

» Idź do

- Spis treści
- Przykładowy rozdział

» Katalog książek

- Katalog online
- Zamów drukowany katalog

» Twój koszyk

- Dodaj do koszyka

» Cennik i informacje

- Zamów informacje o nowościach
- Zamów cennik

» Czytelnia

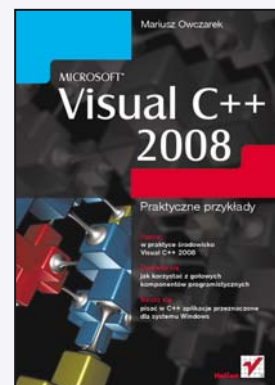
- Fragmenty książek online

» Kontakt

Helion SA
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl
© Helion 1991-2010

Microsoft Visual C++ 2008. Praktyczne przykłady

Autor: Mariusz Owczarek
ISBN: 978-83-246-2587-1
Format: 158×235, stron: 360



- Poznaj w praktyce środowisko Visual C++ 2008
- Dowiedz się, jak korzystać z gotowych komponentów programistycznych
- Naucz się pisać w C++ aplikacje przeznaczone dla systemu Windows

Wiele osób słyszało o języku C++, lecz ogromna większość z nich na samą myśl o bliższym poznaniu go reaguje z podobną rezerwą, jak na pomysł wybrania się na wakacyjny kurs języka mandaryńskiego. Osoby te popełniają jednak niewybaczalny błąd i skutecznie odcinają się od niezwykle interesującego świata nowoczesnego programowania. Na ludzi biegle znających C++ czeka bardzo wiele atrakcyjnych ofert pracy, a niemała popularność systemów Windows sprawia, że najbardziej poszukiwani w tej grupie są specjaliści znający Visual C++, czyli wygodne w użyciu i bardzo rozbudowane środowisko programistyczne firmy Microsoft. Nie trzeba chyba dodawać, że zarabiają oni zwykle znacznie więcej niż najlepsi nawet tłumacze z języka mandaryńskiego...

Jeśli chcesz dołączyć do tej programistycznej elity, już dziś sięgnij po odpowiednie źródło wiedzy. Książka „Microsoft Visual C++ 2008. Praktyczne przykłady” umożliwi Ci gładkie rozpoczęcie przygody z Visual C++, prezentując podstawowe informacje na temat języka, opisując środowisko programistyczne, sposób tworzenia aplikacji oraz technikę zarządzania danymi i korzystania z plików. Nauczysz się z niej również właściwego stosowania elementów kontrolnych i komponentów programistycznych. Poznasz metody przetwarzania wielowątkowego, zagadnienia związane z grafiką i łączeniem się z siecią, a ponadto dowiesz się, jak skonstruowany jest kod programu. Na tym jednak nie koniec, bowiem na kilku rozbudowanych i bardzo zróżnicowanych przykładach poznasz też praktyczne zastosowanie całej tej teorii, a wszystko przy użyciu bezpłatnej wersji środowiska programistycznego firmy Microsoft.

- Podstawowe informacje na temat Visual C++ 2008
- Instalacja środowiska programistycznego
- Elementy składowe aplikacji i sposoby używania kontrolerek
- Zarządzanie danymi i używanie plików
- Używanie najważniejszych komponentów programistycznych
- Wykorzystanie elementów graficznych
- Podstawy korzystania z wątków
- Korzystanie z komponentów sieciowych
- Opis języka C++ i C++/CLI oraz struktury napisanych w nich programów
- Praktyczne projekty aplikacji opracowanych za pomocą Visual C++ 2008

**Nie zwlekaj – już dziś wkrocz w magiczny świat programowania
z wykorzystaniem Visual C++ 2008!**

Spis treści

Co znajdziesz w tej książce?	9
Rozdział 1. Podstawy środowiska Visual C++ 2008	11
Język C++ a .NET Framework	11
Opis środowiska	12
Pobieranie i instalacja środowiska	12
Główne okno VC++ 2008	13
Tworzenie nowej aplikacji w VC++ 2008	13
Wygląd środowiska w trybie budowy aplikacji	16
Struktura projektu	17
Klasa okna głównego	18
Rozdział 2. Podstawowe elementy aplikacji	21
Główne okno	21
Przyciski	26
Etykiety	27
Pola tekstowe	29
Wprowadzanie danych do aplikacji za pomocą pól tekstowych	31
Wprowadzanie danych z konwersją typu	32
Wyświetlanie wartości zmiennych	34
Pole tekstowe z maską formatu danych	35
Pola wyboru, przyciski opcji, kontenery grupujące	37
Rozdział 3. Menu i paski narzędzi	41
Rodzaje menu	41
Komponent MenuStrip	41
Menu podręczne	47
Skróty klawiaturowe w menu	49
Paski narzędzi	51
Rozdział 4. Tablice, uchwyt i dynamiczne tworzenie obiektów	55
Tablice	55
Uchwyt	59
Dynamiczne tworzenie obiektów — operator gcnew	60
Dynamiczna deklaracja tablic	61

Rozdział 5. Komunikacja aplikacji z plikami	63
Pliki jako źródło danych	63
Wyszukiwanie plików	64
Odczyt własności plików i folderów	65
Odczyt danych z plików tekstowych	66
Zapisywanie tekstu do pliku	69
Zapis danych do plików binarnych	71
Odczyt z plików binarnych	72
Rozdział 6. Okna dialogowe	75
Okno typu MessageBox	75
Okno dialogowe otwarcia pliku	77
Okno zapisu pliku	79
Okno wyboru koloru	80
Wybór czcionki	81
Rozdział 7. Możliwości edycji tekstu w komponencie TextBox	83
Właściwości pola TextBox	83
Kopiowanie i wklejanie tekstu ze schowka	85
Wyszukiwanie znaków w tekście	86
Wstawianie tekstu między istniejące linie	87
Wprowadzanie danych do aplikacji	88
Prosta konwersja typów — klasa Convert	88
Konwersja ze zmianą formatu danych	89
Konwersja liczby na łańcuch znakowy	92
Rozdział 8. Komponent tabeli DataGridView	95
Podstawowe właściwości komponentu DataGridView	95
Zmiana wyglądu tabeli	98
Dopasowanie wymiarów komórek tabeli do wyświetlanego tekstu	101
Odczytywanie danych z komórek tabeli	102
Zmiana liczby komórek podczas działania aplikacji	106
Tabela DataGridView z komórkami różnych typów	110
Przyciski w komórkach — DataGridViewButtonCell	113
Komórki z polami wyboru — DataGridViewCheckBoxCell	114
Grafika w tabeli — komórka DataGridViewImageCell	116
Komórka z listą rozwijaną — DataGridViewComboBoxCell	117
Odnośniki internetowe w komórkach — DataGridViewLinkCell	119
Rozdział 9. Metody związane z czasem — komponent Timer	123
Czas systemowy	123
Komponent Timer	125
Rozdział 10. Grafika w aplikacjach Visual C++	127
Obiekt Graphics — kartka do rysowania	127
Pióro Pen	133
Pędzle zwykłe i teksturowane	135
Rysowanie pojedynczych punktów — obiekt Bitmap	137
Rysowanie trwale — odświeżanie rysunku	138
Rozdział 11. Podstawy aplikacji wielowątkowych	141
Wątki	141
Komunikacja z komponentami z innych wątków — przekazywanie parametrów	143
Przekazywanie parametrów do metody wątku	145
Klasa wątku — przekazywanie parametrów z kontrolą typu	146
Komponent BackgroundWorker	148

Rozdział 12. Połączenie aplikacji z siecią internet	153
Komponent WebBrowser	153
Przetwarzanie stron WWW — obiekt HtmlDocument	156
Protokół FTP	160
Pobieranie zawartości katalogu z serwera FTP	161
Pobieranie plików przez FTP	162
Wysyłanie pliku na serwer FTP	164
Rozdział 13. Dynamiczne tworzenie okien i komponentów	167
Wyświetlanie okien — klasa Form	167
Komponenty w oknie tworzonym dynamicznie	169
Przesyłanie danych z okien dialogowych	170
Okno tytułowe aplikacji	171
Obsługa zdarzeń dla komponentów tworzonych dynamicznie	172
Aplikacja zabezpieczona hasłem	173
Rozdział 14. Struktura programów C++ i C++/CLI	175
Programy korzystające z konsoli w VC++ 2008	175
Ogólna postać programu pisanego w C++	176
Dyrektywy	177
Dyrektywa #include	177
Dyrektywa #define	178
Dyrektywa #if — kompilacja warunkowa	178
Typy zmiennych	178
Zmienne typu int	178
Zmienne typu float	179
Typ double	179
Typ char	179
Modyfikatory typów	179
Rzutowanie (konwersja) typów	179
Typ wyliczeniowy	180
Operatory	180
Zapis danych do plików i odczyt z nich za pomocą operatorów << i >>	182
Wskaźniki	184
Tablice	184
Operatory new i delete	185
Instrukcje	186
Instrukcje warunkowe	186
Instrukcje iteracji	187
Funkcje	188
Przeciążanie funkcji	189
Niejednoznaczność	189
Przekazywanie argumentów przez wartość i adres	190
Wskaźniki do funkcji	191
Funkcja main()	192
Przekazywanie parametrów do funkcji main()	193
Struktury i unie	195
Struktury	195
Klasy	196
Konstruktory i destrukторы	199
Przeciążanie konstruktorów	201
Przeciążanie operatorów	202
Statyczne metody i pola klasy	203
Wskaźnik zwrotny this	204
Dziedziczenie	205

Własne kontrolki dziedziczące po standardowych	208
Przestrzenie nazw	211
Wyjątki	212
Aplikacje, aplikacje	215
Rozdział 15. Prosty edytor tekstu	217
Opis	217
Główne okno	217
Budowa interfejsu użytkownika	219
Otwieranie pliku z dysku	223
Zapisanie pliku tekstowego	226
Obsługa schowka	227
Cofanie komend (Undo)	228
Wycięcie tekstu do umieszczenia w schowku	228
Aby skopiować tekst do schowka	228
Aby wkleić tekst ze schowka	229
Test	229
Rozdział 16. Program do rysowania	231
Opis	231
Budowa interfejsu użytkownika	231
Wczytywanie pliku graficznego z dysku	233
Wyświetlanie grafiki w oknie	234
Rysowanie w oknie za pomocą myszy	236
Zmiana koloru linii	239
Zapis pliku graficznego na dysku	240
Test	243
Rozdział 17. Figury Voronoi	245
Opis	245
Interfejs użytkownika	246
Struktura projektu C++/CLI	246
Wyznaczanie odległości między punktami	248
Rysowanie figur	248
Obsługa programu	251
Test	252
Rozdział 18. Automat komórkowy	253
Opis	253
Interfejs użytkownika	254
Inicjalizacja planszy	256
Zaznaczanie pól w siatce	257
Krok w trybie Gra w życie	258
Krok w trybie Seeds	260
Kroki w trybie automatycznym	261
Obsługa programu	262
Inne ciekawe układy do Gry w życie	262
Oscylatory	263
Obiekty latające	264
Test	266
Rozdział 19. Wieże Hanoi	267
Opis	267
Interfejs użytkownika	268
Deklaracja zmiennych klasy Form1	270

Obsługa menu	270
Funkcja rysująca krążki	272
Przekładanie krążków	275
Różne końcowe metody	278
Test	279
Rozdział 20. Aplikacja bazy danych	281
Opis	281
Instalacja PostgreSQL	281
Wyłączenie usługi bazy	284
Inicjalizacja bazy	285
Organizacja i typy danych w bazach PostgreSQL	286
Język SQL	288
Utworzenie bazy danych	288
Interfejs użytkownika	290
Włączenie sterowników bazy PostgreSQL do projektu	291
Łączenie z bazą i odczyt danych	292
Dodawanie danych do bazy	294
Zmiana danych w bazie	296
Kasowanie danych	299
Obsługa bazy	300
Test	300
Rozdział 21. Aplikacja do nauki słówek	303
Opis	303
Interfejs użytkownika	303
Deklaracja pól klasy	305
Odczyt danych z pliku tekstowego	306
Odpytywanie ze słówek	308
Zapis listy wyrazów do pliku tekstowego	309
Obsługa	310
Test	310
Rozdział 22. Aplikacja do monitorowania systemu	313
Opis	313
Interfejs użytkownika	313
Ustawienie parametrów kontrolki performanceCounter	314
Odczyt parametrów z kontrolki performanceCounter	316
Uruchamianie timera — ikona w pasku zadań	316
Rysowanie wykresów	318
Opis	321
Test	322
Rozdział 23. Klient FTP	323
Opis	323
Interfejs użytkownika	323
Klasa do obsługi FTP	324
Pobieranie katalogu	326
Pobieranie plików	328
Wysyłanie plików	329
Poruszanie się po folderach	329
Rozbudowa	331
Test	331

Rozdział 24. Aplikacja do drukowania grafiki	333
Opis	333
Interfejs użytkownika	333
Otwieranie rysunku	334
Obracanie rysunku	335
Drukowanie	336
Opis działania	338
Test	338
 Odpowiedzi do testów	 341
Skorowidz	343

Rozdział 12.

Połączenie aplikacji z siecią internet

Komponent WebBrowser

Czasami istnieje potrzeba wyświetlania w oknie aplikacji danych pobranych bezpośrednio ze stron WWW. W VC++ 2008 mamy komponent, który jest właściwie kompletną przeglądarką stron opartą na Internet Explorerze.

Za pomocą tego komponentu w prosty sposób można wyświetlać zawartość całych stron WWW w oknie aplikacji. Może być on użyty nie tylko do przeglądania stron w sieci, ale także do wyświetlania dokumentów HTML z komputera lokalnego (na przykład plików pomocy aplikacji). Podstawowe właściwości komponentu `WebBrowser` przedstawia tabela 12.1.

Tabela 12.1. Wybrane właściwości kontrolki `WebBrowser`

Właściwość	Znaczenie
<code>AllowNavigation</code>	Właściwość umożliwiająca zablokowanie przeglądarki tak, że nie można przejść na inne strony niż aktualna. Wartość <code>false</code> oznacza zablokowanie.
<code>Url</code>	Adres strony do wyświetlenia w przeglądarce. Ta właściwość jest typu <code>Uri^</code> .
<code>CanGoBack</code>	Wartość <code>true</code> oznacza, że odwiedzana strona nie jest pierwszą (istnieje historia).
<code>CanGoForward</code>	Wartość <code>true</code> oznacza, że użytkownik cofa się w historii odwiedzanych stron i wyświetlana strona nie jest ostatnią odwiedzoną.
<code>Document</code>	Właściwość typu <code>HtmlDocument</code> zawierająca aktualnie wyświetlaną w kontrolce stronę. Może być użyta do pobrania danych ze strony.
<code>DocumentText</code>	Zawiera źródło HTML strony aktualnie wyświetlonej w przeglądarce.
<code>DocumentTitle</code>	Tytuł aktualnie wyświetlanej strony.

Najprostszy sposób wyświetlenia strony WWW pokazuje przykład.

Przykład 12.1

Po naciśnięciu przycisku wyświetl w oknie aplikacji stronę *helion.pl*.

Rozwiązanie

Do okna aplikacji wstaw kontrolkę *WebBrowser* (zakładka okna narzędziowego, ostatnia kontrolka w dziale *Common Controls*) oraz przycisk *Button*.

Powiększ rozmiary okna aplikacji i kontrolki *WebBrowser* tak, aby zwiększyć komfort oglądania stron.

Do zdarzenia *Click* przycisku przypisz następującą metodę:

```
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {  
    Uri^ adres= gcnew Uri("http://helion.pl");  
    webBrowser1->Url=adres;  
}
```

Uruchom aplikację, po naciśnięciu przycisku w oknie kontrolki pojawi się odpowiednia strona WWW (rysunek 12.1).



Rysunek 12.1. Wyświetlanie stron WWW w komponentie *WebBrowser*



Adres przekazywany do właściwości *Url* należy zawsze poprzedzać prefiksem. W przypadku stron WWW jest to *http://*.

W podobny sposób można wyświetlić plik lokalny.

Przykład 12.2

Utwórz w katalogu na dysku *C:* folder *Aplikacja*, a następnie w tym folderze plik *pomoc.html* o dowolnej treści.

Wyświetl w oknie kontrolki WebBrowser plik *pomoc.html* znajdujący się w folderze *C:\Aplikacja* lub inny plik HTML.

Rozwiązanie

Zbuduj program identyczny jak w poprzednim przykładzie, zmień jedynie adres dokumentu.

```
Uri^ adres= gcnew Uri("c:\\aplikacja\\pomoc.html");
```

Program będzie teraz wyświetlał plik lokalny.

Klasa kontrolki WebBrowser posiada też wiele metod, które umożliwiają nawigację po stronach WWW. Przedstawia je tabela 12.2.

Tabela 12.2. Wybrane metody obiektu WebBrowser

Metoda	Działanie
GoBack()	Przenosi użytkownika do poprzedniej strony w historii. Działa tylko, jeżeli właściwość CanGoBack==true.
GoForward()	Przenosi do następnej strony w historii, jeżeli użytkownik cofał się wcześniej.
GoHome()	Przenosi do strony domowej. Strona domowa jest tą samą, jaka została określona w Internet Explorerze.
GoSearch()	Przenosi do strony domyślnej wyszukiwarki. Również lokalizacja tej strony jest pobierana z Internet Explorera.
Navigate(System::String adres) Navigate(Uri adres)	Wyświetla w kontrolce stronę o adresie <i>adres</i> .
Stop()	Zatrzymuje wczytywanie aktualnej strony.

Bez problemu można dodać możliwość przechodzenia do stron wcześniej odwiedzonych, tak jak w przeglądarce. Chociaż pisanie kolejnej przeglądarki internetowej mija się właściwie z celem, to nawigację można wykorzystać do opracowania na przykład plików pomocy czy prezentacji, którą będzie można oglądać wewnątrz aplikacji.

Przykład 12.3

Napisz przeglądarkę stron WWW z możliwością poruszania się po historii odwiedzanych stron.

Rozwiązanie

Utwórz aplikację i dodaj do jej okna komponent WebBrowser, dwa przyciski i pole tekstowe. We właściwości Text pierwszego przycisku wpisz *Wstecz*, a drugiego — *Naprzód*.

W polu tekstowym będziemy wpisywać stronę do odwiedzenia, jej wczytanie powinno nastąpić, kiedy użytkownik naciśnie klawisz *Enter*. Aby tak się stało, trzeba obsłużyć zdarzenie *KeyDown* dla pola tekstowego. Zaznacz pole tekstowe myszą w widoku

projektowania aplikacji, odnajdź to zdarzenie w panelu zdarzeń, a następnie kliknij je dwukrotnie. Utworzy się metoda obsługi tego zdarzenia, jednym z parametrów tej metody będzie rodzaj naciśniętego klawisza. Oto kod tej metody, w którym przejście do strony następuje przy wykryciu naciśnięcia klawisza *Enter*.

```
private: System::Void textBox1_KeyDown(System::Object^ sender,
    System::Windows::Forms::KeyEventArgs^ e) {
    if (e->KeyData==System::Windows::Forms::Keys::Enter)
        webBrowser1->Navigate("http://" + textBox1->Text);
}
```

Teraz wystarczy już tylko zaprogramować metody przycisków — odpowiednio: cofające lub przynoszące do przodu w historii.

```
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {
    webBrowser1->GoBack();
}
private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e) {
    webBrowser1->GoForward();
}
```

Po wypróbowaniu zapisz projekt na dysku, ponieważ będziemy jeszcze z niego korzystać.

Przetwarzanie stron WWW — obiekt `HtmlDocument`

Do lepszego zrozumienia tej części rozdziału konieczna jest znajomość podstaw składni języka HTML. Celem użycia kontrolki `WebBrowser` nie będzie raczej napisanie kolejnej przeglądarki WWW, bo takich jest już dużo. Zamiast tego za pomocą tej kontrolki można wykonywać operacje na dokumentach pisanych w języku HTML. Środowisko .NET Framework zawiera klasę `HtmlDocument`, która reprezentuje dokument tego typu. Za pomocą tego obiektu uzyskujemy dostęp do poszczególnych części składowych pliku HTML. Te części składowe są zawarte w obiektach `HtmlElement`. Obiekt typu `HtmlDocument` grupuje więc kilka obiektów `HtmlElement`.

Właściwości obiektu `HtmlDocument` przedstawia tabela 12.3.

Obiekt `HtmlElement` posiada właściwości ogólne odnoszące się do wszystkich rodzajów sekcji kodu HTML. Najbardziej interesujące są zwykle właściwości szczególne, odnoszące się do konkretnych części kodu, na przykład znacznik `SRC` w kodzie wstawiania obrazka oznacza ścieżkę do pliku graficznego. Dostęp do tych szczególnych właściwości uzyskujemy za pomocą metod `GetAttribute()` lub `SetAttribute()`. Argumentami tych metod jest znacznik w kodzie, do którego chcemy uzyskać dostęp (na przykład `SRC` dla odczytania ścieżki dostępu do obrazka).

Tabela 12.3. Niektóre właściwości obiektu *HtmlDocument*

Właściwość	Znaczenie
All	Tabela obiektów <i>HtmlElement</i> zawierająca wszystkie części składowe dokumentu.
Body	Element zawierający część dokumentu po znaczniku <i>BODY</i> .
Cookie	Zawiera wszystkie znaczniki kontekstu (ang. <i>cookies</i>) powiązane z danym dokumentem.
Encoding	Kodowanie używane przez aktualny dokument.
Forms	Tabela obiektów <i>HtmlElement</i> zawierająca wszystkie części po znacznikach <i>FORM</i> .
Images	Obiekty <i>HtmlElement</i> reprezentujące części dokumentu po znacznikach <i>IMG</i> (obrazy).
Links	Zbiór odnośników do innych stron zawartych w aktualnym dokumencie.
Title	Tytuł dokumentu.

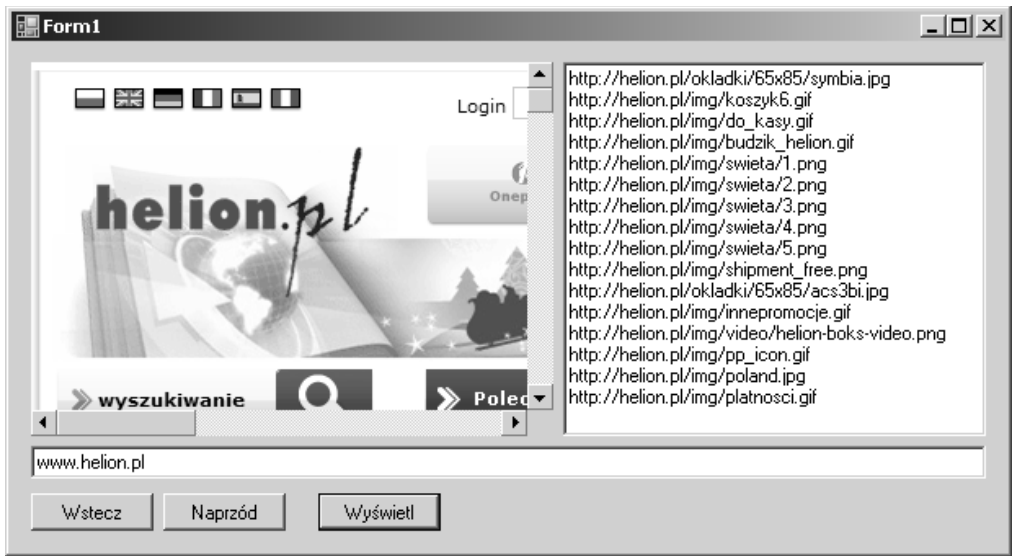
Przykład 12.4

Wypisz w polu tekstowym ścieżki do wszystkich plików graficznych na stronie WWW wyświetlonej w kontrolce *WebBrowser*.

Rozwiązanie

Otwórz projekt z przykładu 12.3.

Zmniejsz trochę obszar kontrolki *WebBrowser* i dodaj do okna aplikacji kolejny przycisk *Button* oraz pole tekstowe *TextBox*; całość niech wygląda jak na rysunku 12.2.



Rysunek 12.2. Aplikacja pokazująca obiekty graficzne na stronie

Po naciśnięciu trzeciego przycisku w polu tekstowym powinny się pojawić odnośniki do wszystkich obrazków zawartych na wyświetlanej stronie WWW. Aby to zrobić, skorzystamy z właściwości `Image` obiektu `HtmlDocument`. Właściwość `Image` jest typu tablicowego, do odczytu jej elementów użyjemy enumeratora (porównaj przykład 4.2). Do zdarzenia `Click` przycisku przypisz metodę:

```
private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e) {
    System::Collections::IEnumerator^ element=
        webBrowser1->Document->Images->GetEnumerator();

    element->MoveNext();
    while ((element->MoveNext())&&(element!=nullptr)) {
        textBox2->AppendText(((HtmlElement^)(element->Current))->
            GetAttribute("SRC")->ToString());
        textBox2->AppendText(System::Environment::NewLine);
    }
}
```

Działanie programu dla strony *helion.pl* pokazuje rysunek 12.2.

W łatwy sposób można też napisać program, który będzie sprawdzał, czy dana strona WWW posługuje się jakimiś znacznikami kontekstu. Wykorzystamy do tego odpowiednią właściwość obiektu `HtmlDocument`.

Przykład 12.5

Wypożaż przeglądarkę w możliwość podglądu znaczników kontekstu na danej stronie.

Rozwiązanie

Utwórz aplikację identyczną jak w przykładzie 12.4.

Tym razem metoda wywoływana przy naciśnięciu trzeciego przycisku jest następująca:

```
private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e) {
    System::String^ cookie;
    cookie=webBrowser1->Document->Cookie;
    textBox2->Clear();
    if (cookie!=nullptr)
        textBox2->AppendText(cookie);
    else
        textBox2->AppendText("Nie znaleziono znaczników kontekstu!");
}
```

Przykład 12.6

Po naciśnięciu przycisku wyświetl w polu tekstowym wszystkie odnośniki znajdujące się na danej stronie.

Rozwiązanie

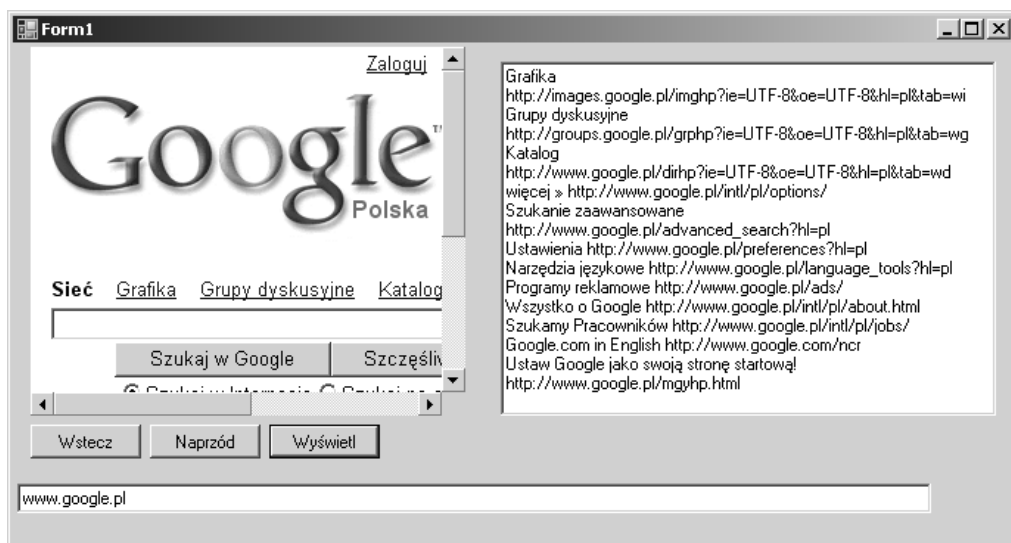
Utwórz aplikację jak w przykładzie 12.4.

Po naciśnięciu trzeciego przycisku odczytamy zawartość właściwości `Links` dla danej strony. Podobnie jak to było we właściwości `Image`, jest to tablica obiektów `HtmlElement`, którą będziemy odczytywać za pomocą enumeratora.

Oto odpowiednia metoda trzeciego przycisku. Właściwość `InnerText` obiektu `HtmlElement` pozwala na odczytanie tekstu związanego z odnośnikiem.

```
private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e) {
    System::Collections::IEnumerator^ odnosnik=
        webBrowser1->Document->Links->GetEnumerator();
    odnosnik->MoveNext();
    while ((odnosnik->MoveNext())&&(odnosnik!=nullptr)) {
        textBox2->AppendText(((HtmlElement^)(odnosnik->Current))->
            InnerText->ToString()+" ");
        textBox2->AppendText(((HtmlElement^)(odnosnik->Current))->
            GetAttribute("href")->ToString());
        textBox2->AppendText(System::Environment::NewLine);
    }
}
```

Rysunek 12.3 pokazuje aplikację w działaniu.



Rysunek 12.3. Wyświetlanie odnośników ze strony WWW

Protokół FTP

Protokół FTP służy do przesyłania plików przez internet. Można go użyć we wnętrzu aplikacji na przykład do automatycznego pobrania uaktualnienia lub potrzebnych plików z danymi.

Implementacja FTP w .NET Framework jest na poziomie, który nazwałbym „półniskim”, co oznacza, że nie trzeba mieć wiedzy o FTP, aby się nim posługiwać, ale nie jest to też kwestia użycia jednej metody pobierającej lub wysyłającej pliki. Połączenia FTP umożliwiają obiekty dwóch klas: `FtpWebRequest` i `FtpWebResponse`. Pierwszy z nich reprezentuje zapytanie do serwera FTP, a drugi odpowiedź serwera. Do poprawnej pracy będą potrzebne dwie właściwości obiektu `FtpWebRequest`, które przedstawia tabela 12.4.

Tabela 12.4. Wybrane właściwości obiektu *FtpWebRequest*

Właściwość	Znaczenie
<code>Credentials</code>	Zawiera login i hasło stosowane przy logowaniu do serwera FTP.
<code>Method</code>	Określa rodzaj operacji do wykonania na serwerze. Możliwe wartości to: ◆ <code>WebRequestMethods::Ftp::DownloadFile</code> — pobranie pliku, ◆ <code>WebRequestMethods::Ftp::UploadFile</code> — wysłanie pliku, ◆ <code>WebRequestMethods::Ftp::ListDirectoryDetails</code> — pobranie szczegółowych informacji o plikach znajdujących się na serwerze.

Oprócz tych właściwości będziemy używać dwóch metod opisanych w tabeli 12.5.

Tabela 12.5. Metody obiektu *FtpWebRequest* służące do pobierania lub wysyłania danych

Metoda	Działanie
<code>GetResponse()</code>	Zwraca obiekt typu <code>FtpWebResponse</code> , z którego możemy czytać dane wysyłane przez serwer.
<code>GetRequestStream()</code>	Zwraca strumień, do którego można pisać dane, jakie mają być wysłane na serwer FTP.

Ogólnie zaprogramowanie pobierania danych z FTP wymaga następujących czynności:

- Utworzenie obiektu `FtpWebRequest` — parametrem dla konstruktora obiektu jest adres serwera; w przypadku pobrania lub wysłania pliku parametrem jest pełna ścieżka wraz z nazwą pliku.
- Zapisanie we właściwości `Credentials` loginu i hasła do serwera.
- Wybranie czynności (wysłanie bądź pobranie pliku, pobranie zawartości katalogu) i odpowiednie ustawienie właściwości `Method`.
- Wysłanie zapytania do serwera i pobranie odpowiedzi (czyli obiektu `FtpWebResponse`) za pomocą metody `GetResponse()`.
- Pobranie strumienia odpowiedzi z obiektu `FtpWebResponse` i pobieranie z niego danych (zawartości pliku lub katalogu).

W przypadku wysłania pliku na serwer czynności od a) do c) są takie same, a dalej mamy:

- d) Otwarcie strumienia do pisania na serwer metodą `GetRequestStream()` obiektu `FtpWebRequest`.
- e) Zapisanie danych (zawartości pliku lokalnego) do tego strumienia.

Pobieranie zawartości katalogu z serwera FTP

Teraz zobaczmy, jak praca z FTP wygląda w praktyce.

Przykład 12.7

Pobierz zawartość podanego katalogu z podanego serwera FTP.

Rozwiązanie

Do nowego okna aplikacji wstaw dwa pola tekstowe `TextBox` oraz przycisk `Button`.

Aby program działał, dodaj do niego możliwość korzystania z przestrzeni nazw `System::Net` i `System::IO`, które zawierają potrzebne klasy.

```
using namespace System::Net;
using namespace System::IO;
```

Ustaw właściwość `Multiline` pola tekstowego `textBox2` na `true` i powiększ je tak, aby mogło wyświetlić kilka linii tekstu.

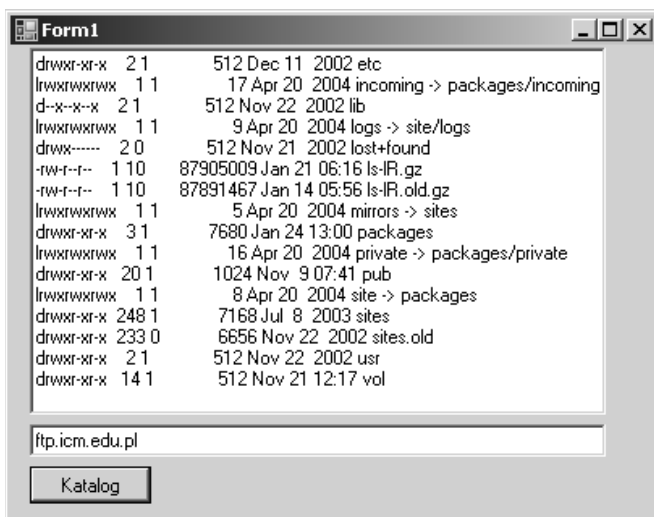
Metodę obsługującą zdarzenie `Click` napisz jak niżej:

```
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {
    Uri^ adres = gcnew Uri("ftp://" + textBox1->Text);
    FtpWebRequest^ req =
        dynamic_cast<FtpWebRequest^>(WebRequest::Create(adres));
    req->Credentials = gcnew
        NetworkCredential("anonymous", "moja_nazwa@moj_adres.pl");
    req->Method = WebRequestMethods::Ftp::ListDirectoryDetails;
    FtpWebResponse^ resp;
    resp = dynamic_cast<FtpWebResponse^>(req->GetResponse());
    Stream^ resp_stream = resp->GetResponseStream();
    StreamReader^ reader = gcnew StreamReader(resp_stream);
    String^ linia;
    textBox2->Clear();
    while (!reader->EndOfStream) {
        linia = reader->ReadLine();
        textBox2->AppendText(linia + System::Environment::NewLine);
    }
}
```

Po uruchomieniu programu wpisz adres dowolnego publicznego serwera FTP i naciśnij przycisk. W polu tekstowym otrzymasz zawartość głównego katalogu. Działanie aplikacji przedstawia rysunek 12.4. Zapisz aplikację na dysku.

Rysunek 12.4.

*Aplikacja pobierająca
zawartość folderu
z serwera FTP*



Pobieranie plików przez FTP

Po znalezieniu potrzebnego pliku można go pobrać na dysk. Rozszerzymy aplikację o taką możliwość.

Przykład 12.8

Dodaj do aplikacji z przykładu 12.7 możliwość pobrania pliku.

Rozwiązanie

Otwórz aplikację z poprzedniego przykładu i umieść trzeci przycisk Button oraz jeszcze jedno pole tekstowe do wpisywania nazwy pliku do pobrania.

Metoda zdarzenia Click dla drugiego przycisku będzie miała postać:

```
private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e) {
    Uri^ adres = gcnew Uri("ftp://" + textBox1->Text + "/" + textBox3->Text);
    FtpWebRequest^ req =
        dynamic_cast<FtpWebRequest^>(WebRequest::Create(adres));
    req->Credentials = gcnew
        NetworkCredential("anonymous", "moja_nazwa@moj_adres.pl");
    req->Method = WebRequestMethods::Ftp::DownloadFile;
    FtpWebResponse^ resp = dynamic_cast<FtpWebResponse^>(req->GetResponse());
    Stream^ resp_stream = resp->GetResponseStream();
}
```

```

FileStream^ stru_plik =
    gcnew FileStream("../"+textBox3->Text, FileMode::Create);
// czytaj plik z serwera i zapisuj do strumienia
int ile_bajtow;
array<Byte> ^ bufor = gcnew array<Byte>(1024);
do {
    ile_bajtow=resp_stream->Read(bufor,0,bufor->Length);
    stru_plik->Write(bufor,0,ile_bajtow);
} while(ile_bajtow!=0);
stru_plik->Flush();
stru_plik->Close();
resp_stream->Close();
resp->Close();
}

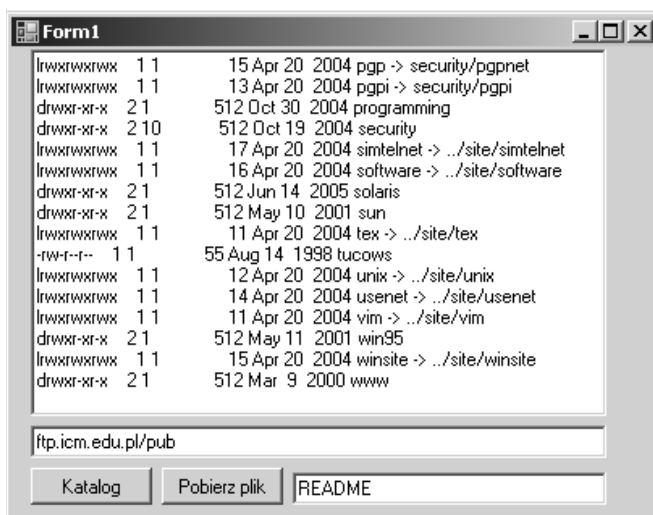
```

Powyższa metoda działa przy założeniu, że pole tekstowe `textBox1` służy do wpisywania adresu FTP, `textBox2` do wyświetlania zawartości katalogu, a `textBox3` do wpisywania nazwy pliku do pobrania.

Po uruchomieniu aplikacji najpierw należy wpisać adres serwera wraz z folderem, w którym znajduje się plik, a następnie nacisnąć przycisk *Katalog*. Po wyświetleniu listy plików sprawdź, czy plik znajduje się na tej liście, a następnie wpisz w polu tekstowym jego pełną nazwę i naciśnij przycisk *Pobierz plik*. Po pobraniu plik będzie się znajdował w folderze roboczym aplikacji. Wygląd aplikacji przedstawia rysunek 12.5.

Rysunek 12.5.

*Aplikacja
do pobierania plików*



W przypadku pobierania dłuższych plików metoda pobierania powinna uruchamiać się w osobnym wątku. W części z przykładami znajduje się klient korzystający z oddzielnych wątków.

Wysyłanie pliku na serwer FTP

Czas na zaprogramowanie możliwości przesyłania pliku na serwer.

Przykład 12.9

Dołącz do aplikacji z poprzedniego przykładu możliwość przesyłania plików z dysku lokalnego na serwer.

Rozwiązanie

Otwórz aplikację z poprzedniego przykładu i umieść trzeci przycisk Button oraz komponent systemowego okna otwarcia pliku OpenFileDialog. We właściwości Text trzeciego przycisku wpisz Wyślij plik.

Po naciśnięciu przycisku powinno się pojawić standardowe okno wyboru pliku, w którym będzie można wskazać plik do wysłania.

Wysyłanie pliku zrealizujemy w metodzie obsługującej zdarzenie Click przycisku.

```
private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e) {
    if (openFileDialog1->ShowDialog()==
        System::Windows::Forms::DialogResult::OK) {
        if (openFileDialog1->FileName!="")
            return;
        Uri^ adres =
            gcnew Uri("ftp://" + textBox1->Text + "/" +
                Path::GetFileName(openFileDialog1->FileName));
        FtpWebRequest^ req =
            dynamic_cast<FtpWebRequest^>(WebRequest::Create(adres));
        req->Credentials=gcnew
            NetworkCredential("anonymous","moja_nazwa@moj_adres.pl");
        req->Method=WebRequestMethods::Ftp::UploadFile;
        FileStream^ stru_plik =
            gcnew FileStream(openFileDialog1->FileName, FileMode::Open);
        Stream^ req_stream = req->GetRequestStream();
        int ile_bajtow;
        array<Byte> ^ bufor = gcnew array<Byte>(1024);
        do {
            ile_bajtow=stru_plik->Read(bufor,0,1024);
            req_stream->Write(bufor,0,ile_bajtow);
        } while(ile_bajtow!=0);
        req_stream->Close();
        MessageBox::Show( "Plik wysłany",
            "Potwierdzenie",MessageBoxButtons::OK,
            MessageBoxIcon::Information);
    }
}
```

Po uruchomieniu programu należy wpisać adres serwera wraz z katalogiem, w którym chcemy umieścić plik (na przykład *ftp.mojftp.net/upload*), a następnie najlepiej dla sprawdzenia pobrać spis plików za pomocą przycisku *Katalog*. Teraz naciśnij przycisk

Wyślij plik, wybierz plik w oknie dialogowym i kliknij *OK* — plik zostanie wysłany na serwer. Po wysłaniu możesz znowu pobrać zawartość katalogu, aby sprawdzić, czy plik został wysłany.



Wskazówka

Aby program działał, serwer musi zezwalać na przyjmowanie plików. Jeżeli chcesz się zalogować na prywatny serwer, musisz podać swój login i hasło we właściwości *Credentials*.



Wskazówka

Wpisywanie swojego loginu i hasła na stałe do programu jest niebezpieczne — raczej należy dodać możliwość wpisywania tych danych do pól tekstowych, skąd będą za każdym razem pobierane.