С. М. Оніщенко, В. М. Франчук
НПУ імені М. П. Драгоманова

Створення Web-додатків в середовищі Delphi (Kylix)

Основні поняття

World Wide Web є однією з найновіших послуг Internet: WWW з'явилася в 1990 році в європейському дослідницькому центрі CERN (*англ.* "The European Laboratory for Particle Physics" – Європейський центр високих енергій; *франц.* Conseil Europeen pour la Recherche Nucleaire – Європейська організація з ядерних досліджень), а в 1992 почалося практичне застосування цієї технології за межами CERN. З кінця 1993 року почалося масове захоплення WWW, що призвело до того, що сьогодні цей вид інформаційного сервісу Internet є найбільш популярним, найбільш динамічно розвивається і фактично визначає сучасний стан всесвітньої мережі.

На WWW-серверах можна знайти різноманітнівідомості: інформаційні системи університетів і наукових організацій; юридичні довідникові системи; рекламу комерційних фірм; електронні версії суспільно-політичних і спеціалізованих друкованих видань і багато іншого.

Розробляючи інформаційні системи на основі WWW-серверів, їх автори ставлять перед собою різні задачі і, як наслідок, з'являються системи різної якості, рівня складності, з різними функціональними засобами і різною швидкістю поповнення і поновлення даних.

Сучасний Web-дизайн передбачає перш за все зручні й інтуїтивно зрозумілі засоби користувача для пошуку даних, їх представлення відповідно сучасним вимогам раціонального співвідношення графічних зображень та текстових повідомлень, кольорових гам.

До головних характеристик Web-сайтів відносять швидкість пошуку даних, якість графічних зображень, вірогідність та відповідність сучасному стану подій мультимедіа повідомлень. А також такі послуги, як форум, гостьова книга, чат, конференція, які є додатковими засобами пошуку даних, спілкування з іншими користувачами сайту, обміну даними.

Основу WWW складають Web-вузли-комп'ютери, на яких виконується спеціальна програма – Web-сервер, яка очікує запит клієнта на документ. Документи, зазвичай, зберігаються на Web-вузлі у форматі HTML. Клієнтом Web-сервера є програма-браузер, яка виконується на віддаленому комп'ютері і здійснює запит до Web-сервера, приймає документ і відображує його на екрані.

Абревіатура CGI (Common Gateway Interface) позначає частину Web-сервера, яка може взаємодіяти з іншими програмами, що виконуються на цьому ж Web-вузлі, і в цьому розумінні є шлюзом (gateway – шлюз) для передавання даних, отриманих від клієнта, програмам опрацювання, таким як СУБД, електронні таблиці і інші. До CGI входять також загальне середовище (набір змінних) і протоколи для взаємодії з цими програмами.

Загальна схема роботи CGI складається з наступних етапів.

- Отримання Web-сервером даних від клієнта-браузера. Для передавання даних Web-серверу в мові HTML є засіб – форма. Форма задається в HTML-документі за допомогою тегів <FORM>…</FORM> і складається з набору полів введення, які відображаються браузером у вигляді графічних елементів управління: селекторних кнопок, рядків введення тексту, кнопок управління і т.д.
- Аналіз і опрацювання отриманих даних. Дані, отримані з HTML-форм, передаються для опрацювання CGI-програмі. Вони не завжди можуть бути опрацьовані CGI-програмою самостійно. Наприклад, вони можуть містити запит до деякої бази даних, яку CGI-програма не опрацьовує. В цьому випадку CGI-програма на основі отриманих даних формує запит до відповідної програми, яка виконується на цьому ж комп'ютері.
- Створення нового HTML-документа і пересилання його броузеру. Після опрацювання отриманих даних CGI-програма створює динамічний або віртуальний HTML-документ, або формує посилання на уже існуючий документ і передає результат броузеру.

HTML-форми

HTML-форми призначені для пересилання даних від віддалених користувачів Web-серверу. За їх допомогою можна організувати простий обмін повідомленнями між користувачами і сервером, наприклад, реєстрацію користувача на сервері або вибір потрібного документа з подано списку. Форми підтримуються всіма популярними браузерами.

Тег <FORM> має параметри ACTION, METHOD і ENCTYPE. В загальному вигляді форма задається наступним чином:

<FORM ACTION="URL" METHOD=*метод передавання* ENCTYPE=*MIME-тип*>
вміст форми
</FORM>

Параметр ACTION є єдиним обов'язковим. Його значенням є URL-адреса CGI-програми, за якою будуть опрацьовуватися дані, отримані з даної форми.

Параметр METHOD визначає метод пересилання даних форми від браузера до Web-сервера. Він може набувати одного з двох значень – GET (за замовчуванням) і POST.

Взаємодія між клієнтом-браузером і Web-сервером здійснюється за правилами, що задані протоколом HTTP, і складається із запитів клієнта і відповідей сервера. Запит поділяється на три частини. В першому рядку запиту міститься HTTP-команда, що називається методом, URL-адреса запитуваного файла і номер версії протоколу HTTP. Друга частина – заголовок запиту. Третя частина – тіло запиту, власне дані, що надсилаються до сервера. Метод повідомляє сервер про мету запиту. В протоколі HTTP визначені кілька методів. Для передавання даних в CGI-програму використовуються два методи: GET і POST.

При використанні методу GET дані форми пересилаються у складі URL-запиту, до якого приєднуються після символу "?" у вигляді сукупності пар *змінна=значення*, відокремлених символом "&". В цьому випадку перший рядок запиту може мати наступний вигляд:

GET /cgi-bin/cgi-program.pl ? name=Ivan&surname=Petrov HTTP/1.1

URL запиту Дані форми

Після виділення даних з URL сервер надає їх змінній середовища QUERY_STRING, яка може бути використана CGI-програмою.

При використанні методу POST дані форми пересилаються Web-серверу в тілі запиту, а потім передаються сервером в CGI-програму через потік стандартного введення.

Значення параметра ENCTYPE є медіа-тип, що визначає формат кодування даних при передаванні від броузера до сервера. Браузер кодує дані для того, щоб уникнути їх спотворення в процесі передавання.

Є два значення цього параметра:

- *application/x-www-form-urlencoded,* визначає стандартний метод кодування і використовується за замовчуванням;

- *multipart/form-data*, додатковий метод.

Другий метод потрібен тільки у випадку , коли до вмісту форми приєднується локальний файл, обраний за допомогою елемента форми <INPUT TYPE=FILE>. В інших випадках слід використовувати метод кодування за замовчуванням.

Схема кодування *application/x-www-form-urlencoded* однакова для обох методів пересилання (GET і POST) і полягає в наступному.

Для кожного елемента форми, що має ім'я, вказане через параметр NAME, формується пара "*name=value*", де *value* – значення елемента, введене користувачем або вказане за замовчуванням. Якщо значення відсутнє, відповідна пара має вигляд "*name=*". Для радіокнопок і перемикачів використовуються значення тільки обраних елементів. Якщо елемент обрано, а значення параметра VALUE не визначено, за замовчуванням використовується значення "ON".

Всі пари записуються в рядок і відокремлюються символом &. Імена і значення є звичайним текстом, тому вони можуть містити символи, які недопустимі у складі URL (за методом GET пересилаються дані як частина URL). Такі символи замінюються послідовністю, що складається з символа % і їх шіснадцяткового ASCII-коду. Сивмол пропуск може замінюватися не тільки кодом %20, а й знаком + (плюс). Ознака кінця рядка замінюється кодом %0D%0A. Таке кодування називається URL-кодуванням.

Закодовані дані пересилаються серверу одним з методів GET або POST. Основна відмінність полягає в тому, як саме за методом передаються дані CGI-програмі.

При використанні метода GET дані форми пересилаються серверу у складі URL-запиту, до якого додаються після символа "?". Тіло запиту в цьому випадку є порожнім.

Частина URL після символа "?" називається рядком запиту. Коли Web-сервер отримує запит, змінній середовища QUERY_STRING надається значення рядка запиту і завантажується CGI-програма, вказана в першій частині URL до символа "?". CGI-програма звертається до змінної середовища QUERY_STRING для опрацювання закодованих у ній даних.

За методом GET можна передавати дані CGI-програмі взагалі без використання форм. Дані можна передавати за допомогою гіперпосилання з HTML-документа. Такі дані є статичними, а в формі дані можна змінювати.

Рядок запиту – не єдиний спосіб передавання даних через URL. Іншим способом є додаткові дані про шлях (extra path information), що є частиною URL, розміщеною після імені CGI-програми. Сервером виділяється ця частина і зберігається в змінній середовища PATH_INFO. CGI-програма може потім використовувати цю змінну для отримання даних.

При використанні метода POST дані форми пересилаються серверу в тілі запиту.

Методи GET і POST доцільно використовувати з урахуванням їх особливостей. Метод GET ефективний при пересиланні даних форм, які складаються з невеликої кількості коротких полів. При пересиланні великого об'єму даних слід користуватися методом POST, оскільки браузер або сервер можуть накладати обмеження на розмір даних, які передаються в складі URL. Метод POST є надійнішим при пересиланні конфіденційних даних.

CGI-сценарії

Призначення CGI-програми полягає в створенні нового HTML-документа, використовуючи дані запиту, і передаванні його клієнту. Якщо такий документ уже існує , то передати посилання на нього. Яку мову можна використати для написання CGI-програми? Сам інтерфейс не накладає обмежень на вибір мови програмування. Знаючи, яка задача розв'язується CGI-програмою і яким чином вона отримує вхідні дані, можна назвати властивості, якими має володіти мова CGI-програмування.

- Засоби опрацювання тексту. Потрібні для декодування вхідних даних у вигляді рядка, що складається з окремих полів, відокремлених символами-обмежувачами.
- Засоби доступу до змінних середовища. Необхідні, оскільки за допомогою змінних середовища дані подаються на вхід CGI-програмі.

– Засоби взаємодії з іншими програмами. Необхідні для звернення до СУБД, програм опрацювання графіки і інших спеціальних програм.

Вибір мови залежить і від операційної системи Web-сервера. Більша частина серверів призначена для роботи під управлінням операційної системи UNIX. В цій операційній системі поширено використання скриптів – текстових командних файлів, що є програмами, які складаються із звернень до команд операційної системи і управляючих конструкцій мови shell – командної оболонки мови UNIX. Програма shell є інтерпретатором команд операційної системи, і також має вбудовані засоби, характерні для мов програмування: рядкові змінні і управляючі конструкції – оператори повторення, переходу, умовні оператори і т.д. Shell виконує командні файли як інтерпретатор, тобто читає з файла і виконує команди одна за одною з урахуванням управляючих конструкцій, не перетворюючи текст у виконуваний двійковий код. За допомогою скриптів здійснюється значна частина роботи з конфігурування ОС. Очевидно, з розповсюдженням командних процедур в UNIX і пов'язана поява в цій операційній системі ряду мов інтерпретацій, призначених для створення сценаріїв: Perl, Python, Tcl. Всі вони можуть використовуватися для створення CGI сценаріїв, як і традиційні мови програмування C/C++, Object Pascal. Найпоширенішим стала мова Perl (Practical Extraction and Report Language).

Змінні середовища CGI

Відповідно до методу дані форми передаються в CGI- програму або через стандартний потік введення (POST), або через змінну середовища QUERY_STRING (GET). Крім цих даних CGI- програмі доступні і інші дані, отримані від клієнта або надані Web-сервером. Ці дані зберігаються в змінних середовища. В таблиці подано змінні, які використовуються в CGI.

Змінні середовища	***Опис***
GATEWAY_INTERFACE	Версія CGI, яка використовується на сервері
SERVER_NAME	Доменне ім'я або IP-адреса сервера
SERVER_SOFTWARE	Ім'я і версія програми-сервера, яка відповідає на запит клієнта (наприклад, Apache 1.3)
SERVER_PROTOCOL	Ім'я і версія протоколу даних, який був використаний для запиту (наприклад, HTTP 1.0)
SERVER_PORT	Номер порта комп'ютера, на якому працює сервер (за замовчуванням – 80)
REQUEST_METHOD	Метод, який використовується для подання запиту (GET, POST)
PATH_INFO	Додаткові дані про шлях
PATH_TRANSLATED	Ті ж дані, що і в змінній PATH_INFO з префіксом, що вказаує шлях до кореневого каталогу дерева Web-документів
SCRIPT_NAME	Відносне маршрутне ім'я CGI-сценарія (наприклад, /cgi-bin/program.pl)
DOCUMENT_ROOT	Кореневий каталог дерева Web-документів
QUERY_STRING	Рядок запиту – дані, передані у складі URL запиту після символа "?"
REMOTE_HOST	Ім'я віддаленої машини, з якої зроблено запит
REMOTE_ADDR	IP-адреса віддаленої машини, з якої зроблено запит
REMOTE_USER	Ідентифікаційне ім'я користувача, який посилає запит
CONTENT_TYPE	Медіа-тип даних запиту, наприклад, "text/html"
CONTENT_LENGTH	Кількість байт в тілі запиту, переданих в CGI-програму через стандартний потік введення
HTTP_HOST	Хост-ім'я комп'ютера, на якому працює сервер
HTTP_RFOM	Адреса електронної пошти користувача, який відправив запит
HTTP_ACCEPT	Список медіа-типів, які може приймати клієнт
HTTP_USER_AGENT	Браузер, яким клієнт користується для відправлення запиту
HTTP_REFERER	URL-адреса документа, на який клієнт вказував перед зверненням до CGI-програми

Опрацювання даних форми

Дані форми надходять до CGI-програми в закодованому вигляді, тому першим кроком опрацювання за CGI-сценарієм має бути виконано декодування отриманих даних. При пересиланні даних за методом GET дані форми надаються змінній середовища QUERY_STRING, при пересиланні за методом POST – передаються в програму через стандартний потік введення і теж можуть бути збережені в деякій внутрішній змінній. Отже, декодування даних зводиться до наступної послідовності дій з текстовим рядком:

– заміна кожної групи %hh, яка складається з шістнадцяткового ASCII-коду hh з префіксом %, на відповідний ASCII-символ;
– заміна символів "+" пропусками;
– виділення окремих пар *ім'я=значення*, відокремлених символом "&";
– виділення з кожної пари *ім'я=значення* імені і значення відповідного поля форми.

Після виділення і декодування даних можна починати їх опрацювання.

CGI-сценарії на Object Pascal

Середовище розробки Delphi, в якому використовується мова програмування Object Pascal, має зручний інтерфейс для візуального програмування і широкі можливості для написання програм (додатків). Мова програмування Object Pascal задовільняє всім вимогам CGI-програмування.

Створення найпростішого Web-додатку на Delphi (Kylix), що запускається на стороні сервера і "взаємодіє" з користувачем (при необхідності) через броузер, нічим не відрізняється від створення консольної програми (для DOS).

Розглянемо приклад створення програми, за якою у вікні браузера з'являється повідомлення-привітання.

Для цього потрібно відкрити у Delphi (Kylix) новий консольний проект.

```
program primer;
  {$APPTYPE CONSOLE}
 uses SysUtils;

 begin
  // Insert user code here
 end.
```

Виведення коду HTML у вікні браузера здійснюється за допомогою команди **writeln**. Наступний текст потрібно вставити між службовими словами **begin end**.

```
 writeln ('content-type: text/html');
 writeln;
 writeln ('<html>');
 writeln ('<head>');
 writeln ('<meta HTTP-EQUI="Content-Type Content="text-html"; charset=windows-1251">');
 writeln ('<title>Delphi best facility for making Web-publications!</title>');
 writeln ('</head>');
 writeln ('<body bgcolor="white">');
 writeln ('Hello, world!');
 writeln ('</body>');
 writeln ('</html>');
```

Важливо звернути увагу на рядок

```
writeln ('content-type: text/html');
```

що для браузера визначає опис наступного вмісту, а саме коду HTML. Після виведення тексту content-type: text/html, потрібно вивести порожній рядок, інакше браузер може видати повідомлення про помилку.

Введення програми закінчено, залишилося виконати її компіляцію.

Для перевірки правильності виконання програми потрібен Web-сервер. Можна використати будь-який, навіть стандартний домашній Web-сервер від Microsoft з підтримкою CGI. Програму потрібно розмістити в папці публікацій сервера для CGI скриптів (за звичай це – C:\Inetpub\Script – в WINDOWS або /var/www/cgi-bin – в UNIX) і завантажити сам сервер.

В браузері потрібно перейти за адресою http://localhost/cgi-bin/primer.exe. У вікні браузера буде виведено результат виконання програми – повідомлення Hello, world! Найпростіший Web-додаток на Delphi (Kylix) готовий. Ця програма демонструє спосіб виведення даних у вікні браузера.

Розглянемо питання передавання даних у програму. Для консольної програми передавання даних як параметрів здійснюється за допомогою командного рядка, коли після назви програми через пропуск вводяться потрібні значення. Для користувача браузера як командний рядок використовується рядок адреси у браузері.

Приклад. Створити програму, за якою у вікні браузера виводятся дані про системний час у залежності від параметрів, введених в рядку адреси браузера. Передані з командного рядка дані в консольній програмі можуть бути опрацьовані за допомогою функцій ParamCount і ParamStr.

Створити новий консольний проект. Ввести текст і здійснити компіляцію.

```
program CgiDate;
{$APPTYPE CONSOLE}
uses SysUtils;
begin
writeln ('content-type: text/html');
writeln;
writeln ('<html><head>');
writeln ('<title>cgidate</title>');
writeln ('</head><body>>');
writeln ('<h1>Приклад передавання параметрів</h1>');
writeln ('<hr>');
writeln ('<hr>')
if ParamCount >0 then
begin
if ParamStr (1) = 'date' then
writeln (FormatDateTime('"Сьогодні " dddd, mmmm d, yyyy', Now))
else if ParamStr (1) = 'time' then
```

```
writeln (FormatDateTime('"Час" hh:mm:ss AM/PM', Now))
else if ParamStr (1) = 'both' then
writeln (FormatDateTime('"Сьогодні " dddd, mmmm d, yyyy,'
+ '"<p> і час" hh:mm:ss AM/PM', Now))
else
writeln ('Помилка! Неправильний параметр: ' + ParamStr (1) + '.')
end
else
writeln ('Параметр відсутній.');
writeln ('</body></html>');
end.
```

Якщо в рядку адреси браузера ввести

http://localhost/cgi-bin/cgidate.exe?time

у вікні браузера, буде виведено поточний час, а якщо

http://localhost/cgidate.exe?date

- відповідно дату, а при введенні параметра both - буде виведено дату і час.

У випадку якщо жоден із параметрів передано не буде або буде передано з помилками - виведеться про це повідомлення.

Дані адреси і параметри можна безпосередньо вказати в коді HTML і генерувати необхідні зміни на сторінці або інші сторінки, здійснюючи перехід за відповідними посиланнями.

Як правило, якщо деякі дані передаються від користувача Web-додатку, то звичайно для цього використовують форми, а не рядок адреси.

Приклад. Створити програму, за якою опрацьовуються дані, передані через форму.

Для початку потрібно створити сам код HTML, у якому була б наявна форма з полями введення, кнопкою відправлення й інших необхідних атрибутів.

Потім необхідно створити додаток, який адекватно зміг би сприйняти всі ці дані з боку користувача.

Нижче наведено текст програми, за якою виводиться список деяких змінних оточення і їхні значення.

```
program CgiVars;
{$APPTYPE CONSOLE}
uses Windows;
const
VarList: array [1..17] string [30] =
('SERVER_NAME', 'SERVER_PROTOCOL'
'SERVER_PORT', 'SERVER_SOFTWARE'
'GATEWAY_INTERFACE', 'REQUEST_METHOD'
'PATH_TRANSLATED', 'HTTP_REFERER'
'SCRIPT_NAME', 'PATH_INFO'
'QUERY_STRING', 'HTTP_ACCEPT'
'REMOTE_HOST', 'REMOTE_USER'
'REMOTE_ADDR', 'REMOTE_IDENT'
'HTTP_USER_AGENT');
var
I: Integer;
ReqVar: string;
VarValue: array [0..200] Char;
begin
writeln('Content type: text/html');
writeln;
writeln('<html><head>');
writeln('<title>CGI Variables</title>');
writeln('</head><body>');
writeln('<h1>CGI Variables</h1>');
writeln('<hr><PRE>');
for I := Low (VarList) to High (VarList) do
begin
ReqVar := VarList[I];
if (GetEnvironmentVariable (PChar(ReqVar)
VarValue, 200) > 0) then
else
VarValue := '';
writeln (VarList[I]+ ' = ' + VarValue);
end;
writeln('</pre></body></html>');
```

end.

Для створення повномасштабних додатків для Інтернету в Delphi існує спеціальний "помічник" - Web Server Application. З його допомогою можна створити додаток, що генерує динамічні Web-сторінки, створені на CGI, NSAPI чи ISAPI. Єдине обмеження, що накладається безпосередньо на Web-сервер, – він має працювати на базі Windows.

Однією з головних переваг створення подібних додатків саме в середовищі Delphi (Kylix) є те, що потрібно працювати з візуальними компонентами - це значно простіше, ніж створення додатків в інших середовищах - можливість помилки у великих проектах, де використовується візуальне проектування менше, ніж у тих, де все описується винятково кодом. Крім того, використовуючи засоби створення Web-додатків, можна імпортувати вже існуючі додатки в інтернет-середовище.

Приклади використання методів POST і GET.

Метод POST.

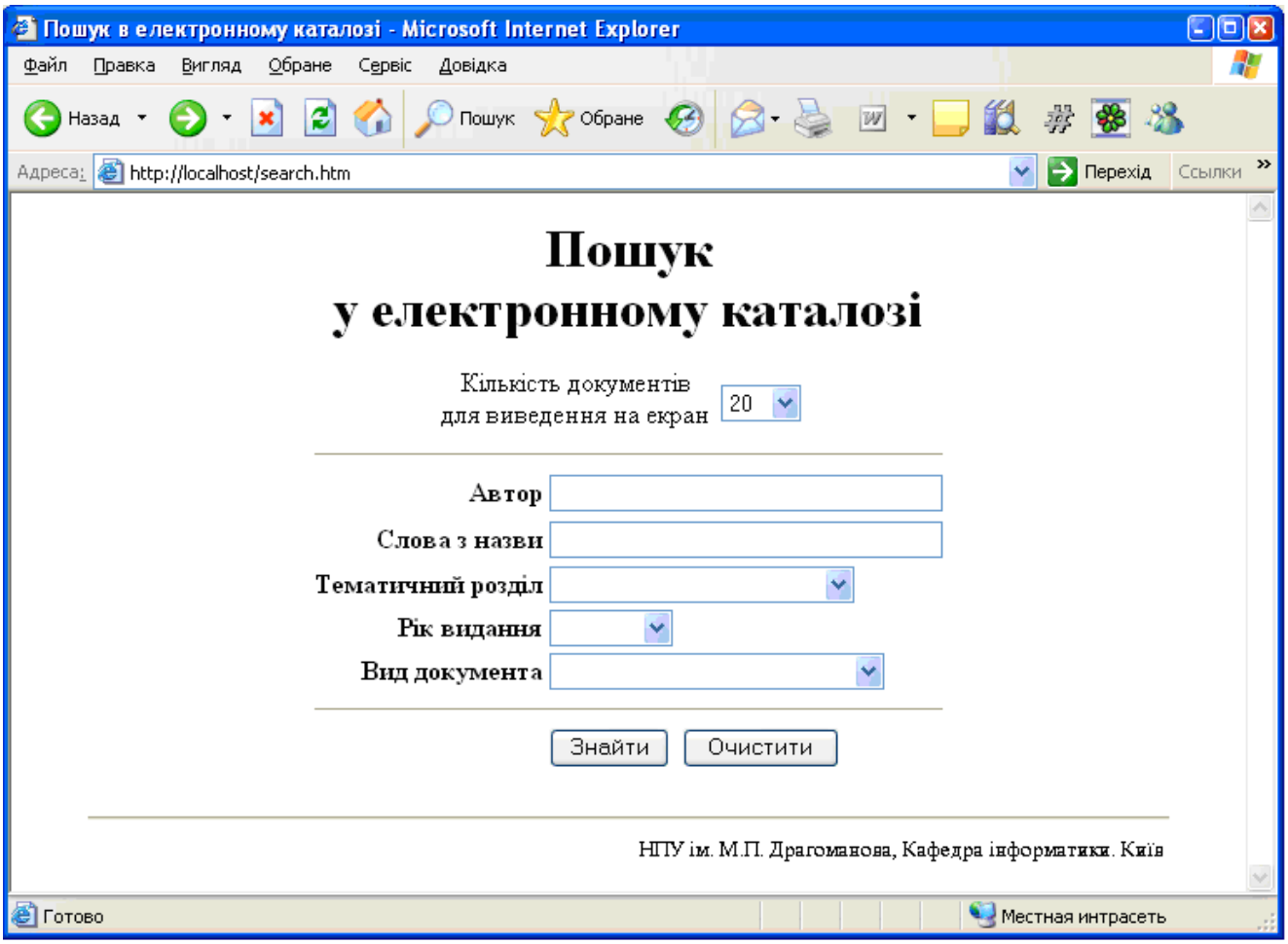
За методом POST на серверний модуль дані клієнта надходять із стандартного потоку введення. При цьому Web-сервер створює всі ті ж змінні оточення із супровідними даними, але додатково посилає модулю в стандартний потік введення тіло запиту, структура якого буде описана нижче. На додаток до цих змінних додається змінна CONTENT_LENGTH, що містить число символів даних, які необхідно прочитувати з потоку введення для подальшого опрацювання. Якщо для отримання даних використовувати оператор Read для читання одного символа з потоку введення, то він повинен викликатися таке число разів, яке вказане в змінній CONTENT_LENGTH. Також можна застосовувати оператор ReadLn, оскільки сервер (принаймні, Apache) в кінець даних додає управляючий символ кінця рядка.

Дані, відправлені серверному модулю в потоці введення, кодуються за наступними правилами.

- Латинські літери не кодуються.
- Вся решта символів, наприклад українські букви, а також різні розділові знаки, спеціальні символи замінюються шістнадцятковим поданням у вигляді %FF. На відміну від мови Object Pascal, де для вказування шістнадцяткового формату числа використовується знак $, в потоці, посланому сервером, перед шістнадцятковим числом стоїть знак %. Переведення в звичне подання символів здійснюється шляхом виділення зі всього вмісту блоків, що складаються з трьох символів, які починаються зі знака %, після якого слідує дві шестнадцяткові цифри. Такі конструкції зручно розглядати як рядки, в яких перший символ можна замінити на $, потім перевести їх в десяткове подання (нижче наведено текст функції strToint). За знайденим числом – кодом символа, за допомогою функції Chr можна отримати символ. Приблизно такий алгоритм реалізований у функції HTTPDecode бібліотеки httpapp.pas, що входить в поставку клієнт-серверного комплекту Delphi.
- Потік введення має наступну форму: ім’я елемента1=зміст1&ім’я елемента2= зміст2&..., де ім'я елемента є атрибутом name відповідного елемента форми Web-сторінки, з якою прийшов запит, а його вміст є або текстом, або певним значенням value.

Приклад. Створити серверний скрипт, за яким отримується запит, а потім відображаються на Web-сторінці дані з деякої бази даних.

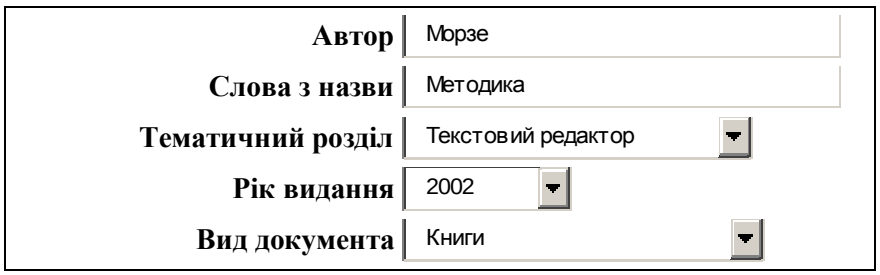
Для цього потрібна Web-сторінка з формою.




```
<html>
<head>
<title> Пошук в електронному каталозі </title>
<meta content="text/html; charset=windows-1251">
</head>
<body>
<center>
<h1>Пошук<br>в електронному каталозі</h1>
<form  action=http:\\localhost\cgi-bin\search.exe method=post>
<center>
<table><tbody><tr><td></td><td></td><td></td>
<td> Кількість документів <br> для виведення на екран </td>
<td>
<select name=number_row>
<option selected>20</option>
<option>50</option>
<option>100</option>
</select>
</td></tr></tbody></table>
<table><tbody><tr><td colspan=2>
<hr size=1>
</td></tr>
<tr>
<td align=right><b>Автор</b></td>
<td><input size=35 name=avtor></td></tr>
<tr>
<td align=right><b>Слова з назви</b></td>
<td><input size=35 name=book></td></tr>
<tr>
<td align=right><b>Тематичний розділ</b></td>
<td>
<select name=tema>
        <option selected></option>
        <option>Операційна система</option>
        <option>Текстовий редактор</option>
        <option>Електронні таблиці</option>
        <option>Бази даних</option>
```

```
        <option>Електронні презентації</option>
        <option>Інтернет</option>
        <option>Інтранет</option>
        <option>Локальні мережі</option>
        <option>Графічні редактори</option>
        <option>Теорія ймовірностей</option>
        <option>Логіка</option>
        <option>Дискретна математика</option>
        <option>Інше</option>
</select>
</td></tr>
<tr>
<td align=right><b>Рік видання</b></td>
<td>
<select name=year>
<option selected></option>
<option>2005</option><option>2004</option>
<option>2003</option><option>2002</option>
<option>2001</option><option>2000</option>
<option>1999</option><option>1998</option>
<option>1997</option><option>1996</option>
<option>1995</option><option>1994</option>
<option>1993</option><option>1992</option>
<option>1991</option><option>1990</option>
<option> до 1990 </option>
</select>
</td></tr>
<tr>
<td align=right><b>Вид документа</b></td>
<td>
<select name=type_doc>
<option selected></option>
<option> Книги </option>
<option> Дисертації </option>
<option> Автореферати дисертацій </option>
<option> Збірники праць </option>
<option> Матеріали конференцій </option>
</select>
</td></tr>
<tr>
<td colspan=2>
<hr size=1>
</td></tr>
<tr>
<td> </td>
<td>
<input type=submit value=Знайти>  
<input type=reset value=Очистити>
</td></tr></tbody></table>
</center>
</form>
<hr width="90%">
<center>
<table width="90%"><tbody><tr>
<td align=right>
<small>НПУ ім. М.П. Драгоманова, Кафедра інформатики. Київ
</small>
</td></tr></tbody></table>
</center>
</body>
</html>
```

Якщо тепер заповнити форму Web-сторінки значеннями, починаючи з першого поля форми, ввівши відповідно



то потік введення, який передається серверному модулю Web-сервером, матиме вигляд:

number_row=20&avtor=%CC%EE%F0%E7%E5&book=%CC%E5%F2%EE%E4%E8%EA%E0&tema=%D2%E5%EA%F1%F2%EE%E2%E8%E9+%F0%E5%E4%E0%EA%F2%EE%F0&year=2002&type_doc=%CA%ED%E8%E3%E8

Як видно, модулю, для того, щоб отримати дані та їх опрацювати, потрібно виконати наступні операції:

1. Після того, як дані будуть передані, наприклад, змінній і, потрібно виділяти значення, які починаються з назви елемента форми, а закінчуються символом &.
2. З рядків, що вийшли, прибрати назви полів і знак =.
3. Перетворити рядки, замінивши всі шістнадцяткові подання символів на звичайні, а знаки + на пропуски.
4. При необхідності деякі рядкові значення перетворити на відповідні їм числові або булеві.

Перші два пункти можна реалізувати за допомогою функції **subst.**

```
function subst(s1,s2: string):string;
begin
  delete(s1,1,Pos(s2+'=',s1) + Length(s2));
  if Pos('&',s1)>0
  then subst:=copy(s1, 1, pos('&',s1)-1)
  else subst := s1;
end;
```

Наступним етапом є переклад символів до нормального вигляду, це можна зробити за допомогою функції **russ.**

```
function russ(s: string):string;
var
st:string;
begin
while length(s)>0 do
begin
if s[1]='+' then
begin
st:=Concat(st,' ');
delete (s,1,1);
end;
if s [1]<>'%' then
begin
st:=Concat(st,s[1]);
delete (s,1,1);
end
else
begin
st:=Concat(st,char(strtoint(Concat('$',s[2],s[3]))));
delete(s,1,3);
end;
end;
Result:=st;
end;
```

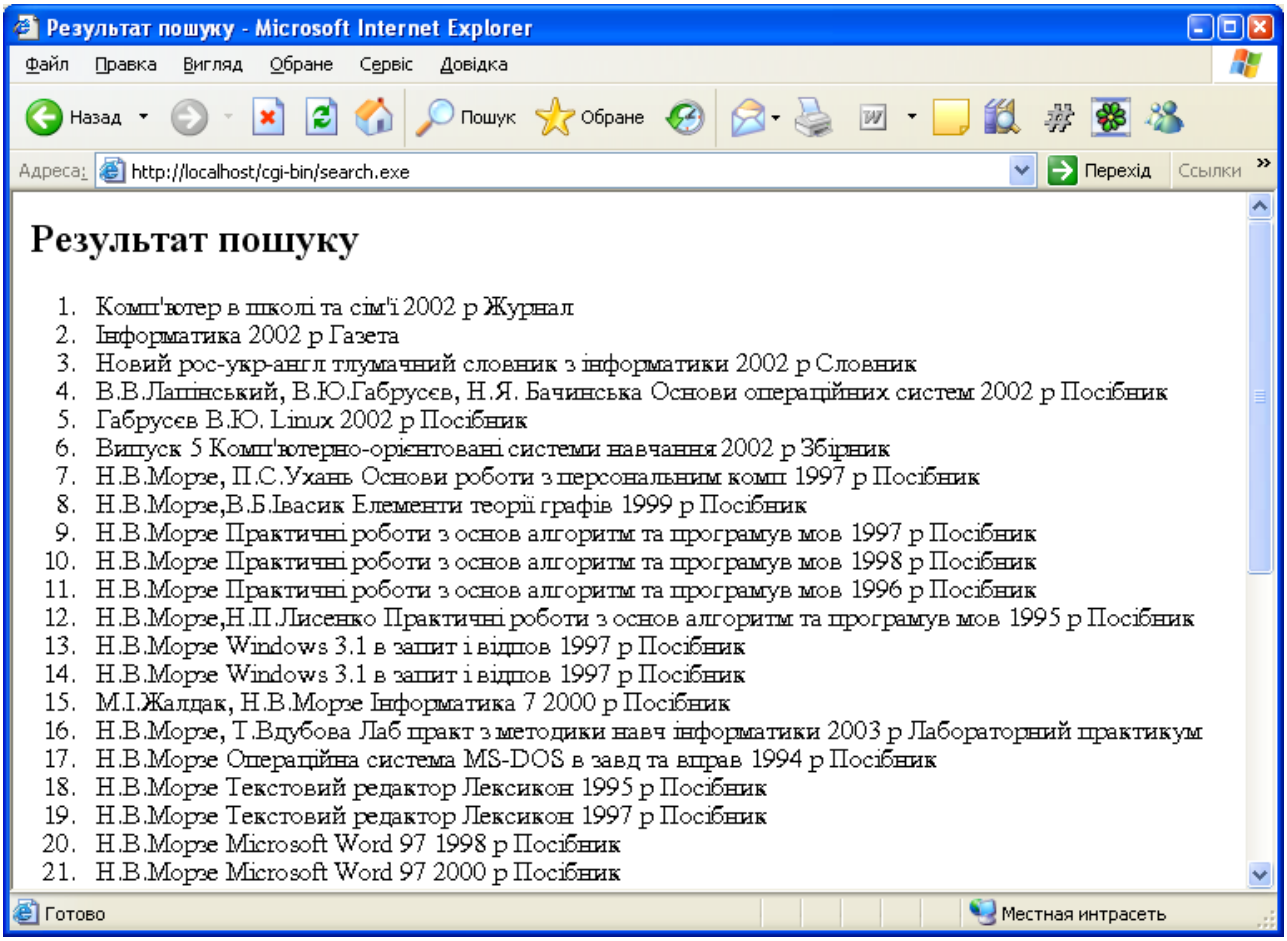
Ця функція послідовно "вибирає" від початку рядка s, що містить дані, які необхідно декодувати, послідовності розміром один або три символи, для попереднього опрацювання. Якщо перший символ рядка – %, то наступні за ним два символи перетворяться в рядкове подання шістнадцяткового числа, з подальшим перетворенням в тип integer і отриманням відповідного символу. Перетворені символи послідовно додаються до рядка st. Знак + додається до рядка st, як пропуск, а вся решта символів переноситься без змін. В кінці виконання функції вміст рядка st передається змінній Result. Отже, є функція, яку можна використовувати у всіх серверних модулях при необхідності декодування даних.

Далі потрібно, використовуючи обидві функції, надати користувачеві форму і залежно від введених значень вивести на Web-сторінку результат пошуку. Текст програми наведено нижче.

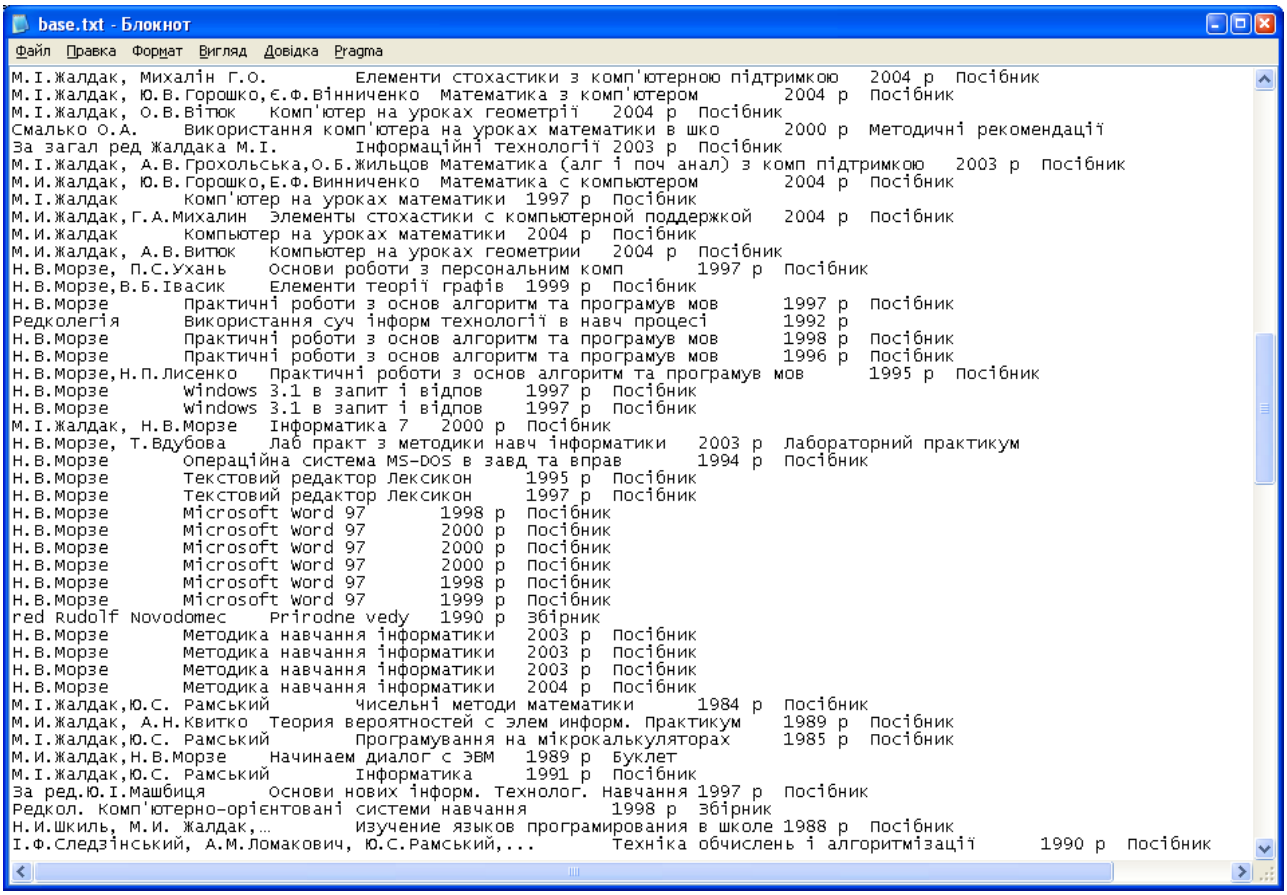
```
begin
readln(i); \\ рядкова змінна
writeln('Content-Type: text/html');
writeln;
writeln('<html>');
```

```
writeln('<title> Результат пошуку </title>');
writeln('<h2> Результат пошуку </h2>');
assignFile (f,'..\www\base.txt'); \\ файл з даними
{$i-}
reset (f);
{$i+}
if ioresult<>0
then writeln ('<p> Немає доступу до бази! </p>')
else
 begin
      WriteLn('<ol>');
      while not eof (f) do
      begin
          readln (f,s);
          if (Pos(russ(subst(i,'avtor')),s)>0) Or
            (Pos(russ(subst(i,'book')),s)>0) Or
            (Pos(russ(subst(i,'year')),s)>0) Or
            (Pos(russ(subst(i,'type_doc')),s)>0)
          then WriteLn('<li>',s,'</li>');
      end;
      CloseFile(f);
      WriteLn('</ol>');
 end;
writeln('</html>');
end.
```

Результат пошуку.



Файл з даними.



Метод GET

Даний метод також служить для передавання даних запиту серверному модулю, проте, на відміну від раніше вивченого методу POST, тут не використовується потік введення. Весь рядок запиту в методі GET передається в змінній оточення QUERY_STRING. Для доступу до неї можна використовувати функцію GetEnvironmentVariabie. В даному методі існує обмеження на число переданих символів, тому його краще використовувати тільки для передавання явно коротких наборів даних. Крім того, в даному методі використовується інший спосіб передавання даних від клієнта до Web-сервера. Замість передавання їх в тілі надісланого клієнтом HTTP-запиту вміст форми разом з назвами елементів додається до адреси запрошуваного документа. Наприклад, якщо замінити значення параметра форми action з POST на GET, то адреса документа, запрошуваного при натисненні на кнопку «Знайти», буде не

http:\\localhost\cgi-bin\search.exe,

а

http:\\localhost\cgi-bin\search.exe?number_row=20
&avtor=%CC%EE%F0%E7%E5&book=%CC%E5%F2%EE%E4%E8%EA%E0&tema=%D2%E5%EA%F1%F2%EE%E2%E8%E9+%F0%E5%E4%E0%EA%F2%EE%F0&year=2002&type_doc=%CA%ED%E8%E3%E8

Як видно, безпосередньо до адреси Web-документа додається символ ?, за яким слідує той же закодований вміст форми.

Для опрацювання цих даних за методом GET потрібно змінній і надати значення

GetEnvironmentVariable (Pchar('QUERY_STRING')),

все інше залишається без змін.

ЛІТЕРАТУРА

1. Англо-український тлумачний словник з обчислювальної техніки, Інтернету і програмування. – Вид. 1 – К.: Видавничий дім "СофтПрес", 2005. – 552 с.
2. Вирт Н. Алгоритмы и структуры данных: пер. с англ. – М.: Мир, 1989.
3. Ковальов В., Сайт http:// mcsa.ru
4. Культин Н. Б., Программирование на Object Pascal в Delphi 5. – СПб.: БХВ – Санкт-Петербург, 2000.
5. Онищенко С.М. Вивчення основ програмування на базі систем програмування Turbo Pascal та Delphi в середній загальноосвітній школі. // Комп'ютерно-орієнтовані системи навчання : Зб. Наук. праць/ Редкол. – К.:"Комп'ютер у школі та сім'ї". – 1998.
6. Онищенко С.М., Франчук В.М. Використання Delphi (Kylix) при вивченні основ програмування // Вісник: Зб. наук. статей НПУ ім. М.П. Драгоманова. – К.: НПУ ім. М.П. Драгоманова, 2004. Вип.6.