити в файлі Web.sitemap сторінку, що відповідає поточному URL-адресою. В іншому випадку вона не зможе визначити поточну позицію і не надасть потрібну інформацію прив'язаним елементів управління.

Вибір певного вузла в якості початкового

Елемент управління SiteMapDataSource має ще дві властивості, які можуть допомогти в налаштуванні дерева навігації - StartingNodeOffset і StartingNodeUrl.

Властивість StartingNodeUrl простіше для розуміння - воно приймає URL-адресу сайту, який повинен бути першим у дереві. Це значення має в точності збігатися з атрибутом url вузла у файлі Web.sitemap. Наприклад, якщо властивість StartingNodeUrl визначено як "~ / default.aspx", то першим вузлом в дереві буде "Головна", і ви будете бачити тільки ті вузли, які знаходяться під ним.

Властивість StartingNodeUrl особливо корисно, якщо потрібно варіювати між невеликою кількістю різних карт сайту (наприклад, менше десяти). Ідеальне рішення в цьому випадку - визначення кількох файлів карти сайту і прив'язка до потрібного з них. На жаль, за замовчуванням XmlSiteMapProvider підтримує тільки один файл карти сайту, тому знадобиться інший механізм. В даному випадку рішення буде полягати в розподілі різних карт сайту на окремі гілки файлу Web.sitemap.

Наприклад, припустимо, що веб-сайт повинен містити розділи Dealer (Дилер) і розділ Employee (Співробітник). Можна створити дві різні структури і визначити кожну з них в різних гілках одного і того ж файлу, як показано нижче:

```
<? Xml version = "1.0" encoding = "utf-8"?>

<SiteMap xmlns = "http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">

 <SiteMapNode title = "Root" description = "Root">

 <SiteMapNode url = "~ / default.aspx" title = "Дилер" description = "Дилер">

                ...

 </ SiteMapNode>
```

<SiteMapNode url = "~ / default_employee.aspx" title = "Співробітник" description = "Співробітник">

...

</ SiteMapNode>

</ SiteMapNode>

</ SiteMap>

Тепер для прив'язки меню до подання Dealer властивість StartingNodeUrl має бути встановлено в "~ / default.aspx". Це можна зробити програмно або, що більш практично, створивши абсолютно іншу майстер-сторінку і реалізувавши її в усіх сторінках Dealer. У сторінках Employee властивість StartingNodeUrl необхідно встановити в "~ / default_employee.aspx". В результаті будуть відображатися тільки ті сторінки, які знаходяться нижче гілки Employee карти сайту.

Можна зробити ще простіше, за допомогою атрибута siteMapFile розбивши всю карту сайту на окремі файли:

<? Xml version = "1.0" encoding = "utf-8"?>

<SiteMap xmlns = "http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">

<SiteMapNode title = "Root" description = "Root">

<SiteMapNode siteMapFile = "Dealers.sitemap" />

<SiteMapNode siteMapFile = "Employees.sitemap" />

</ SiteMapNode>

</ SiteMap>

Але навіть при використанні цієї технології ви будете обмежені одним деревом карти сайту, яка завжди починається з файлу Web.sitemap. Однак управління картою сайту можна спростити за рахунок винесення деякої частини її вмісту в окремі файли.

Ця технологія істотно обмежена, оскільки елемент управління XmlSiteMapProvider не дозволяє дублювати URL-адреси. Це означає, що повторне використання однієї і тієї ж сторінки більш ніж в одній гілці карти сайту неможливо. Хоча і можна спробувати обійти цю проблему, створюючи різні еквівалентні один одному URL-адреси (наприклад, додаючи в кінець URL додаткові параметри рядка запиту), це може привести до виникнення ще більших проблем. Якщо ці обмеження неприйнятні для конкретного сценарію, то найкращим виходом буде проектування власного постачальника карти сайту.

Властивість SiteMapDataSource.StartingNodeOffset застосовується найбільш часто. Воно приймає ціле число, яке дає елементу управління SiteMapDataSource команду на перехід на певну кількість рівнів вниз по дереву (якщо це число позитивне) або вгору по дереву (якщо число негативне). Розробники часто не беруть до уваги один дуже важливий нюанс: коли елемент управління SiteMapDataSource виконує переміщення вниз по дереву, він рухається у напрямку до поточного вузла. Якщо ж цей вузол вже є поточним або зсув виводить його за межі поточного вузла, SiteMapDataSource не знатиме, куди рухатися, тому в кінцевому підсумку ви отримаєте порожній елемент управління навігацією.

Щоб зрозуміти, як це працює, розглянемо наступний приклад. Припустимо, що в даний момент ми знаходимося на наступній сторінці веб-сайту:

Default> Products> Software> Custom> Contact Us

Якщо SiteMapDataSource починається з вузла Default (за замовчуванням), а властивості StartingNodeOffset присвоєно значення 2, то елемент управління перейде на два рівня вниз і причепиться до дерева, розташованому нижче цього вузла. У нашому прикладі цим вузлом є Software:

Software> Custom> Contact Us

Це означає, що можна буде перейти до будь-якому посиланню в групах Software або Custom, але нікуди більше (по крайней мере, якщо спочатку не перейти на рівень вище або не вибрати інший елемент керування).

Якщо ви спробуєте перейти вниз на занадто багато рівнів - наприклад, коли користувач знаходиться на сторінці другого рівня, а властивості StartingNodeOffset присвоєно значення 3 - елемент SiteMapDataSource не зможе підрахувати кількість рівнів, і пов'язані елементи управління залишаться порожніми.

Ще одна корисна технологія - перехід вгору від поточного вузла. Наприклад, якщо властивість StartFromCurrentNode встановити в true, а властивість StartingNodeOffset в -3, то елемент управління SiteMapDataSource перейде на три рівні вгору, починаючи з поточної сторінки (Contact Us) і виконає прив'язку до наступного дереву:

Products> Software> Custom> Contact Us

Ця технологія кілька корисніша, оскільки вона гарантує, що елементи управління навігацією завжди будуть відображати одне й те саме кількість рівнів. При спробі переступити через кореневий вузол просто відобразиться стільки рівнів, скільки можливо. Наприклад, якщо властивості StartingNodeOffset привласнити значення -3, а користувач в даний момент часу буде перебувати на сторінці другого рівня (такий як Software), прив'язка буде виконана до наступного дереву:

Products> Software

Щоб визначитися з правильною комбінацією параметрів SiteMapDataSource, які бажано використовувати, доведеться трохи поекспериментувати.

StartingNodeOffset і StartFromCurrentNode - спеціалізовані властивості, які ніколи не використовуються на багатьох веб-сайтах. Проте, вони можуть виявитися корисними, якщо доведеться мати справу зі складним деревом карти сайту з безліччю рівнів вкладень. У цьому випадку ці властивості можна використовувати для скорочення кількості одночасно відображаються рівнів. Це спрощує читання і розуміння навігаційних посилань (по крайней мере, вони стануть більш компактними і не будуть займати корисний простір на веб-сторінці). Щоб домогтися цього ж ефекту за допомогою елемента управління SiteMapPath (який не використовує SiteMapDataSource), можна встановити властивість SiteMapPath.ParentLevelsDisplayed.

Об'єкти карти сайту

Для відображення ієрархії навігації можна застосовувати не тільки зв'язування без коду. З інформацією про навігації можна взаємодіяти і програмно. Навігація програмними засобами застосовується в двох випадках:

Для зміни зображення сторінки. Наприклад, можна витягти інформацію поточного вузла і застосувати її для конфігурації таких елементів, як заголовок і заголовок сторінки.

Для реалізації іншої логіки навігації. Наприклад, може вимагатися відображення тільки частини повного списку дочірніх вузлів поточної сторінки в програмі читання новин або створення навігаційних кнопок переходу до попередньої / наступної сторінки.

API-інтерфейс карти сайту дуже простий. Щоб його використовувати, потрібно працювати з двома класами з простору імен System.Web.

Початковою точкою служить клас SiteMap, який надає статичні властивості CurrentNode (вузол карти сайту, що представляє поточну сторінку) і RootNode (кореневої вузол карти сайту). Обидва ці властивості повертають об'єкт SiteMapNode. З його допомогою можна отримувати інформацію з карти сайту, включаючи значення title, description і url. Навігаційні властивості, за допомогою яких можна працювати з пов'язаними вузлами, перераховані в таблиці нижче:

властивість	**опис**
ParentNode	Повертає вузол, розташований на один рівень вище в ієрархії навігації, який містить поточний вузол. Для кореневого вузла це властивість повертає нульову посилання
ChildNodes	Надає колекцію всіх дочірніх вузлів. Перевіряє властивість HasChildNodes для з'ясування, чи існують дочірні вузли
PreviousSibling	Повертає попередній вузол, який знаходиться на тому ж рівні (або посилання null, якщо жодного такого вузла не існує)
NextSibling	Повертає наступний вузол, який знаходиться на тому ж рівні (або посилання null, якщо жодного такого вузла не існує)

Таблиця 2.3 - Навігаційні властивості

РОЗДІЛ 3 РЕАЛІЗАЦІЯ

Веб-сайт побудований на базі Web Forms платформи інтернет-проектів ASP.NET, мова програмування C #. Дизайн сайту створений на основі вільно розповсюджуваних шаблонів. Всі веб сторінки розміщені в кореневому каталозі.

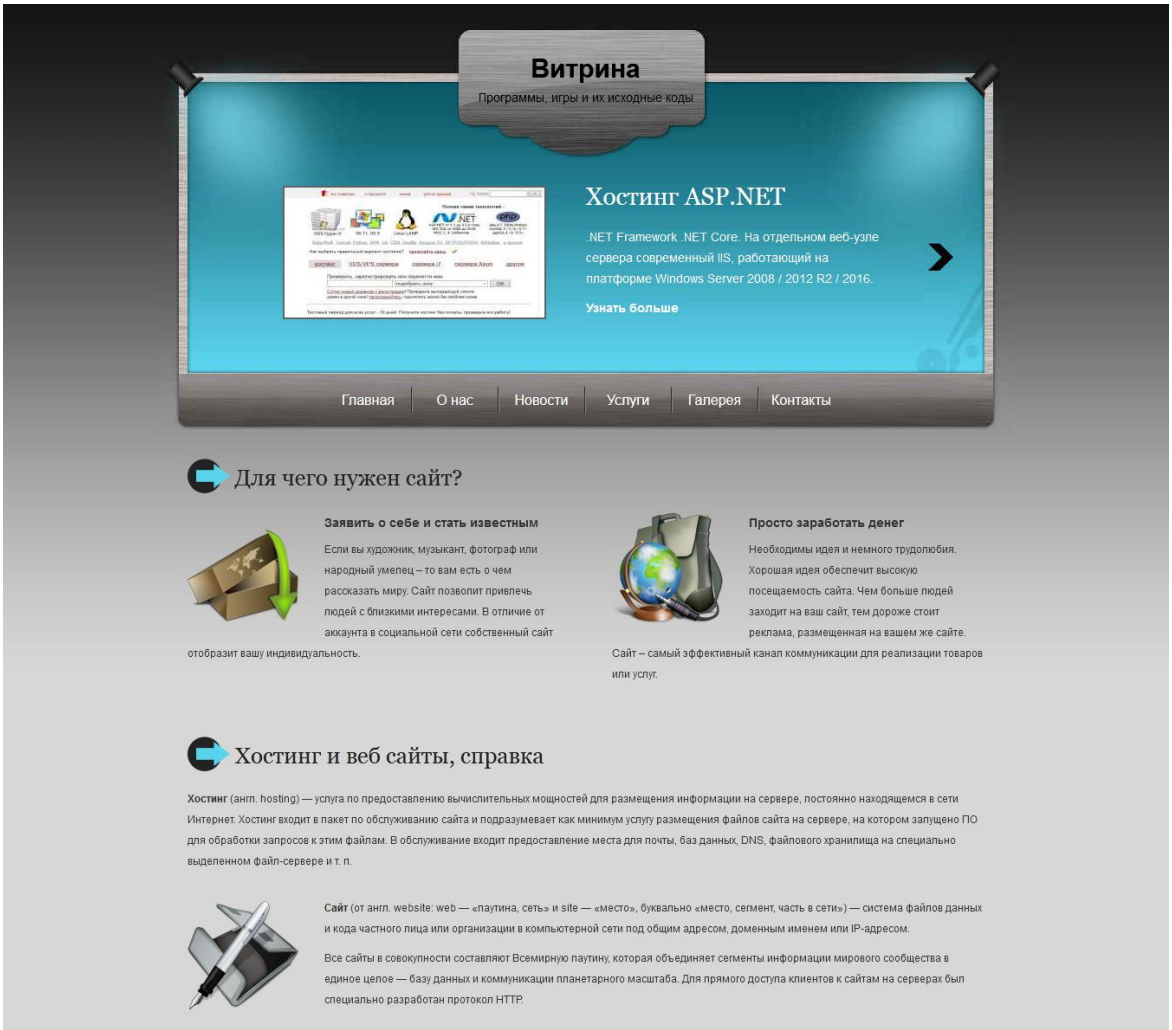

Рис. 3.1 - Головна сторінка сайту

Веб-сайт побудований за допомогою майстер-сторінки і дочірніх сторінок з контентом. Це зроблено для забезпечення одноманітності вигляду сторінок сайту. Майстер-сторінка містить основний дизайн, в дочірніх сторінках тільки контент необхідний для відображення. При завантаженні майстер-сторінка автоматично зливається з запитаної дочірньої сторінкою в єдине ціле.

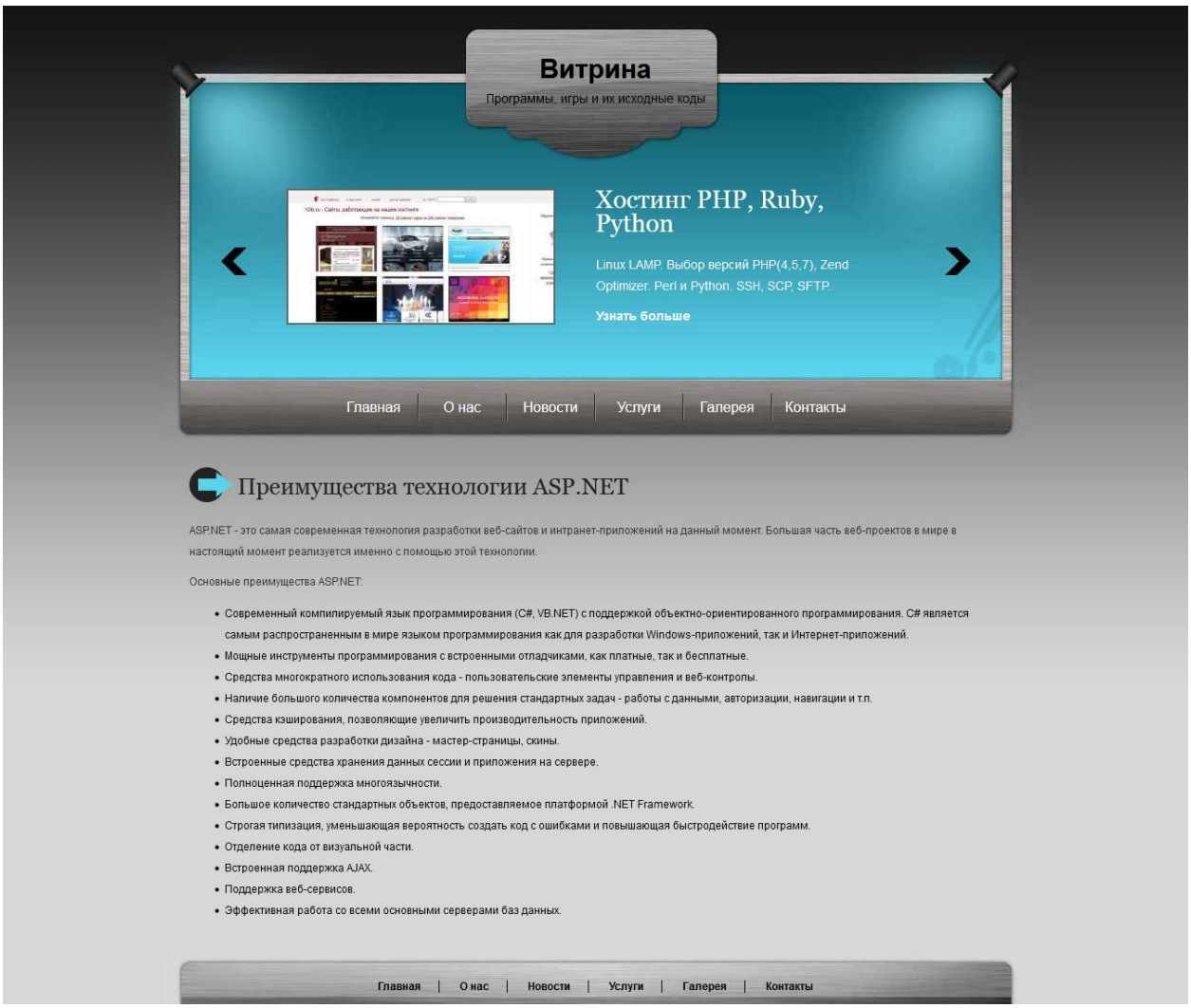

Рис.3.2 - Сторінка «Про нас»

Майстер-сторінка по суті звичайна сторінка ASPX, але з спеціальним розширенням .master. Також, як і будь-яка сторінка ASPX, вона складається з двох сторінок, одна з HTML розміткою (плюс впроваджуваний код), інша з відокремленим програмним кодом на мовах .NET: C # або Visual Basic.

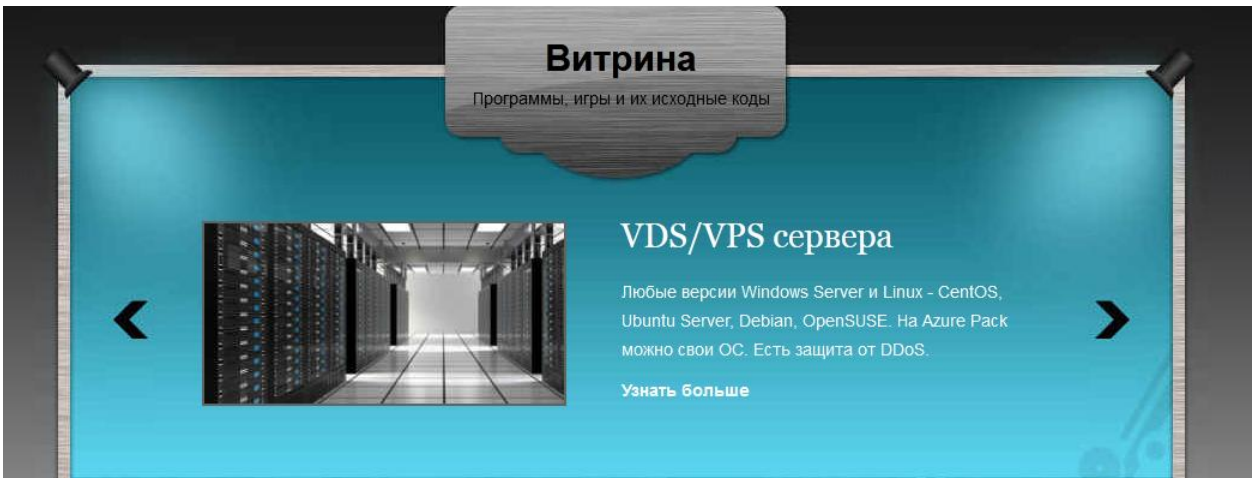

Рис. 3.3 - Галерея пропозицій

Рис. 3.4 - Галерея пропозицій

ВИСНОВОК

У дипломній роботі було зроблено висновок, що навчившись створювати цілісні веб-сторінки, програміст починає замислюватися про загальну картину - тобто про групуванні великого числа веб-сторінок в єдиний логічно пов'язаний веб-сайт.

Однак програміст стикається також з такими завданнями, як забезпечення логічної зв'язності всіх сторінок і спрощення навігації по веб-сайту. У цій частині ми розглянемо питання, які стають актуальними, коли ви перестаєте думати про окремих сторінках і переходите до планування веб-додатки в цілому.

Були виконані такі завдання: а) складання інформаційної моделі;

б) розкриття Web-технології розробки WEB-сайта;

в) створення розробки WEB-сайта.

СПИСОК ЛІТЕРАТУРИ

1. Кириллов В.В. Основы проектирования реляционных баз данных. Учебное пособие. — СПб.: ИТМО, 1994. — 90 с.

2. http://ru.wikipedia.org/. — Вільна енциклопедія.

3. Хоган Б. - HTML5 и CSS3. Веб-разработка по стандартам нового поколения. 2012.

4. Оскерко В.С., Пунчик З.В. Практикум по технологиям баз данных: Учебное пособие. — Минск: БГЭУ, 2004. — 170с.

5. Конспект лекций по дисциплине «Информационные системы в ПК»

6. http://www.lessons-tva.info/. — Безкоштовне дистанційне навчання інформатиці, телекомунікаціям та основам електронного бізнесу.

7. CMS огляд: CMS, движок сайту, система управління сайтом, Mambo, php nuke, netcat, phpbb, invision power board, vbulletin. [Електронний документ] URL http://cmsobzor.ru/news.php

8. Dreamwesver MX 2004 для "чайников". [Текст] Уорнер, Джанни, Гарднер, Сюзанна. Пер. с англ. — М. : Издательский дом "Вильямс", 2004. — 352 с.

9. PHP, MySQL и Dreamweaver MX 2004. Разработка интерактивных Web-сайтов. [Текст] Дронов В. А. — СПб.: БХВ-Петербург, 2005. — 448 с : ил.

ДОДАТОК А

```
<!DOCTYPE html>


<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

   <title>Пример сайта на ASP.NET. Веб формы</title>

   <meta name="keywords" content="пример, веб-сайт, веб-формы,
вебприложение, web forms, ASP.NET" />

   <meta name="description" content="Пример вебсайта на ASP.NET на
бесплатном шаблоне" />

   <link href="/css/style.css" rel="stylesheet" type="text/css" />

   <link href="/css/jquery.ennui.contentslider.css" rel="stylesheet" type="text/css"
/>

   <script language="javascript" type="text/javascript">

      function clearText(field) {

         if (field.defaultValue == field.value) field.value = '';

         else if (field.value == '') field.value = field.defaultValue;

      }

   </script>

   <asp:ContentPlaceHolder ID="head" runat="server">

   </asp:ContentPlaceHolder>
```

```
</head>
<body>
    <form id="form1" runat="server">
    <div id="templatemo_container">

        <div id="templatemo_site_title_wrapper">

            <div id="site_title">
                <h1>
                    <a href="http://www.interestprograms.ru/" target="_parent">Витрина
                <span>Программы, игры и их исходные коды</span></a>
                </h1>
            </div>

        </div>


            <div id="templatemo_banner">

     <div id="one" class="contentslider">
            <div class="cs_wrapper">
```

```
<div class="cs_slider">

<div class="cs_article">
<a href="http://www.1gb.ru/1fd693bdgh" target="_blank">
<img src="images/article01.jpg" alt="Хостинг ASP.NET" />
</a>

<div class="text">
<h2><a href="http://www.1gb.ru/1fd693bdgh"
target="_blank">Хостинг ASP.NET</a></h2>

<p>Выбор  от .NET Framework 1.0  до .NET Framework 4.5. На отдельном веб-узле сервера современный IIS, работающий на платформе Windows Server 2003 / 2008 / 2008 R2 / 2012 R2.
</p>

<a class="readmore" href="http://www.1gb.ru/1fd693bdgh"
target="_blank">Узнать больше</a>
</div>
</div><!-- End cs_article -->

<div class="cs_article">
```

```
<a href="http://www.1gb.ru/1fd693bdgh" target="_blank">

<img src="images/article02.jpg" alt="Хостинг на платформе Microsoft" />

</a>


<div class="text">

<h2> <a href="http://www.1gb.ru/1fd693bdgh" target="_blank">Хостинг 1Gb.ru</a> </h2>


<p>

Профессиональный хостинг на основе самых прогрессивных технологических решений.

Один из лидеров эффективного хостинга на платформе Microsoft.

</p>


<a class="readmore" href="http://www.1gb.ru/1fd693bdgh" target="_blank">Узнать больше</a>

</div>

</div><!-- End cs_article -->


<div class="cs_article">

<a href="http://timeweb.com/ru/?i=3098&a=141" target="_blank">
```


<h2> Хостинг PHP, Perl, Python </h2>

Хостинг для сайтов на PHP, Perl и Python. Хостинг работает на связке вебсерверов Nginx + Apache. PHP представлен версиями 5.2, 5.3, 5.4 и 5.5. Версия MySQL - 5.5.33.

Узнать больше

<!-- End cs_article -->

```
<div class="cs_article">
<a href="http://timeweb.com/ru/?i=3098&a=141" target="_blank">
<img src="images/article04.jpg" alt="Хостинг на вебсерверах Nginx и Apache" />
</a>

<div class="text">
<h2> <a href="http://timeweb.com/ru/?i=3098&a=141" target="_blank">Хостинг TimeWeb</a> </h2>

<p>
Современные и надежные дата-центры. Защита от DDOS-атак. Самые производительные серверы. Недорогие домены. Лучшая сетевая доступность. Круглосуточный мониторинг.
</p>

<a class="readmore" href="http://timeweb.com/ru/?i=3098&a=141" target="_blank">Узнать больше</a>
</div>
</div><!-- End cs_article -->

</div><!-- End cs_slider -->
```

```
        </div><!-- End cs_wrapper -->
    </div><!-- End contentslider -->

    <!-- Site JavaScript -->

<script type="text/javascript" src="js/jquery-1.8.2.min.js"></script>
    <script type="text/javascript" src="js/jquery.easing.1.3.js"></script>
    <script type="text/javascript"
src="js/jquery.ennui.contentslider.js"></script>
    <script type="text/javascript">
        $(function () {
            $('#one').ContentSlider({
                width: '860px',
                height: '210px',
                speed: 800,
                easing: 'easeInOutBack'
            });
        });
            </script>
    <script src="js/jquery.chili-2.2.js" type="text/javascript"></script>
    <script src="js/chili/recipes.js" type="text/javascript"></script>
```

```
</div>
<!-- end of banner -->

<div id="templatemo_menu">
        <ul>
      <li><a href="/">Главная</a></li>
      <li><a href="/about.aspx">О нас</a></li>
      <li><a href="/news.aspx">Новости</a></li>
      <li><a href="/services.aspx">Услуги</a></li>
      <li><a href="/gallery.aspx">Галерея</a></li>
      <li><a href="/contacts.aspx" class="last">Контакты</a></li>
    </ul>
</div>

    <div id="templatemo_content">
      <!-- Начало дочерней страницы -->
      <asp:ContentPlaceHolder ID="Content" runat="server">
```

```
            <% /*место для автоматической вставки кода дочерних страниц*/
%>
        </asp:ContentPlaceHolder>
        <!-- Конец дочерней страницы -->
    </div>


    <div id="templatemo_footer">

        <ul class="footer_menu">
            <li><a href="">Главная</a></li>
            <li><a href="/about.aspx">О нас</a></li>
            <li><a href="/news.aspx">Новости</a></li>
            <li><a href="/services.aspx">Услуги</a></li>
            <li><a href="/gallery.aspx">Галерея</a></li>
            <li class="last_menu"><a href="/contacts.aspx">Контакты</a></li>
        </ul>

        Copyright © 2014 <a href="#">Interestprograms.ru</a>

    </div>
```

```
        <!-- end of footer -->


    </div> <!-- end templatemo_container -->
    </form>
</body>
</html>
<?xml version="1.0" encoding="utf-8" ?>
<data>
  <about>
    <title><![CDATA[Кратко о ASP.NET Web Forms]]></title>
    <content>
    <![CDATA[
    <p>
```

Web Forms являются частью структуры веб-приложений ASP.NET и входят в состав Visual Studio. Это одна из четырех моделей программирования используемых для создания веб-приложений ASP.NET. Веб формы наиболее быстрый способ создания веб сайтов.

```
    </p>


    <p>Веб-формы это веб-страницы которые могут быть написаны с
```

использованием комбинации HTML и серверного кода. Когда пользователь запрашивает страницу, она компилируется и запускается на сервере, и в итоге браузеру отправляется готовый HTML код веб-страницы. Web Forms

предоставляют информацию пользователю в любом браузере или клиентском устройстве.

С помощью Visual Studio, вы можете создать веб-формы ASP.NET. Интегрированная среда разработки позволяет внедрять серверные элементы управления в страницу Web Forms. Вы можете легко установить свойства, методы и события для элементов управления для интерактивного поведения веб-страницы. Для написания кода сервера используются языки .NET, C# или Visual Basic.

]]>

</content>

</about>

<news>

<title><![CDATA[Статистика Интернет]]></title>

<content>

<![CDATA[

861,4 млн – это количество сайтов в интернете по состоянию на 1 января 2014 года.

Сегодня на планете практически 2 млрд пользователей интернета (1,966,514,816 человек), в то время как общее количество населения составляет почти 7 млрд (6,845,609,960 человек)!

По данным опроса Всероссийского центра изучения общественного мнения (ВЦИОМ), проведенного 4-5 октября 2014 г., интернетом пользуются 66% граждан России от 18 лет и старше или 76,3 млн человек. Ежедневно выходят в Сеть – 46% или 53,6 млн взрослых россиян.

]]></content>

</news>

<services>

<title><![CDATA[Процесс создания web-сайта]]></title>

<content>

<![CDATA[

Начинать создание web-сайта на платформе ASP.NET, как и любого другого ПО лучше всего с разработки проекта, который должен включать в себя детальное описание функциональности сайта, его архитектуру и примерный дизайн.

При разработке дизайна, лучше всего использовать Мастер страницы (Master Pages), которые, по сути, являются некоторыми шаблонами страниц сайта. Мастер страницы - одна из самых современных технологий web-программирования, используя их, Вы сможете легко поддерживать единый дизайн сайта. В случае необходимости изменить дизайн будет достаточно

отредактировать Мастер страницы для разделов сайта, все остальные страницы, которых может быть несколько сотен или тысяч, изменять не придётся.

Приступая к реализации функциональности сайта рекомендуется тщательно ознакомиться с классами стандартной библиотеки, особенно теми, которые могут быть полезны в каждом конкретном случае. Следуя данному совету, Вы можете существенно сократить время, затрачиваемое на программирование функциональности web-сайта.

]]>

</content>

</services>

<gallery>

<title><![CDATA[Преимущества технологии ASP.NET]]></title>

<content>

<![CDATA[

ASP.NET - это самая современная технология разработки веб-сайтов и интранет-приложений на данный момент. Большая часть веб-проектов в мире в настоящий момент реализуется именно с помощью этой технологии.

Основные преимущества ASP.NET:

Современный компилируемый язык программирования (C#, VB.NET) с поддержкой объектно-ориентированного программирования. C# является самым распространенным в мире языком программирования как для разработки Windows-приложений, так и Интернет-приложений.

Мощные инструменты программирования с встроенными отладчиками, как платные, так и бесплатные.

Средства многократного использования кода - пользовательские элементы управления и веб-контролы.

Наличие большого количества компонентов для решения стандартных задач - работы с данными, авторизации, навигации и т.п.

Средства кэширования, позволяющие увеличить производительность приложений.

Удобные средства разработки дизайна - мастер-страницы, скины.

Встроенные средства хранения данных сессии и приложения на сервере.

Полноценная поддержка многоязычности.

Большое количество стандартных объектов, предоставляемое платформой .NET Framework.

Строгая типизация, уменьшающая вероятность создать код с ошибками и повышающая быстродействие программ.

Отделение кода от визуальной части.

```
        <li>Встроенная поддержка AJAX.</li>
        <li>Поддержка веб-сервисов.</li>
        <li>Эффективная работа со всеми основными серверами баз данных.</li>
    </ul>
    </p>]]>
      </content>
  </gallery>
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="Content" runat="server">
    <div class="section_w920">

        <h2>Для чего нужен сайт?</h2>

        <div class="section_w430 fl">
            <img class="image_wrapper fl_image" src="images/templatemo_image_01.png" alt="image" />
            <h3>Заявить о себе и стать известным</h3>
            <p>Если вы художник, музыкант, фотограф или народный умелец – то вам есть о чем рассказать миру. Сайт позволит привлечь людей с близкими интересами. В отличие от аккаунта в социальной сети собственный сайт отобразит вашу индивидуальность.</p>
        </div>
```

```
        <div class="section_w430 fr">
            <img class="image_wrapper fl_image"
src="images/templatemo_image_02.png" alt="image" />
            <h3>Просто заработать денег</h3>
            <p>Необходимы идея и немного трудолюбия. Хорошая идея
обеспечит высокую посещаемость сайта. Чем больше людей заходит на ваш
сайт, тем дороже стоит реклама, размещенная на вашем же сайте. Сайт –
самый эффективный канал коммуникации для реализации товаров или
услуг.</p>
        </div>

        <div class="cleaner"></div>
    </div>

    <div class="cleaner_h50"></div>

    <div class="section_w920">
        <h2>Хостинг и веб сайты, справка</h2>
        <p><strong>Хостинг</strong> (англ. hosting) — услуга по
предоставлению вычислительных мощностей для размещения информации
на сервере, постоянно находящемся в сети Интернет. Хостинг входит в пакет
по обслуживанию сайта и подразумевает как минимум услугу размещения
файлов сайта на сервере, на котором запущено ПО для обработки запросов к
```

этим файлам. В обслуживание входит предоставление места для почты, баз данных, DNS, файлового хранилища на специально выделенном файл-сервере и т. п.

Сайт (от англ. website: web — «паутина, сеть» и site — «место», буквально «место, сегмент, часть в сети») — система файлов данных и кода частного лица или организации в компьютерной сети под общим адресом, доменным именем или IP-адресом.

Все сайты в совокупности составляют Всемирную паутину, которая объединяет сегменты информации мирового сообщества в единое целое — базу данных и коммуникации планетарного масштаба. Для прямого доступа клиентов к сайтам на серверах был специально разработан протокол HTTP.

</asp:Content>

```
body {
        margin: 0;
        padding: 0;
        line-height: 2em;
        font-family: Arial, Helvetica, sans-serif;
        font-size: 12px;
        color: #000000;
    background: #d7d7d7 url(/images/templatemo_main_bg.jpg) top repeat-x;
}

a:link, a:visited { color: #000000; font-size:12px; text-decoration: none; font-weight: normal; }
a:active, a:hover { color: #CC0000; font-size:12px; text-decoration: underline; }

p { margin: 0px; padding: 0px; color: #333333; }

img { margin: 0px; padding: 0px; border: none; }

.cleaner { clear: both; width: 100%; height: 0px; font-size: 0px;  }

.cleaner_h10 { clear: both; width:100%; height: 10px; }
```

```
.cleaner_h20 { clear: both; width:100%; height: 20px; }

.cleaner_h30 { clear: both; width:100%; height: 30px; }

.cleaner_h40 { clear: both; width:100%; height: 40px; }

.cleaner_h50 { clear: both; width:100%; height: 50px; }

.cleaner_h60 { clear: both; width:100%; height: 60px; }


.fl { float: left; }

.fr { float: right; }


.m_right { margin-right: 30px; }


.button_01 a {

        display: block;

        width: 106px;

        height: 30px;

        padding: 6px 0 0 0;

        background: url(/images/templatemo_button_01.png) no-repeat;


        color: #191717;

        font-size: 14px;

        font-weight: bold;
```

```
        text-align: center;
        text-decoration: none;
}

h1 {
        margin: 0px;
        padding: 2px 0;
        font-size: 30px;
        font-weight: bold;
}

h2 {
        margin: 0px;
        padding: 0px;
        font-size: 20px;
        font-weight: bold;
}

h3 {
        margin: 0 0 5px 0;
        padding: 2px 0;
```

```
        font-size: 14px;
        font-weight: bold;
        color: #333;
}

h4 {
        margin: 0px;
        padding: 0px;
        font-size: 14px;
        font-weight: bold;
}

.image_wrapper {
        margin-top: 3px;
        margin-bottom: 5px;
}

.fl_image {
        float: left;
        margin-right: 30px
}
```

```
.fr_image {
	float: right;
	margin-left: 20px
}



#templatemo_container {
	width: 980px;
	margin: 0 auto;
}

/* site title */
#templatemo_site_title_wrapper {
	height: 180px;
	background: url(/images/templatemo_site_title_bg.jpg) center top no-repeat;
}

#templatemo_site_title_wrapper #site_title {
	width: 285px;
	padding: 60px 0 0 0;
```

```
        margin: 0 auto;
        text-align: center;
}

#site_title h1 a {
        margin: 0px;
        padding: 0px;
        font-size: 30px;
        color: #000000;
        font-weight: bold;
        text-decoration: none;
}

#site_title h1 a:hover {
        font-weight: bold;
        text-decoration: none;
}

#site_title h1 a span {
        display: block;
        margin-top: 10px;
```

```
        font-size: 14px;
        font-weight: normal;
        color: #000000;
}
/* end of site title */

/* banner */
#templatemo_banner {
        clear: both;
        width: 980px;
        height: 240px;
        background: url(/images/templatemo_banner_bg.jpg) center no-repeat;
}
/* end of banner */

/* menu */
#templatemo_menu {
        width: 980px;
        height: 75px;
        background: url(/images/templatemo_menu_bg.jpg) no-repeat;
}
```

```
#templatemo_menu ul {
	width: 600px;
	padding: 22px 0 0 0;
	margin: 0 auto;
	list-style: none;
}

#templatemo_menu ul li {
	padding: 0;
	margin: 0px;
	display: inline;
}

#templatemo_menu ul li a {
	float: left;
	display: block;
	width: 98px;
	padding: 3px 2px 3px 0;
	margin: 0;
	font-size: 16px;
```

```
        text-align: center;
        text-decoration: none;
        color: #c5c3c2;
        background: url(/images/templatemo_menu_divider.jpg) right repeat-y;
        outline: none;
    color: #ffffff;
}

#templatemo_menu li a:hover{
        text-decoration: underline;
}

#templatemo_menu li .last {
        background: none;
}
/* end of menu */

/* content */
#templatemo_content {
        clear: both;
        padding: 30px;
```

```
    min-height:200px;
}

#templatemo_content a{
        font-style: italic;
}

#templatemo_content p {
        margin-bottom: 10px;
}

#templatemo_content h2 {
        height: 30px;
        font-family: Georgia, "Times New Roman", Times, serif;
        padding: 10px 0 0 55px;
        margin: 0 0 20px 0;
        font-size: 26px;
        color: #232323;
        font-weight: normal;
        background:url(/images/templatemo_header_bg.png) left center no-repeat;
}
```

```
.section_w920 {
	clear: both;
	width: 920px;
}

.section_w430 {
	width: 430px;
}
/* end of content */

/* footer */
#templatemo_footer {
	clear: both;
	width: 860px;
	height: 70px;
	padding: 20px 60px;
	text-align: center;
	background: url(/images/templatemo_footer_bg.jpg) no-repeat;
}
```

```
#templatemo_footer a {
	font-weight: bold;
	color: #000000;
}

#templatemo_footer .footer_menu {
	margin: 0 0 5px 0;
	padding: 0px;
	list-style: none;
}

.footer_menu li {
	margin: 0px;
	padding: 0 20px;
	display: inline;
	border-right: 1px solid #000000;
}

.footer_menu .last_menu {
	border: none;
}
```

```
/* end of footer */


.span-bold{
    font-size:24px;
    font-weight:bold;
```