helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

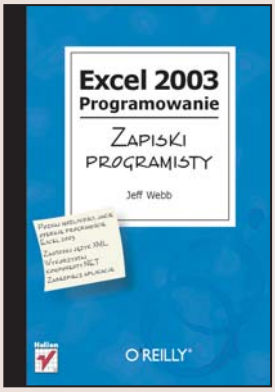
KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

Excel 2003. Programowanie. Zapiski programisty



Autor: Jeff Webb
Tłumaczenie: Radosław Meryk
ISBN: 83-246-0248-8
Tytuł oryginału: Excel 2003 Programming: A Developers Notebook
Format: B5, stron: 352

Poznaj możliwości, jakie oferuje programiście Excel 2003

- Zastosuj język XML
- Wykorzystaj komponenty .NET
- Zabezpiecz aplikacje

Arkusz kalkulacyjny Excel jest coraz częściej wykorzystywany nie tylko do tworzenia zestawień i wykresów, ale również jako narzędzie programistyczne. Za jego pomocą tworzone są zarówno proste aplikacje, jak i złożone systemy operujące na wielu rekordach i połączone z bazami danych. Excel udostępnia twórcom aplikacji mechanizmy pozwalające na stosowanie języka XML, usług sieciowych, bibliotek .NET oraz funkcji Windows API. Dzięki takim możliwościom Excel staje się platformą, w oparciu o którą można stworzyć naprawdę profesjonalne aplikacje.

Książka „Excel 2003. Programowanie. Zapiski programisty" to wzorowana na typowych zeszytach laboratoryjnych publikacja opisująca najciekawsze możliwości Excela, które można wykorzystać, tworząc aplikacje. Nie ma w niej nużących wywodów teoretycznych ani diagramów UML. Znajdziesz w niej natomiast przedstawione w zwartej i zrozumiałej formie praktyczne wiadomości, dzięki którym poznasz prawdziwą potęgę Excela jako narzędzia programistycznego.

- Udostępnianie obszarów roboczych i skoroszytów
- Tworzenie kont użytkowników i dodawanie uprawnień
- Zapisywanie skoroszytów w formacie XML
- Pobieranie danych z sieci
- Stosowanie usług sieciowych
- Łączenie Excela z bibliotekami .NET
- Ochrona i szyfrowanie skoroszytów

Stwórz profesjonalne aplikacje w Excelu

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Seria „Zapiski programisty" **7**

Przedmowa **13**

Rozdział 1. Zaprogramuj nowego Excela **17**

Gdzie są moje dane, koleś? 18
Jak poszczególne części tworzą całość? 19
Wypróbuj to 20
Szybki start — listy i XML 20
Szybki start — SharePoint 25
Szybki start — usługi sieciowe 27
Szybki start — bezpieczeństwo i .NET 30
Szybki start — InfoPath 37
Co dalej? 45

Rozdział 2. Udostępnianie obszarów roboczych i list **47**

Dostęp do usług SharePoint 48
Tworzenie udostępnionego obszaru roboczego 52
Udostępnianie skoroszytu 57
Otwieranie udostępnionego skoroszytu 59
Wyświetlanie witryny SharePoint 61
Anulowanie udostępniania 63

Dodawanie użytkowników i uprawnień 65
Udzielanie dostępu anonimowym użytkownikom 68
Tworzenie listy 74
Udostępnianie listy 77
Aktualizacja udostępnionej listy 80
Wstawianie udostępnionej listy do arkuszy 81
Usuwanie lub odłączanie udostępnionej listy 83
Wykorzystanie usługi sieciowej Lists 84

Rozdział 3. Praca z XML 91

Mówimy językiem XML 92
Różne historie na temat XML 94
Zapisywanie skoroszytów w formacie XML 95
Transformacje arkuszy XML 99
Przekształcanie plików XML na arkusze 107
Zastosowanie map XML 113
Eksportowanie z wykorzystaniem map XML 120
Wykorzystanie szablonów w XML 129
Reakcje na zdarzenia XML 133
Programowanie z wykorzystaniem map XML 134
Pobieranie mapy XML na podstawie listy lub zakresu 138

Rozdział 4. Pobieranie danych z sieci 141

Wykonywanie kwerend sieciowych 142
Modyfikacje kwerend sieciowych 147
Wykonywanie okresowych aktualizacji 149
Zarządzanie kwerendami sieciowymi 153
Korzystanie z usług sieciowych 155
Wykorzystanie pakietu Web Services Toolkit 157
Korzystanie z usług sieciowych za pomocą XML 162
Asynchroniczne wywoływanie usług sieciowych 165
Modyfikacje formatu wyników XML dla Excela 167

Rozdział 5. Programowanie Excela za pomocą pakietu .NET 169

Praca z pakietem .NET 170
Tworzenie komponentów .NET 172
Wykorzystanie komponentów .NET 174
Obsługa błędów i zdarzeń pochodzących od pakietu .NET 178
Debugowanie komponentów .NET 181
Dystrybucja komponentów .NET 183
Wykorzystanie Excela jako komponentu w pakiecie .NET 186
Praca z obiektami Excela w pakiecie .NET 189
Obsługa zdarzeń Excela w .NET 191
Obsługa błędów Excela w .NET 192
Dystrybucja aplikacji .NET, które wykorzystują Excela 196
Tworzenie aplikacji .NET wykorzystujących Excela 200
Ustawianie zasad zabezpieczeń w środowisku .NET 203
Obsługa zdarzeń w aplikacjach .NET 205
Debugowanie aplikacji .NET Excela 208
Wyświetlanie formularzy Windows 211
Dystrybucja aplikacji .NET Excela 214
Dystrybucja dokumentów Excela wykorzystujących kod .NET 217
Migracja do środowiska .NET 218

Rozdział 6. Bezpieczeństwo 223

Warstwy zabezpieczeń 224
Wykorzystanie zabezpieczeń Windows 226
Ochrona hasłem i szyfrowanie skoroszytów 230
Programowanie z wykorzystaniem haseł i szyfrowania 234
Zabezpieczenia elementów skoroszytu 237
Programowanie z zabezpieczeniami 242
Wykorzystanie zabezpieczeń bazujących na tożsamości (IRM) 247
Programowanie z wykorzystaniem uprawnień 253
Wykorzystanie podpisów cyfrowych 255

Ustawienia zabezpieczeń makr 261
Dystrybucja ustawień zabezpieczeń 265
Często zadawane pytania 270

Rozdział 7. Tworzenie formularzy InfoPath 275

Czy stosowanie formularzy InfoPath to dobre rozwiązanie? 276
InfoPath i Excel 287
Udostępnianie danych 290
Łączenie formularza z bazą danych 294
Wypełnianie elementów sterujących ze źródła danych 301
Sprawdzanie poprawności danych 307
Tworzenie skryptów w InfoPath 311
Programowanie InfoPath w środowisku .NET 321
Generowanie HTML 329
Zabezpieczanie się przed wprowadzaniem modyfikacji w projekcie 331

Skorowidz 335

ROZDZIAŁ 4.

Pobieranie danych z sieci

W tym rozdziale:

- ✓ Wykonywanie kwerend sieciowych
- ✓ Modyfikacje kwerend sieciowych
- ✓ Wykonywanie okresowych aktualizacji
- ✓ Zarządzanie kwerendami sieciowymi
- ✓ Korzystanie z usług sieciowych
- ✓ Wykorzystanie pakietu *Web Services Toolkit*
- ✓ Korzystanie z usług sieciowych za pomocą XML
- ✓ Asynchroniczne wywoływanie usług sieciowych
- ✓ Modyfikacje formatu wyników XML dla Excela

Dziś trudno sobie przypomnieć czasy, w których sieciowa pajęczyna (ang. *Web*) nie miała znaczenia, a przecież jeszcze niedawno w ogóle nie istniała. Excel powstał na długo przed jej upowszechnieniem się, dlatego należało go przystosować do nowych warunków. Obecnie wykorzystuje się trzy sposoby pobierania danych z sieci:

- *Kwerendy sieciowe* (ang. *Web queries*) — umożliwiają importowanie danych bezpośrednio ze stron WWW i umieszczanie ich w tabelach zapytań w arkuszu Excela. Chociaż była to jedna z pierwszych własności dostępu do sieci wprowadzonych w Excelu (dodano ją w 1997 roku), w dalszym ciągu jest bardzo przydatna.

- *Usługi sieciowe* (ang. *Web services*) — zdalne wykonywanie aplikacji w sieci w celu uzyskania wyników w formacie XML. Liczba usług dostępnych w sieci szybko rośnie, ponieważ standard ten jest coraz częściej stosowany. Usługi sieciowe to standardowy sposób wymiany parametrów i pobierania wyników w sieci — mechanizm, którego brakuje w zapytaniach sieciowych.
- *Dostęp do bazy danych przez sieć* — obecnie oferowany przez większość systemów baz danych. Technika ta zależy od dostawcy systemu bazy danych. W tej książce nie została opisana.

W tym rozdziale przedstawiłem sposób wykorzystania zapytań sieciowych i usług sieciowych w celu pobierania danych z sieci i importowania ich do Excela. Zaprezentowane przykłady demonstrują różne zadania programistyczne związane z obiema technikami, między innymi:

Kod wykorzystywany w tym rozdziale oraz dodatkowe przykłady są dostępne w pliku r04.xls.

- przekazywanie parametrów,
- formatowanie wyników,
- asynchroniczne pobieranie danych,
- wyświetlanie wyników za pomocą map XML.

Wykonywanie kwerend sieciowych

Choć kwerendy sieciowe nie stanowią niczego nowego, w dalszym ciągu jest to przydatna technika pobierania danych z sieci. Nauczenie się ich zastosowań (a także ograniczeń) ułatwia zrozumienie techniki alternatywnej — usług sieciowych.

Kwerendy sieciowe to szybki sposób importowania danych ze stron WWW do arkuszy z wykorzystaniem obiektu `QueryTable`.

Jak to zrobić?

Aby wykonać kwerendę sieciową:

1. Wybierz polecenie *Dane/Importuj dane zewnętrzne/Nowa kwerenda sieci Web*. Excel wyświetli okno dialogowe *Nowa kwerenda sieci Web* (rysunek 4.1).
2. W pasku adresu wpisz adres strony WWW, z której chcesz zaimportować dane, i kliknij *Przejdź*, aby przejść do tej strony. Najłatwiej znaleźć żądaną stronę w przeglądarce, a następnie wyciąć i wkleić ten adres do okna dialogowego *Nowa kwerenda sieci Web*.



Rysunek 4.1. Wykorzystanie kwerend sieciowych do importowania danych bezpośrednio ze strony WWW

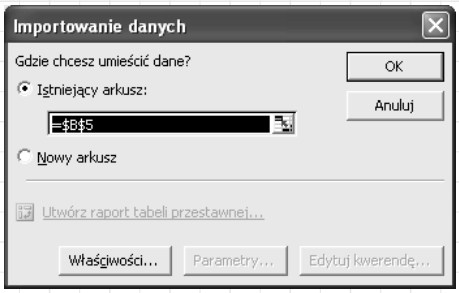
3. Excel umieszcza niewielkie żółte pola obok elementów, które można importować ze strony. Kliknięcie żądanego elementu lub elementów zamienia żółte pole na zielony znak zaznaczenia.

4. Kliknij przycisk *Opcje*, aby ustawić sposób formatowania importowanych elementów. Opcje formatowania pokazano na rysunku 4.2.



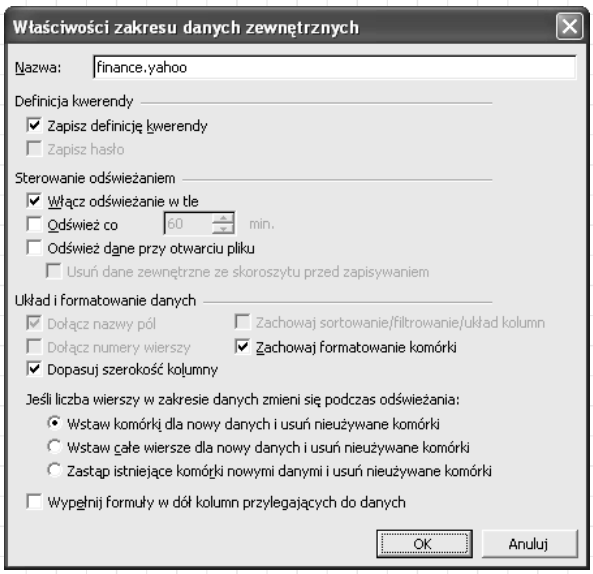
Rysunek 4.2. Ustawianie opcji formatowania dla kwerendy

5. Zamknij okno dialogowe *Opcje* i kliknij *Importuj*. Excel wyświetli okno dialogowe *Importowanie danych*, tak jak widać na rysunku 4.3.



Rysunek 4.3. Wybór miejsca docelowego importowanych danych

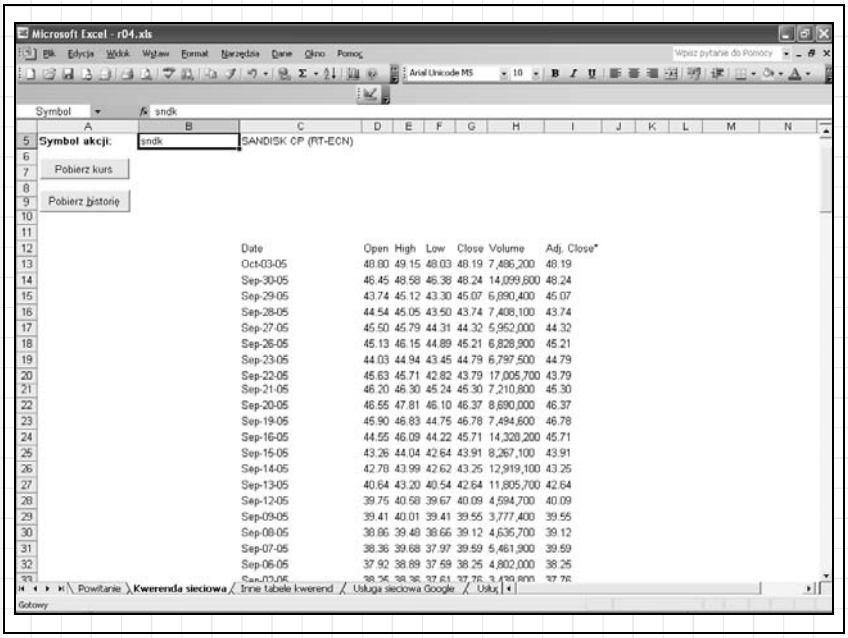
6. Kliknij *Właściwości* w celu określenia sposobu wykonywania kwerendy (np. sposób odświeżania danych). Ustawienia właściwości kwerendy zaprezentowano na rysunku 4.4.



Rysunek 4.4. Właściwości kwerendy umożliwiają ustawienie jej nazwy, sposobu odświeżania danych i wstawiania komórek

7. Zamknij okno dialogowe *Właściwości* i kliknij *OK*, aby zaimportować dane.

Na rysunku 4.5. pokazano dane o akcjach zaimportowane z witryny WWW Yahoo!. Serwis Yahoo! to dobre źródło dla kwerendy sieciowej, ponieważ jest to usługa darmowa, która nie wymaga ani rejestracji, ani logowania.



Rysunek 4.5. Zastosowanie kwerendy sieciowej do pobrania danych o cenie akcji

Jak to działa?

Jeśli wybierzemy polecenie *Narzędzia/Makro/Zarejestruj Nowe Makro*, a następnie wykonamy opisaną powyżej kwerendę sieciową, uzyskamy kod w następującej postaci:

```
With ActiveSheet.QueryTables.Add(Connection:= _
        "URL;http://finance.yahoo.com/q/ecn?s=SNDK", _
         Destination:=Range("C2"))
        .Name = "Real-Time Quote"
        .FieldNames = True
        .RowNumbers = False
        .FillAdjacentFormulas = False
        .PreserveFormatting = True
        .RefreshOnFileOpen = False
        .BackgroundQuery = True
        .RefreshStyle = xlOverwriteCells
        .SavePassword = False
        .SaveData = True
        .AdjustColumnWidth = True
```

```
        .RefreshPeriod = 0
        .WebSelectionType = xlSpecifiedTables
        .WebFormatting = xlWebFormattingNone
        .WebTables = "22"
        .WebPreFormattedTextToColumns = True
        .WebConsecutiveDelimitersAsOne = True
        .WebSingleBlockTextImport = False
        .WebDisableDateRecognition = False
        .WebDisableRedirections = False
        .Refresh BackgroundQuery:=False
    End With

    With ActiveSheet.QueryTables.Add(Connection:= _
        "URL;http://finance.yahoo.com/q/hp?a=01&b=5&c=2003" & _
         "&d=01&e=5&f=2004&g=d&s=sndk", _
         Destination:=Range("A9"))
        .Name = "Price History"
        .FieldNames = True
        .RowNumbers = False
        .FillAdjacentFormulas = False
        .PreserveFormatting = True
        .RefreshOnFileOpen = False
        .BackgroundQuery = True
        .RefreshStyle = xlOverwriteCells
        .SavePassword = False
        .SaveData = True
        .AdjustColumnWidth = True
        .RefreshPeriod = 0
        .WebSelectionType = xlSpecifiedTables
        .WebFormatting = xlWebFormattingNone
        .WebTables = "30"
        .WebPreFormattedTextToColumns = True
        .WebConsecutiveDelimitersAsOne = True
        .WebSingleBlockTextImport = False
        .WebDisableDateRecognition = False
        .WebDisableRedirections = False
        .Refresh BackgroundQuery:=False
    End With
```

Niektóre ważniejsze właściwości i metody, które na powyższym listingu wyróżniono pogrubieniem, zostały opisane poniżej:

- Metoda `Add` tworzy kwerendę i dodaje ją do arkusza.
- Właściwość `RefreshStyle` informuje Excela o tym, że istniejące dane mają być nadpisane. Jest to alternatywa dla wstawiania nowych komórek każdorazowo przy odświeżaniu kwerendy.
- Właściwość `WebTables` identyfikuje na stronie te elementy, które mają być zaimportowane. Excel przypisuje indeks do każdego takiego

elementu. Jeśli właściwość `WebSelectionType` zostanie ustawiona na wartość `xlEntirePage`, to można zaimportować jeden lub kilka elementów albo całą stronę.

- Metoda `Refresh` importuje dane do arkusza. Bez tej metody wyniki kwerendy nie mogą być wyświetlone.

Sama kwerenda składa się z właściwości `Connection`, `WebTables` oraz właściwości formatowania. Jeśli zapiszemy kwerendę sieciową do pliku (z rozszerzeniem *.iqy*), dane mają następującą postać:

```
WEB
1
http://finance.yahoo.com/q/hp?a=01&b=5&c=2003&d=01&e=5&f=2004&g=d&s=sndk

Selection=30
Formatting=None
PreFormattedTextToColumns=True
ConsecutiveDelimitersAsOne=True
SingleBlockTextImport=False
DisableDateRecognition=False
DisableRedirections=False
```

Kiedy Excel uaktualnia kwerendę sieciową, w pasku stanu u dołu ekranu wyświetla się niewielki, zielony glob (rysunek 4.6). Ten symbol wskazuje, że kwerenda jest odświeżana z internetu.



Rysunek 4.6. Excel odświeża zapytanie z internetu

Modyfikacje kwerend sieciowych

Kwerendy sieciowe można modyfikować przez dwukrotne kliknięcie kwerendy i wybranie polecenia *Edytuj kwerendę*. Jednak w wielu przypadkach potrzebny jest sposób bardziej zautomatyzowany. Na przykład w poprzednim przykładzie można było zezwolić użytkownikowi na zmianę symbolu akcji. W tym celu wykorzystamy kod do dwóch działań:

1. Modyfikacji właściwości `Connection` kwerendy.
2. Odświeżania kwerendy.

Rejestrowanie kodu to doskonała metoda nauczenia się sposobów wykonywania określonych operacji przez Excela, ale nic poza tym. Zarejestrowane kwerendy można zmodyfikować po to, by dynamicznie zmienić ciąg kwerendy na podstawie informacji wprowadzonych przez użytkownika.

Jak to zrobić?

Poniższy kod pozwala użytkownikom na wprowadzanie nazwy zakresu w arkuszu w celu uzyskania bieżących i historycznych danych o cenach określonej akcji:

```
Dim ws As Worksheet, qt As QueryTable
Set ws = ThisWorkbook.Sheets("Kwerendy sieciowe")
Set qt = ws.QueryTables("Real-Time Quote")
qt.Connection = "URL;http://finance.yahoo.com/q/ecn?s=" & _
  ws.range("Symbol").Value
qt.Refresh
Set qt = ws.QueryTables("Price History")
qt.Connection =
"URL;http://finance.yahoo.com/q/hp?a=01&b=5&c=2003&d=01&e=5&f=2004&g=d&s=" & _
  ws.Range("Symbol").Value
qt.Refresh
```

Jak to działa?

Jeśli uruchomimy kod zaprezentowany poniżej, zauważymy, że kwerenda nie od razu się zaktualizuje. Domyślnie, zapytania sieciowe wykonywane są w tle w sposób asynchroniczny. Dzięki temu unika się problemu z zajętością Excela w czasie, gdy witryna WWW odpowiada na kwerendę. Niestety, może to doprowadzić do powstania błędu, jeśli kwerenda zostanie odświeżona ponownie, zanim nastąpi reakcja na pierwsze żądanie. Aby tego uniknąć, należy zrezygnować z wykonywania kwerend w tle. Na przykład zaprezentowany poniżej kod wyłącza asynchroniczne kwerendy — przed wykonaniem kolejnej instrukcji Excel będzie czekał na odpowiedź:

```
qt.BackgroundQuery = False
qt.Refresh
```

lub prościej:

```
qt.Refresh False
```

Dzięki wprowadzeniu tej instrukcji Excel będzie oczekiwać zakończenia kwerendy. W tym czasie użytkownik nie może modyfikować komórek ani wykonywać innych zadań. Jeśli stanowi to zbyt duże utrudnienie, w celu uniknięcia asynchronicznych kolizji można wykorzystać właściwość `Refreshing` obiektu `QueryTable`:

```
Set qt = ws.QueryTables("Real-Time Quote")
If Not qt.Refreshing Then
    qt.Connection = "URL;http://finance.yahoo.com/q/ecn?s=" & _
        ws.[Symbol].Value
    qt.Refresh
Else
   MsgBox "Podobna kwerenda nie została jeszcze obsłużona, poczekaj chwilę
   i spróbuj ponownie."
End If
```

W powyższym kodzie przed wywołaniem metody `Refresh` następuje sprawdzenie, czy kwerenda sieciowa nie zaczęła się wykonywać wcześniej. Jeśli poprzednia kwerenda w dalszym ciągu się wykonuje, użytkownik otrzymuje komunikat z prośbą, by spróbował ponownie później. Warto zwrócić uwagę, że kod sprawdza status kwerendy wykonywanej przez pojedynczą tabelę kwerendy. Inne tabele kwerendy mogą czekać na zaległe wyniki i nie powodować kolizji — przed próbą modyfikacji lub odświeżenia kwerendy trzeba sprawdzić właściwość `Refreshing` tylko docelowej tabeli kwerendy.

Wykonywanie okresowych aktualizacji

Jeśli dane w kwerendzie sieciowej zmieniają się często, dobrym rozwiązaniem jest zastosowanie automatycznej, okresowej aktualizacji informacji. Zapytania sieciowe działają asynchronicznie w tle, dlatego spowodowanie, aby aktualizowały się okresowo, jest kwestią ustawienia właściwości:

```
Set qt = ws.QueryTables("Real-Time Quote")
qt.RefreshPeriod = 1
```

Od tej chwili kwerenda będzie aktualizowała się co minutę. W celu wyłączenia aktualizacji kwerendy w tle wystarczy ustawić właściwość `RefreshPeriod` na 0:

```
qt.RefreshPeriod = 0
```

Co ciekawe, mimo że właściwość `BackgroundQuery` ma wartość `False`, w dalszym ciągu można okresowo wykonywać kwerendy. W takim przypadku interfejs użytkownika Excela na czas odświeżania kwerendy okresowo staje się nieaktywny.

Kiedy korzystamy z obiektu w sposób asynchroniczny, dużego znaczenia nabierają zdarzenia. Tabele kwerend nie dodają automatycznie swoich zdarzeń do listy zdarzeń arkusza, tak jak przyciski poleceń lub inne elementy sterujące. Aby dołączyć zdarzenia tabel kwerend, trzeba podjąć specjalne działania.

Wykonywanie kwerend sieciowych w tle może wydawać się nieco dziwne, zwłaszcza jeśli ustawiono ich okresową aktualizację. Większość działań Excela to operacje synchroniczne, więc użytkownicy mogą być zdziwieni, gdy Excel zatrzyma się na chwilę, uaktualni kilka komórek i będzie kontynuował swoje działanie, jakby nic się nie wydarzyło. Taka konfiguracja może stwarzać duży problem wtedy, gdy źródło kwerendy sieciowej zmieni się, przez co wykonanie kwerendy się nie powiedzie — co pewien czas użytkownikowi będzie wyświetlał się komunikat o błędzie, a to może wprowadzić go w zakłopotanie (rysunek 4.7).



Rysunek 4.7. Kwerendy sieciowe, których wykonanie się nie powiedzie, mogą powodować asynchroniczne wyświetlanie komunikatów o błędach

Jak to zrobić?

Do obsłużenia błędów generowanych przez asynchroniczne kwerendy sieciowe powinno się napisać procedury obsługi zdarzeń obiektu `QueryTable`. Aby można było śledzić te zdarzenia, zmienną obiektową `QueryTable` należy zadeklarować z kwalifikatorem `WithEvents`. Można go zastosować tylko w module klasy albo w module obiektów Excela (na przykład module kodu dla arkusza lub skoroszytu).

Na przykład w celu obsługi zdarzeń asynchronicznych dla obiektu `QueryTable` w module arkusza `wsWebQuery` wykonaj następujące czynności:

1. Wyświetl okno kodu dla arkusza przez dwukrotne kliknięcie arkusza `wsWebQuery` w oknie *Visual Studio Project Explorer*.
2. Dodaj następującą deklarację do modułu kodu arkusza na poziomie klasy (poza definicją procedury):

```
Dim WithEvents qt As QueryTable
```

3. Aby utworzyć puste procedury obsługi zdarzeń, wybierz obiekt `qt` na liście obiektów, w górnej części okna kodu, a następnie z listy zdarzeń wybierz `AfterRefresh`.

4. Wprowadź poniższy kod w celu wyłączenia (włączenia) przycisków poleceń i uzyskania od użytkownika informacji w razie wystąpienia błędu:

```
Private Sub qt_BeforeRefresh(Cancel As Boolean)
    ' Wyłączenie przycisku polecenia.
    cmdQuote.Enabled = False
End Sub

Private Sub qt_AfterRefresh(ByVal Success As Boolean)
    ' Pobranie informacji od użytkownika, jeśli aktualizacja się nie
    ' powiedzie.
    If Not Success Then
        If MsgBox("Wystąpił błąd pobierania danych z sieci " & _
         "Anulować kolejne aktualizacje?", vbYesNo, "Web Query") = vbYes
          Then _
          qt.RefreshPeriod = 0
    End If
    ' Ponowne włączenie przycisku polecenia.
    cmdQuote.Enabled = True
End Sub
```

5. Napisz kod inicjujący obiekt `QueryTable` i rozpoczynający aktualizacje. Poniższa procedura wiąże istniejący obiekt `QueryTable` ze zdefiniowanymi wcześniej procedurami obsługi zdarzeń i ustawia symbol akcji wykorzystywany w kwerendzie:

```
Private Sub cmdQuote_Click()
    ' Pobranie obiektu QueryTable i powiązanie go z obiektem obsługi
    ' zdarzeń.
    Set qt = ActiveSheet.QueryTables("Real-Time Quote")
    ' Ustawienie kwerendy.
    qt.Connection = "URL;http://finance.yahoo.com/q/ecn?s=" &
    [Symbol].Value
    ' Ustawienie okresu odświeżania oraz właściwości odświeżania
    ' asynchronicznego.
    qt.RefreshPeriod = 1
    qt.BackgroundQuery = True
    ' Odświeżenie danych.
    qt.Refresh
End Sub
```

Teraz, jeśli wykonanie kwerendy nie powiedzie się, użytkownik będzie mógł wstrzymać automatyczne aktualizacje.

Dziwne, ale prawdziwe

W Excelu zdarzenie asynchroniczne może wystąpić w czasie, kiedy użytkownik modyfikuje kod w edytorze Visual Basica. W takiej sytuacji często pojawia się błąd wykonania spowodowany tym, że aktualnie pisana procedura jest niekompletna. W związku z tym warto zatrzymać okresowe aktualizacje na czas modyfikacji kodu obsługi zdarzeń tabeli kwerend. Można to zrobić, ustawiając w oknie *Immediate* właściwość `RefreshPeriod` tabeli kwerendy na wartość 0.

Jak to działa?

Przewidywanie potencjalnych kolizji zdarzeń asynchronicznych jest dość trudne. Zazwyczaj w celu obsługi takich zdarzeń blokuje się inne operacje w procedurze obsługi `BeforeRefresh` i ponownie się je uaktywnia w procedurze `AfterRefresh` (przez wyłączenie i włączenie przycisku polecenia, tak jak opisano w kroku 4.). Takie działanie uniemożliwia modyfikowanie kwerendy w czasie jej wykonywania. Inny sposób polega na sprawdzaniu właściwości `Refreshing` (zaprezentowano go wcześniej). Jeszcze inne rozwiązanie to całkowita rezygnacja z wykorzystywania kwerend asynchronicznych.

Na przykład poniższy kod pobiera historię cen akcji. Ponieważ dane historyczne nie są zbyt ulotne, kod wykonuje kwerendę synchronicznie i czeka na wyniki:

```
' Wyświetla jeden rok historii cen akcji bieżącego symbolu.
Private Sub cmdHistory_Click()
    Dim ws As Worksheet, qt2 As QueryTable, conn As String
    Set ws = ThisWorkbook.ActiveSheet
    ' Utworzenie ciągu kwerendy.
    conn = "URL;http://chart.yahoo.com/d?" & _
    YahooDates(Date - 365, Date) & ws.[Symbol].Value
    ' Pobranie kwerendy.
    Set qt2 = ws.QueryTables("Price History_1")
    ' Wyzerowanie starej historii.
    qt2.ResultRange.Clear
    ' Ustawienie właściwości połączenia.
    qt2.Connection = conn
    ' Wyłączenie wykonywania kwerend w tle.
    qt2.BackgroundQuery = False
    ' Odświeżenie danych.
    qt2.Refresh
End Sub
```

```
' Przekształca daty początkową i końcową na ciąg kwerend Yahoo
' dla historii cen akcji.
Function YahooDates(dtstart As Date, dtend As Date) As String
    ' Przykładowy ciąg kwerendy Yahoo ma następującą postać:
    ' a=10&b=4&c=2003&d=1&e=5&f=2004&g=d&s=sndk
    Dim str As String
    str = "a=" & Month(dtstart) - 1 & "&b=" & Day(dtstart) & _
    "&c=" & Year(dtstart) & "&d=" & Month(dtend) - 1 & _
    "&e=" & Day(dtend) & "&f=" & Year(dtend) & "&g=d&s="
    Debug.Print str
    YahooDates = str
End Function
```

Uruchomienie powyższego kodu powoduje, że Excel zamienia wskaźnik myszy na symbol oczekiwania i nie obsługuje działań użytkownika do czasu zakończenia wykonywania kwerendy. Dzięki temu ścieżka logiczna programowania jest znacznie prostsza.

Zarządzanie kwerendami sieciowymi

W większości przykładów przedstawionych wcześniej w tym rozdziale pobieraliśmy obiekt `QueryTable`, modyfikowaliśmy jego właściwości, a następnie wywoływaliśmy metodę `Refresh`. Równie dobrze można było wykorzystać metodę `Add` kolekcji `QueryTables` i utworzyć te kwerendy „w locie". W takim przypadku należy jednak pamiętać, aby usunąć wcześniej utworzoną kolekcję `QueryTables`.

Pozbywanie się niepotrzebnych tabel kwerend w arkuszu z pozoru przypomina zbędne porządki, ale w rzeczywistości jest bardzo ważne, ponieważ pozwala uniknąć redundantnych i niepotrzebnych kwerend działających w tle. Kwerendy działające w tle obniżają wydajność, powodują spontaniczne połączenia z internetem oraz, jak wspomniałem wcześniej, mogą być przyczyną asynchronicznych błędów. To naprawdę może wprowadzić użytkowników w błąd!

Jak to zrobić?

Poniższy kod tworzy trzy nowe tabele kwerend w aktywnym arkuszu:

```
Dim ws As Worksheet, qt As QueryTable, i As Integer
Set ws = ActiveSheet
For i = 1 To 3
    Set qt = ws.QueryTables.Add("URL;http://finance.yahoo.com/
    q/ecn?s=yhoo", [A12])
```

```
            qt.Name = "Tymczasowa kwerenda"
            qt.WebTables = "22"
            qt.WebSelectionType = xlSpecifiedTables
            qt.WebFormatting = xlWebFormattingNone
            qt.BackgroundQuery = False
            qt.RefreshStyle = xlOverwriteCells
            qt.Refresh
        Next
```

Uruchomienie tego kodu powoduje utworzenie trzech tabel kwerend odpowiednio w arkuszach *Tymczasowa_kwerenda*, *Tymczasowa_kwerenda_1* i *Tymczasowa_kwerenda_2*. Nie istnieje łatwy sposób zarządzania tabelami kwerend za pomocą interfejsu użytkownika Excela, ale wciśnięcie klawiszy *Ctrl+G* powoduje wyświetlenie nazw nowych tabel kwerend w oknie dialogowym *Przechodzenie do* (rysunek 4.8).



Rysunek 4.8. Excel automatycznie numeruje tabele kwerend w przypadku identycznych nazw bazowych

Tabele kwerend można usuwać ręcznie. Wystarczy wejść do zakresu z przypisaną nazwą i wybrać opcję *Edycja/Wyczyść/Wszystko*. Takie działanie pozostawia jednak nazwę w arkuszu i kolejne nazwy są indeksowane _4, _5 itd. Najprostszym sposobem na usunięcie omyłkowo nazwanych bądź próbnych tabel kwerend jest napisanie odpowiedniego kodu. Poniższa procedura wyświetla wszystkie tabele kwerend w arkuszu i pozwala na ich usunięcie bądź pozostawienie:

```
Sub RemoveOldQueries()
    Dim ws As Worksheet, qt As QueryTable, nm As Name
    Set ws = ActiveSheet
    For Each qt In ws.QueryTables
```

```
        If MsgBox("Czy usunąć " & qt.Name & "?", vbYesNo, _
        "Kwerendy sieciowe") = vbYes Then
            qt.Delete
        End If
    Next
    For Each nm In ws.Names
        If MsgBox("Czy usunąć " & nm.Name & "?", vbYesNo, _
        "Nazwy") = vbYes Then
            nm.Delete
        End If
    Next
End Sub
```

Korzystanie z usług sieciowych

Z perspektywy Excela usługi sieciowe przydają się przede wszystkim do pobierania zmieniających się danych z internetu, ale można także wykorzystać je do wysyłania danych, przetwarzania zdalnych danych oraz uruchamiania innego kodu w zdalnych komputerach. Usługi sieciowe zaprojektowano w taki sposób, by z kodu można się nimi posługiwać tak, jak wywołaniami procedur. Może się zatem zdarzyć, że ktoś korzysta z nich i nie wie nawet, że uruchamia zdalny kod.

To możliwe, ale niezbyt prawdopodobne, ponieważ metody usług sieciowych często bazują na XML. Oznacza to, że programiści Excela, zanim efektywnie skorzystają z usług sieciowych, muszą poznać bibliotekę Microsoft XML. Jednak nauka się opłaca. Za pomocą XML można, wykorzystując mapy XML Excela, importować wyniki usług sieciowych bezpośrednio do list arkuszy (co jest bardzo przydatne).

Kwerendy a usługi sieciowe

Kwerendy sieciowe sprawdzają się doskonale w przypadku okazyjnych operacji importowania danych do arkusza. Problem polega na tym, że bazują one na pozycji elementów na stronie. Jeśli struktura źródłowej strony WWW zmieni się, wykonanie kwerendy może się nie udać. Oznacza to, że kwerendy sieciowe nie najlepiej nadają się do rozwiązań instalowanych przez użytkowników. Świadczy o tym również duża liczba zgłoszeń z prośbami o pomoc techniczną w przypadku modyfikacji bądź przeniesienia źródłowej strony WWW.

Usługi sieciowe nie mają takich ograniczeń i zazwyczaj oferują lepszy interfejs do pobierania danych z internetu. Usługi sieciowe nie są jednak dostępne dla wszystkich danych w internecie, więc w bardzo wielu przypadkach kwerendy sieciowe są wciąż bardzo przydatne.

Aby wykonywać spersonalizowane kwerendy, trzeba komponować skomplikowane, specyficzne dla witryny właściwości `Connection` (ciągi zapytania). Każda witryna WWW ma własny system wysyłania i odbierania danych za pomocą tych ciągów. Prawidłowe ich rozpoznanie przy użyciu techniki wstecznej inżynierii może być bardzo trudne.

Jak to działa?

Kiedy Excel wywołuje usługę sieciową, wysyła żądanie przez internet pod adres usługi sieciowej, a następnie czeka na odpowiedź. Zarówno żądanie, jak i odpowiedź zazwyczaj mają format danych XML.

Usługi sieciowe, podobnie jak wiele technologii związanych z internetem, należą do zmieniających się standardów. Standardy te mają duże wsparcie ze strony różnych firm, nie ma zatem obaw, że usługi sieciowe w przyszłości przestaną być obsługiwane. Jednak standardy ciągle ewoluują, a to sprawia, że istnieją różne podejścia do implementacji, lokalizacji i dostępu do usług sieciowych. Na szczególną uwagę programistów Excela zasługują następujące fakty:

- Sposoby lokalizowania usług sieciowych w internecie. Jeden z nich bazuje na usługach katalogowych, na przykład *http://uddi.microsoft.com/*, choć znacznie bardziej popularnym sposobem jest przeglądanie witryn tematycznych bądź witryn krzyżowych, takich jak *http://www.xmethods.net*.
- Sposoby opisywania usług sieciowych w internecie. W przypadku Excela jedyny, który zasługuje na uwagę, to WSDL.
- Sposoby wywoływania usług sieciowych. Niektóre z nich obsługują tylko protokół SOAP, natomiast inne, na przykład Amazon, obsługują dostęp bezpośrednio przez adresy URL.

Przykłady w tym rozdziale skupiają się na dwóch powszechnie wykorzystywanych usługach sieciowych oferowanych odpowiednio przez firmy Google i Amazon.com. Usługi te niemal idealnie nadają się do wykorzystania w takim rozdziale, jak ten, ponieważ są darmowe, użyteczne, dobrze udokumentowane oraz demonstrują użycie zarówno dostępu SOAP, jak i za pośrednictwem adresów URL.

Gdzie można pobrać potrzebne pakiety?

Przed kontynuowaniem lektury należy pobrać następujące zestawy narzędzi:

Zestaw narzędzi	Lokalizacja
Microsoft Office Web Services Toolkit	Wyszukaj frazy „*Web Services Toolkit*" pod adresem *http://www.microsoft.com/downloads*
Usługa sieciowa Google	*http://www.google.com/apis*
Usługa sieciowa Amazon	Kliknij łącze *Web Services* pod adresem *http://www.amazon.com/*

Obie wymienione wcześniej usługi sieciowe wymagają rejestracji w celu uzyskania *identyfikatora programisty* (ang. *developer ID*), który należy przekazać w wywołaniach metod. W przykładach kodu zaprezentowanych w tym rozdziale podałem swój identyfikator, ale jeśli ktoś chce wykorzystywać je we własnym kodzie, musi podać osobisty identyfikator.

Wykorzystanie pakietu Web Services Toolkit

Pakiet Web Services Toolkit umożliwia wyszukiwanie usług sieciowych i odwoływanie się do nich z poziomu Visual Basica. Po utworzeniu odwołania do usługi sieciowej narzędzie generuje klasy, które oferują znany interfejs do kodu XML spodziewanego z usługi sieciowej. Klasy wygenerowane przez narzędzie obsługują odpowiedzi z usługi sieciowej, przekształcając je z surowego XML na obiekty, właściwości i metody.

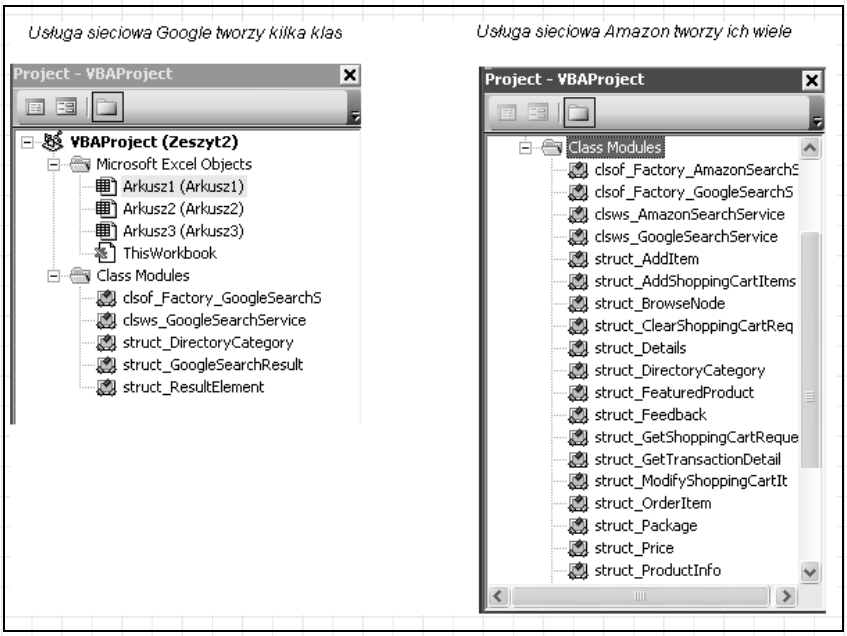
Pakiet Web Services Toolkit nie jest instalowany wraz z pakietem Office 2003. Trzeba go pobrać z witryny Microsoft i zainstalować w swoim komputerze.

W zależności od wykorzystywanej usługi sieciowej pakiet Web Services Toolkit może wygenerować wiele lub tylko kilka nowych klas (rysunek 4.9).

Jak to zrobić?

Aby skorzystać z usług sieciowych w Visual Basicu, najpierw wykonaj następujące czynności:

1. Znajdź pakiet Microsoft Office Web Services Toolkit przez wyszukanie frazy „*Web Services Toolkit*" pod adresem *http://www.microsoft.com/downloads*.



Rysunek 4.9. Pakiet Office Web Services Toolkit tworzy klasy pośredniczące dla wywoływanych usług sieciowych

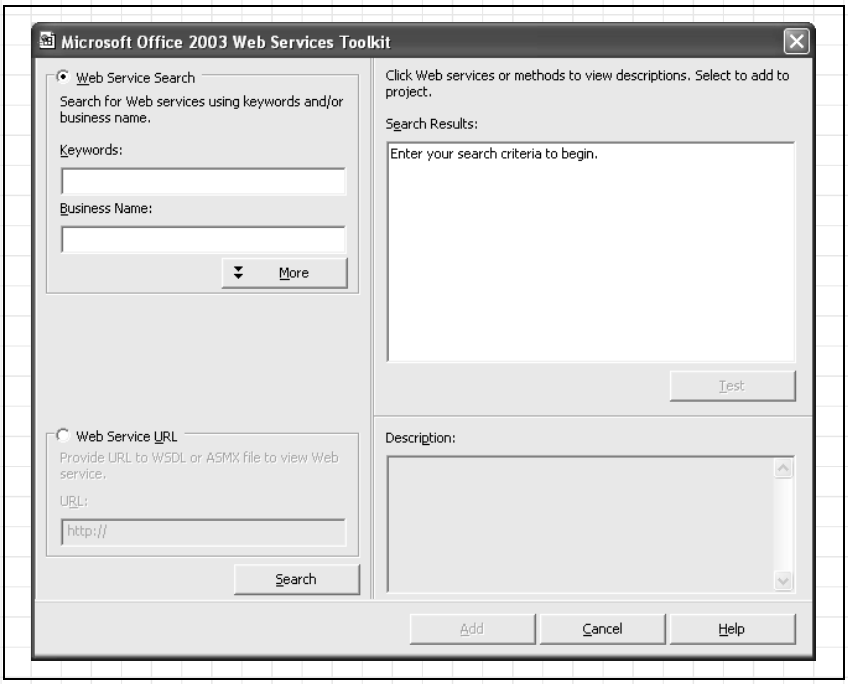
Pakiet Web Services Toolkit ułatwia korzystanie z usług sieciowych dzięki generowaniu klas na podstawie opisu usług. Klasy można następnie wykorzystać w standardowy dla programowania obiektowego sposób do utworzenia egzemplarza usługi sieciowej i wywoływania jej metod oraz właściwości.

2. Pobierz program instalacyjny pakietu Web Services Toolkit (*setup.exe*).
3. Uruchom pobrany program instalacyjny i postępuj zgodnie ze wskazówkami wyświetlanymi przez kreatora instalacji.
4. Uruchom Excela i otwórz edytor Visual Basica.
5. W edytorze Visual Basica z menu *Tools* wybierz polecenie *Web Service References*. Wyświetli się okno dialogowe *Microsoft Office 2003 Web Services Toolkit* (rysunek 4.10).

W trakcie tworzenia odwołania do usługi sieciowej pakiet Web Services Toolkit automatycznie dodaje odwołania do biblioteki Microsoft Office SOAP oraz Microsoft XML. Następnie pakiet generuje klasy pośredniczące dla usługi sieciowej.

Aby przekonać się, jak to działa, wykonaj następujące czynności:

1. Z menu *Tools* edytora Visual Basic wybierz polecenie *Web Service References*.



Rysunek 4.10. Pakiet Microsoft Office 2003 Web Services Toolkit służy do tworzenia odwołań do usług sieciowych

2. Zaznacz opcję *Web Service URL* i wpisz następujący adres w polu tekstowym poniżej tej opcji:

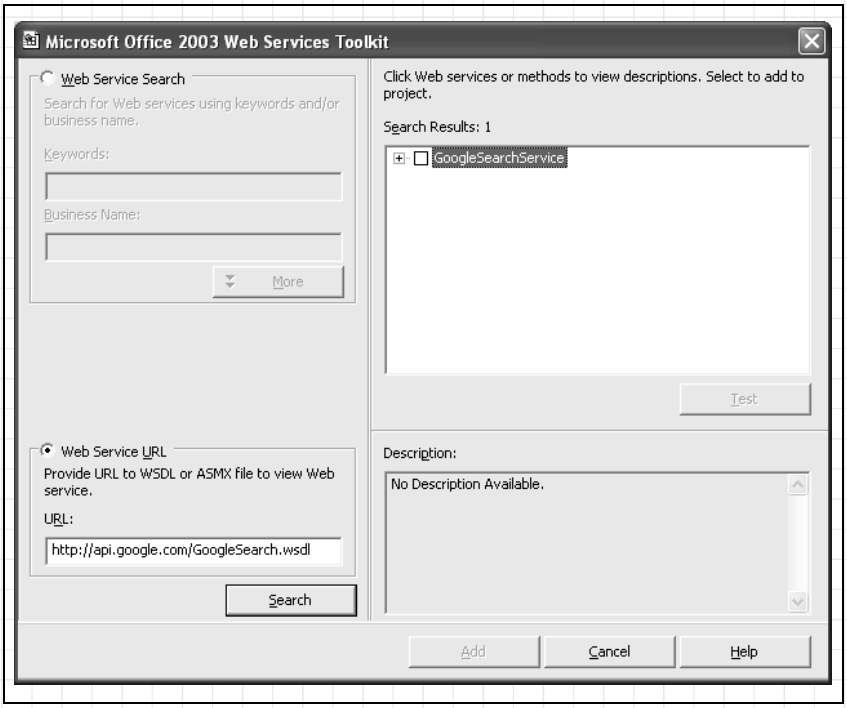
   `http://api.google.com/GoogleSearch.wsdl`[1]

[1] W momencie powstawania oryginału tej książki usługa sieciowa Google bezproblemowo działała z pakietem Office, jednak w okolicach sierpnia 2005 roku format pliku *.wsdl* usługi sieciowej Google zmienił się, co spowodowało pewne problemy z jej wykorzystywaniem. Aby móc korzystać z tej usługi, można zastosować pewne obejście. Oto ono:

1. Należy pobrać plik *GoogleSearch.wsdl* spod adresu *http://api.google.com/GoogleSearch.wsdl* i zapisać na lokalnym dysku.
2. Trzeba wyedytować plik w dowolnym edytorze i w sekcji `<message name = "doGoogleSearch">` zmienić typ wszystkich elementów na `anyType`, na przykład `xsd:string` na `xsd:anyType`. Zapisać plik na lokalnym dysku.
3. W klasie `clsws_GoogleSearchService` (wygenerowanej podczas dodawania odwołania do usługi sieciowej Google) należy podać ścieżkę do lokalnego pliku *.wsdl*, czyli zamiast `Private Const c_WSDL_URL As String =http://api.google.com/GoogleSearch.wsdl` wstawić `Private Const c_WSDL_URL As String = "nasza_lokalizacja\GoogleSearch.wsdl"`.

Po wykonaniu tego prostego zabiegu można bezproblemowo korzystać z usługi sieciowej Google w Excelu — *przyp. tłum.*

3. Kliknij *Search*. Pakiet Web Services Toolkit wyświetli usługi sieciowe dostępne w witrynie Google (rysunek 4.11).

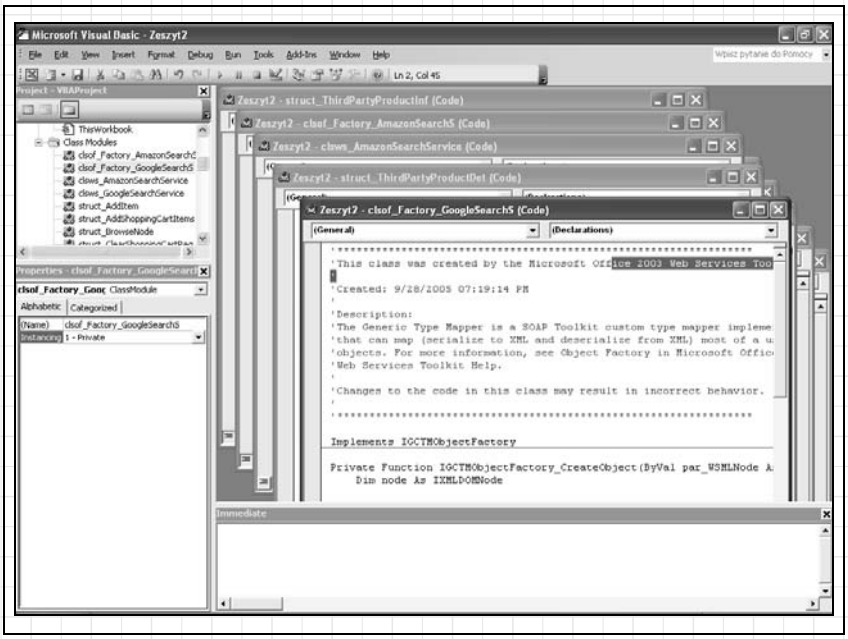


Rysunek 4.11. Tworzenie odwołania do usługi sieciowej Google

4. Zaznacz pole *GoogleSearchService* i kliknij *Add*. Pakiet Web Services Toolkit doda odwołania do bibliotek SOAP i XML oraz utworzy klasy pośredniczące dla każdej z usług (rysunek 4.12).

Jak to działa?

Klasy pośredniczące (ang. *proxy classes*) to moduły kodu, które zastępują kod działający na serwerze udostępniającym usługę sieciową. Lokalna kopia tego kodu jest potrzebna po to, by można było przeprowadzić kompilację. Klasy pośredniczące umożliwiają dostęp do właściwości i metod wywoływanych w usłudze sieciowej — tworzą pakiety wywołań, wysyłają je i odbierają odpowiedzi.



Rysunek 4.12. Pakiet Web Services Toolkit tworzy klasy pośredniczące dla usługi sieciowej Google

Kod klas pośredniczących nie jest prosty. Na szczęście, nie trzeba go zbyt dokładnie rozumieć — wystarczy utworzyć egzemplarz głównej klasy (identyfikowanej prefiksem `clsws`) i skorzystać z jej właściwości oraz metod. W poniższym kodzie wykorzystano wygenerowane klasy do wyszukiwania w witrynie Google prac, które napisałem na temat Excela:

```
Dim I As Integer, wsGoogle As New clsws_GoogleSearchService
Dim wsResult As struct_GoogleSearchResult, wsElement As
struct_ResultElement
Dim devKey As String, searchStr As String
' Ten klucz pochodzi z witryny Google, służy do identyfikacji programisty.
devKey = "ekN14fFQFHK7lXIW3Znm+VXrXI7Focrl"
' Elementy do wyszukania.
searchStr = "Jeff Webb Excel"
' Wywołanie usługi sieciowej wyszukiwania.
Set wsResult = wsGoogle.wsm_doGoogleSearch(devKey, _
 searchStr, 0, 10, False, "", False, "", "", "")
' Dla każdego wyniku.
For i = 0 To wsResult.endIndex - 1
     ' Pobranie indywidualnego wyniku.
     Set wsElement = wsResult.resultElements(i)
     ' Wyświetlenie wyniku.
     Debug.Print wsElement.title, wsElement.URL
Next
```

Istotnie, niezbyt proste. Większość komplikacji pochodzi z samej usługi sieciowej. Google wymaga klucza licencji do korzystania ze swojej usługi. W zmiennej `devKey` umieściłem swój klucz Google, który umożliwia przeprowadzanie 1000 żądań wyszukiwania dziennie, zatem chcąc korzystać z usługi, trzeba się postarać o własny klucz. Początkowo jednak można korzystać z mojego.

Metoda `wsm_doGoogleSearch` wysyła żądanie wyszukiwania do serwisu Google. W tej metodzie konieczne jest podanie mnóstwa argumentów, a w odpowiedzi uzyskuje się strukturę zdefiniowaną w innej klasie pośredniczącej. Z tego powodu wyniki należy przypisać do obiektu `Set`. W podobny sposób pobiera się elementy wyniku do obiektu `Set`.

A co z...

... adresami usług sieciowych Google i Amazon? Zestawiono je w tabeli 4.1. Są to adresy, które wprowadza się w polu *Web Service URL* okna dialogowego *Web Services Toolkit* w celu utworzenia odwołania do tych usług.

Tabela 4.1. Adresy opisu usług sieciowych

Usługa sieciowa	URL
Amazon	*http://soap.amazon.com/schemas3/AmazonWebServices.wsdl*
Google	*http://api.google.com/GoogleSearch.wsdl*

Korzystanie z usług sieciowych za pomocą XML

Aby skorzystać z usług sieciowych, nie trzeba używać pakietu Web Services Toolkit. W niektórych przypadkach nawet łatwiej jest wywołać usługę sieciową bezpośrednio — bez korzystania z wygenerowanych klas pośredniczących.

Usługi sieciowe proponowane przez różne firmy w różny sposób definiują swoje interfejsy. Na przykład usługa sieciowa Google oferuje metody, które pobierają argumenty w postaci ciągów znaków, natomiast w usłudze Amazon trzeba podawać złożone argumenty XMLNodeList.

Konstruowanie i analizowanie argumentów `XMLNodeList` dla usługi sieciowej Amazon jest bardzo trudne. O wiele prościej jest wywołać tę usługę sieciową bezpośrednio za pomocą jej adresu URL i bezpośrednio odebrać odpowiedź w formacie XML.

Jak to zrobić?

Poniższy kod wykonuje wyszukiwanie książek na temat wombatów w witrynie Amazon:

```
Dim SearchUrl As String
' Utworzenie nowego obiektu DOMDocument i ustawienie jego opcji.
Dim xdoc As New DOMDocument
xdoc.async = True
xdoc.preserveWhiteSpace = True
xdoc.validateOnParse = True
xdoc.resolveExternals = False

' Utworzenie żądania wyszukiwania.
SearchUrl = "http://xml.amazon.com/onca/xml2" & _
        "?t=" & "webservices-20" & _
        "&dev-t=" & "D1UCR04XBIF4A6" & _
        "&page=1" & _
        "&f=xml" & _
        "&mode=books" & _
        "&type=lite" & _
        "&KeywordSearch=wombat"

' Wysłanie żądania i czekanie na odpowiedź.
Loaded = xdoc.Load(SearchUrl)
' Wyświetlenie wyników.
Debug.Print xdoc.XML
```

Ponieważ wyniki są zwracane w formacie XML, możemy na ich podstawie utworzyć mapę XML i zaimportować wyniki na listę, tak jak widać poniżej:

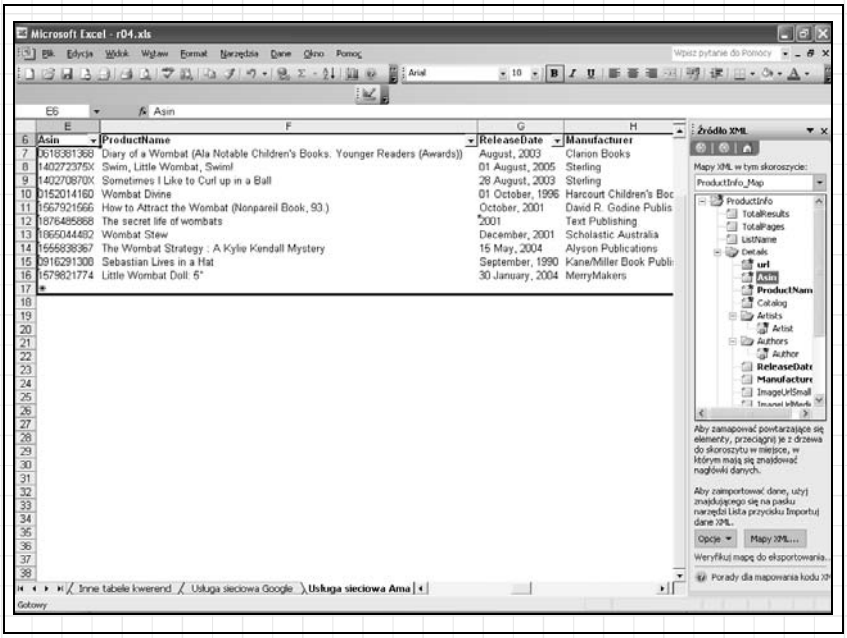
```
Set wb = ThisWorkbook
wb.XmlImportXml doc..XML, wb.XmlMaps("ProductInfo_Mapa"), True
```

Na rysunku 4.13 zaprezentowano wynik importowania wyników wyszukiwania książek o wombatach w usłudze Amazon w postaci listy w arkuszu.

Jak to działa?

W dokumentacji usługi sieciowej Amazon można znaleźć opis wywoływania jej metod za pomocą adresu URL, a nie przy użyciu klas pośredniczących i protokołu SOAP. Oznacza to, że nie trzeba korzystać z pakietu Web Services Toolkit w celu tworzenia klas pośredniczących dla usługi sieciowej Amazon — wystarczy dodać odwołanie do biblioteki *Microsoft XML*.

Tę metodę dostępu do usług sieciowych czasami określa się jako REST (*Representational State Transfer*). Akronim przydaje się jako kryterium wyszukiwania podczas szukania tego typu interfejsu dla określonej usługi



Asin	ProductName	ReleaseDate	Manufacturer
0618381368	Diary of a Wombat (Ala Notable Children's Books. Younger Readers (Awards))	August, 2003	Clarion Books
140272375X	Swim, Little Wombat, Swim!	01 August, 2005	Sterling
140270870X	Sometimes I Like to Curl up in a Ball	28 August, 2003	Sterling
0152014160	Wombat Divine	01 October, 1996	Harcourt Children's Boo
1567921666	How to Attract the Wombat (Nonpareil Book, 93.)	October, 2001	David R. Godine Publis
1876485868	The secret life of wombats	2001	Text Publishing
1865044482	Wombat Stew	December, 2001	Scholastic Australia
1555838367	The Wombat Strategy : A Kylie Kendall Mystery	15 May, 2004	Alyson Publications
0916291308	Sebastian Lives in a Hat	September, 1990	Kane/Miller Book Publi
1579821774	Little Wombat Doll: 5"	30 January, 2004	MerryMakers

Rysunek 4.13. Wyświetlanie wyników XML z usługi sieciowej za pomocą mapy XML i listy

sieciowej. Wystarczy na przykład wpisać w wyszukiwarce Google frazę *„REST Google API"*, aby przyjrzeć się debacie na temat powiązanych ze sobą własności protokołów REST i SOAP.

Usługa sieciowa Google nie obsługuje bezpośredniego dostępu za pomocą adresu URL, ale można uniknąć posługiwania się klasami pośredniczącymi i wywoływać ją bezpośrednio przy użyciu protokołu SOAP. Poniższy kod wykonuje wyszukiwanie informacji o wombatach i importuje wyniki za pomocą mapy XML bezpośrednio na listę:

```
Dim soap As New SoapClient30, xn As IXMLDOMNodeList, strXML As String
soap.MSSoapInit "http://api.google.com/GoogleSearch.wsdl"
Set xn = soap.doGoogleSearch("ekN14fFQFHK7lXIW3Znm+VXrXI7Focrl", _
"wombats", 0, 10, False, "", False, "", "", "")
' Utworzenie ciągu znaków zawierającego wyniki wyszukiwania w formacie XML.
strXML = "<GoogleSearchResults>"
For i = 1 To xn.Length - 1
    strXML = strXML & xn(i).XML
Next
strXML = strXML & "</GoogleSearchResults>"
' Import wyników za pomocą mapy XML na listę.
Set wb = ThisWorkbook
wb.XmlImportXml strXML, wb.XmlMaps("GoogleSearchResults_Mapa"), True
```

A co z...

... protokołem REST (*Representational State Transfer*)?	Zajrzyj na stronę *http://rest.blueoxen.net/cgi-bin/wiki.pl*
... dokumentacją MSXML?	Poszukaj frazy „*MSXML Documentation*" pod adresem *http://msdn.microsoft.com*
... obiektem `DOMDocument`?	Poszukaj frazy „`DOMDocument`" pod adresem *http://msdn.microsoft.com/*
... obiektem `IXMLDOMNodeList`?	Poszukaj frazy „`IXMLDOMNodeList`" pod adresem *http://msdn.microsoft.com*

Asynchroniczne wywoływanie usług sieciowych

Nie zawsze chcemy, by użytkownicy bezczynnie czekali na odpowiedź usługi sieciowej. W niektórych przypadkach uzyskanie odpowiedzi od zdalnego komputera zajmuje dłuższą chwilę. W celu rozwiązania tego problemu można wywołać usługę sieciową asynchronicznie — nie można jednak tego zrobić za pomocą klas wygenerowanych przez pakiet Web Services Toolkit!

Zaletą wywoływania usług sieciowych bezpośrednio, bez korzystania z klas pośredniczących, jest łatwa obsługa odpowiedzi w trybie asynchronicznym. Obiekt `DOMDocument` obsługuje zdarzenie `ondataavailable`, które zachodzi w chwili, gdy obiekt zakończy ładowanie danych XML ze źródła. Oznacza to, że można uruchomić żądanie usługi sieciowej, oddać sterowanie nad aplikacją użytkownikowi i wyświetlić wyniki po obsłużeniu żądania. Możliwość asynchronicznej obsługi żądań ma specjalne znaczenie wtedy, gdy usługa sieciowa zwraca dane o dużej objętości.

Jak to zrobić?

Aby skorzystać z obiektu `DOMDocument` do asynchronicznej obsługi usługi sieciowej, wykonaj następujące czynności:

1. Zadeklaruj obiekt `DOMDocument` na poziomie modułu klasy. Klasa może być skoroszytem, arkuszem lub modułem kodu. Na przykład wewnątrz klasy arkusza *wsAmazon* zadeklarowano następującą zmienną:

```
Dim WithEvents xdoc As DOMDocument
```

2. Aby utworzyć pustą procedurę obsługi zdarzenia `ondataavailable` (pokazaną poniżej), wybierz obiekt `xdoc` z listy obiektów w górnej części okna kodu, a następnie zdarzenie `ondataavailable` z listy zdarzeń.

```
Private Sub xdoc_ondataavailable()

End Sub
```

3. W dalszej części kodu zainicjuj obiekt xdoc, ustaw jego właściwość async na wartość True, a następnie wywołaj usługę sieciową, posługując się metodą Load obiektu xdoc. Na przykład po kliknięciu przez użytkownika przycisku *Pobierz tytuły* w arkuszu *Usługa sieciowa Amazon* pokazana poniżej procedura obsługi zdarzenia przeszuka serwis Amazon.com w poszukiwaniu wprowadzonego w arkuszu słowa kluczowego:

```
Sub cmd_Titles_Click()
  Dim SearchUrl As String
  ' Utworzenie nowego obiektu DOMDocument i ustawienie jego opcji.
  Set xdoc = New DOMDocument
  xdoc.async = True
' Utworzenie żądania wyszukiwania.
  SearchUrl = "http://xml.amazon.com/onca/xml2" & _
            "?t=" & "webservices-20" & _
            "&dev-t=" & "D1UCR04XBIF4A6" & _
            "&page=1" & _
            "&f=xml" & _
            "&mode=books" & _
            "&type=lite" & _
            "&KeywordSearch=" & txtSearch.text
' Wysłanie żądania i czekanie na jego obsługę.
  Loaded = xdoc.Load(SearchUrl)
End Sub
```

4. Dopisz do procedury ondataavailable fragment kodu, który reaguje na dane z usługi sieciowej po ich zwróceniu. Na przykład poniższy kod importuje wyniki za pośrednictwem mapy XML i wyświetla je na liście:

```
Private Sub xdoc_ondataavailable()
  Dim wb As Workbook
  ' Zaimportowanie wyników przez mapę XML na listę.
    Set wb = ThisWorkbook
    wb.XmlImportXml xdoc.XML, wb.XmlMaps("ProductInfo_Mapa"), True
End Sub
```

Jak to działa?

Kiedy uruchomimy powyższy kod przez kliknięcie przycisku *Pobierz tytuły*, Excel zwróci sterowanie użytkownikowi natychmiast po kliknięciu przycisku. Lista zostanie natomiast zaktualizowana dopiero w chwili uzyskania odpowiedzi z usługi sieciowej.

Biblioteka *Microsoft SOAP* nie obsługuje wywołań asynchronicznych, dlatego w Excelu nie można korzystać w sposób asynchroniczny z usług sieciowych, które oferują wyłącznie protokół SOAP. Narzędzia SOAP

dostępne z pakietem .NET obsługują jednak wywołania asynchroniczne, więc programując w języku Visual Basic .NET poza Excelem, można korzystać z asynchronicznych wywołań SOAP.

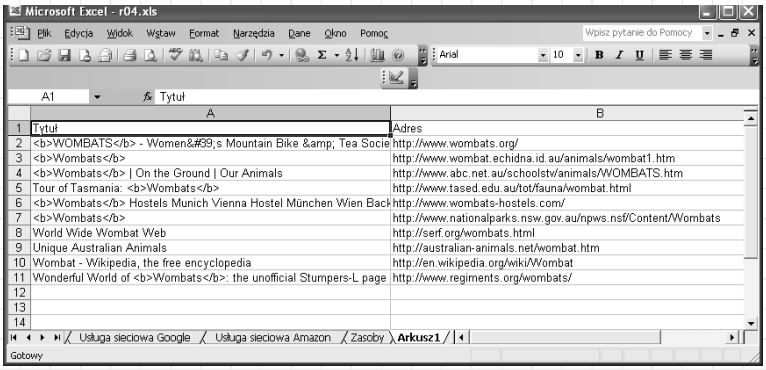
A co z...

... pakietem SOAP Toolkit?	Wyszukaj frazy *„SOAP Toolkit 3.0"* pod adresem *http://www.microsoft.com/downloads*
... tworzeniem komponentów .NET do wykorzystania w Excelu?	Zajrzyj do rozdziału 5. „Programowanie Excela za pomocą pakietu .NET"

Modyfikacje formatu wyników XML dla Excela

Oto jeden z największych problemów napotykanych w trakcie programowania z wykorzystaniem usług sieciowych w Excelu — jeśli wyniki zwracane przez usługę sieciową są sformatowane w dowolny sposób, to formatowanie jest zazwyczaj zdefiniowane przez znaczniki, których Excel nie potrafi automatycznie zinterpretować. Aby formatowanie wyglądało poprawnie, trzeba zastosować pewną sztuczkę.

Łatwo zauważyć, że w przypadku pobierania wyników z usługi sieciowej do Excela za pośrednictwem mapy XML uzyskana treść nie jest automatycznie formatowana. Znaczniki HTML (XML) na przykład `<b>` oraz `<i>` występują w arkuszu jako `<b>` oraz `<i>`, a nie jako pogrubienie i kursywa (rysunek 4.14).



Rysunek 4.14. Excel nie interpretuje automatycznie formatowania HTML

Jak to zrobić?

Choć nie istnieje prosty sposób zapobieżenia temu problemowi, można go rozwiązać dzięki wykorzystaniu własności automatycznego formatowania tekstu w Excelu. Excel automatycznie formatuje tekst HTML wklejany ze

schowka, wystarczy zatem umieścić dane w formacie HTML w schowku, a następnie wkleić je do komórek arkusza.

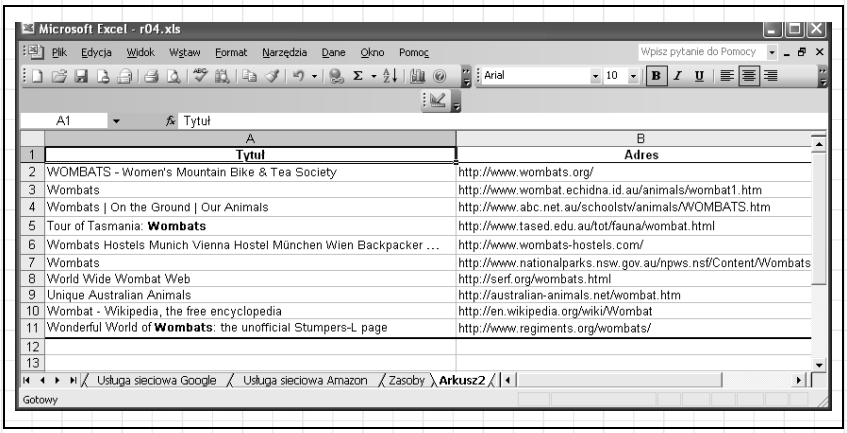
W Excelu dostęp do schowka można uzyskać za pomocą obiektu `DataObject`. Poniższy kod umieszcza dane z komórek arkusza w schowku jako HTML, a następnie wkleja je z powrotem do arkusza. Dzięki temu formatowanie w Excelu jest interpretowane właściwie:

```
Sub TestReformat()
    ' Wywołanie pomocniczej funkcji interpretującej kody formatowania HTML.
    ReformatHTML ActiveSheet.UsedRange
End Sub

Sub ReformatHTML(rng As Range)
    Dim clip As New DataObject, cell As Range
    For Each cell In rng
        clip.SetText "<html>" & cell.Value & "</html>"
        clip.PutInClipboard
        cell.PasteSpecial
    Next
End Sub
```

Jak to działa?

Kiedy w arkuszu uruchomimy procedurę `TestReformat`, Excel właściwie zinterpretuje kody formatowania HTML w analogiczny sposób do operacji wytnij-wklej dla danych ze strony WWW (rysunek 4.15).



	A	B
1	**Tytuł**	**Adres**
2	WOMBATS - Women's Mountain Bike & Tea Society	http://www.wombats.org/
3	Wombats	http://www.wombat.echidna.id.au/animals/wombat1.htm
4	Wombats | On the Ground | Our Animals	http://www.abc.net.au/schoolstv/animals/WOMBATS.htm
5	Tour of Tasmania: **Wombats**	http://www.tased.edu.au/tot/fauna/wombat.html
6	Wombats Hostels Munich Vienna Hostel München Wien Backpacker ...	http://www.wombats-hostels.com/
7	Wombats	http://www.nationalparks.nsw.gov.au/npws.nsf/Content/Wombats
8	World Wide Wombat Web	http://serf.org/wombats.html
9	Unique Australian Animals	http://australian-animals.net/wombat.htm
10	Wombat - Wikipedia, the free encyclopedia	http://en.wikipedia.org/wiki/Wombat
11	Wonderful World of **Wombats**: the unofficial Stumpers-L page	http://www.regiments.org/wombats/
12		
13		

Rysunek 4.15. Formatowanie HTML po uruchomieniu procedury ReformatHTML