kasia szymczak

Własny program do wysyłania SMS-ów przez Internet

Monopol jest OK

Wysyłanie SMS-ów za pośrednictwem bramek WWW/SMS naszych operatorów nie należy do wygodnych rozwiązań, zwłaszcza jeśli mamy znajomych korzystających z różnych sieci GSM. Dlaczego więc nie robić tego za pomocą jednego programu?

Artur Janowski

SMS-y (czyli krótkie wiadomości tekstowe) są symbolem obecnych czasów. Prostota ich wysyłania oraz „intymność" otrzymywania dają im przewagę nad niespodziewanymi telefonami od sympatii podczas narady z szefem. Wszystko to sprawia, iż mamy do czynienia z prawdziwym boomem na SMS-y. Wszyscy krajowi operatorzy telefonii komórkowej oferują na swoich stronach WWW możliwość wysyłania krótkich wiadomości tekstowych via Internet. Jest to usługa bezpłatna dla osób mających „darmowy" dostęp do Sieci. Bramki SMS, jak nazywane są tego typu strony, wymagają od użytkownika wykonania szeregu operacji, związanych przede wszystkim z wypełnianiem i zatwierdzaniem odpowiednich formularzy internetowych.

Zajmuje to zawsze trochę czasu, zwłaszcza przy próbie wysłania większej liczby wiadomości, a każda kolejna minuta korzystania z Sieci sporo nas kosztuje. Tak, w pracy można prawie wszystko, a co „po godzinach"? Zwykle w takich przypadkach, chcąc oszczędzić cenny czas, sięgamy po programy typu Gadu-Gadu. Nie każdy jednak ma swobodny dostęp do komputera, a wspomniane programy wymagają często instalacji oraz trochę pracy przy ich konfiguracji. Czemu zatem nie napisać czegoś prostszego, mieszczącego się na dyskietce, z którą można zawsze pójść do kolegi czy kawiarenki internetowej, i maksymalnie uprościć sobie proces wysyłania SMS-ów? Jakiego jednak użyć narzędzia do stworzenia takiej „pchełki"? Moje wrodzone zamiłowanie do Pascala połączone z fizjologiczną chyba awersją do narzędzi firmy Microsoft skłoniły mnie do napisania aplikacji w Delphi (darmowa wersja Delphi – Personal Edition – znajduje się na krążku CHIP-CD 10/2001).

Wszystko jest w Sieci

Aby nie wyważać już otwartych drzwi, postanowiłem poszukać w Internecie informacji przydatnych do zbudowania przykładowej aplikacji. Okazało się, że jest ich dość sporo, a ponadto temat SMS-ów jest jednym z wątków często przewijających się na grupach dyskusyjnych. Nigdzie natomiast nie znalazłem opisu kompleksowego rozwiązania naszego problemu – tj. takiego, które obejmowałoby całościowo problem wszystkich trzech operatorów jednocześnie, z wykorzystaniem Delphi (poza nieeleganckimi próbami sprzedaży gotowych komponentów służących temu celowi, co wg regulaminów operatorów stanowi nadużycie).

Formularze HTML

Przed rozpoczęciem tworzenia programu powinienem najpierw przedstawić w skrócie zasadę wysyłania krótkich wiadomości tekstowych z bramek WWW/SMS udostępnianych przez naszych rodzimych operatorów GSM (Era, Plus, Idea). Każdy z nich oferuje możliwość nadawania wiadomości tylko i wyłącznie do abonentów swojej sieci (wyjątkiem jest tu niezależny od ogólnie dostępnej bramki SMS Polkomtelu projekt o nazwie „Miasto Plusa", przeznaczony tylko dla abonentów sieci Plus GSM. Umożliwia on wysyłanie SMS-ów do użytkowników wszystkich polskich sieci komórkowych).

Na szczęście koncepcja obsługi „internetowych" SMS-ów u wszystkich operatorów jest podobna i opiera się na wykorzystaniu formularzy HTML do komunikacji z serwerami WWW. Po wypełnieniu i zatwierdzeniu takiego formularza przez nadawcę wprowadzone dane, tj. numer telefonu odbiorcy, dane nadawcy oraz treść wiadomości, przesyłane są z użyciem metody POST do serwera WWW operatora (osoby, które nie wiedzą, jak skonstruowane są formularze HTML, powinny sięgnąć do dowolnego podręcznika tego języka). Ten ostatni – po skontrolowaniu poprawności informacji, a w niektórych przypadkach również autoryzowaniu konkretnego formularza przez serwer (Era) – wywołuje procedurę odpowiedzialną za techniczne dostarczenie SMS-a.

Od tego momentu los naszej wiadomości zależy od serwerów operatorów (przede wszystkim od ich obciążenia), a informacja zwrotna o rzekomym dostarczeniu jest faktycznie sygnalizacją przyjęcia SMS-a w poczet oczekujących na wysłanie. Stąd zdarzające się często opóźnienia w dostarczaniu wiadomości – zdaniem operatorów jest to przecież usługa bezpłatna i nikt tak naprawdę nie ma prawa się żalić.

Co będziemy robili?

Zadaniem mojej aplikacji (nazwanej tu GateWay) jest uproszczenie procesu wysyłania krótkich wiadomości tekstowych bez nadmiernego uzależniania się od konkretnego operatora. Powinna więc ona potrafić rozpoznać na podstawie numeru adresata sieć danego operatora i w konsekwencji wypełnić

Artykuł:	Opis:	Brief dla WWW:	SŁOWA KLUCZOWE
Monopol jest OK	Programowanie w Delphi.	Zamiast korzystać z bramek WWW/SMS poszczególnych operatorów GSM, możemy napisać własny programik, który będzie to robił za nas. Brzmi zachęcająco? No to do dzieła.	Software | Narzędzia programistyczne | Środowiska programistyczne

automatycznie formularze znajdujące się na odpowiedniej stronie WWW. Dzięki temu wiadomość przeznaczona dla abonenta Idei nie będzie np. wysyłana przez stronę Ery – z wiadomych względów nie da to żadnego efektu. Fragment kodu odpowiedzialny za podjęcie takiej decyzji (przy założeniu, że wartością zmiennej **numer_i** jest numer telefonu adresata) może mieć postać:

```
prefix:=numer_i div 1000000;
case prefix of
501..504:idea;
601,603,605,607,609,691:plus;
600,602,604,606,608,692:era;
end;
```

gdzie **idea**, **plus**, **era** są nazwami procedur związanych z wypełnieniem formularzy na stronach odpowiednich operatorów. Każda z nich musi wysłać do serwera określonego operatora konkretne parametry, dzięki którym serwer będzie „wiedział", jakie czynności powinien podjąć. Ponieważ parametry te są wysyłane metodą POST (nie są jawnie przekazywane w adresie strony, jak to się dzieje w przypadku metody GET), ich poznanie wymaga odrobiny pracy związanej z analizą kodu strony WWW z bramką.

Istnieje jednak kilka sposobów na szybkie poznanie tych parametrów. Przedstawię według mnie najprostszą, a w opisywanej sytuacji znakomicie się sprawdzającą metodę. Wystarczy tylko zapisać na lokalnym dysku kopię sieciowej strony operatora, odpowiedzialnej za wysyłanie SMS-ów (np. w IE poleceniem **Plik | Zapisz jako**). Teraz wśród zapisanych plików należy znaleźć ten, który zawiera kod HTML z definicją formularza bramki SMS. W przypadku Plusa definicja ta rozpoczyna się tak:

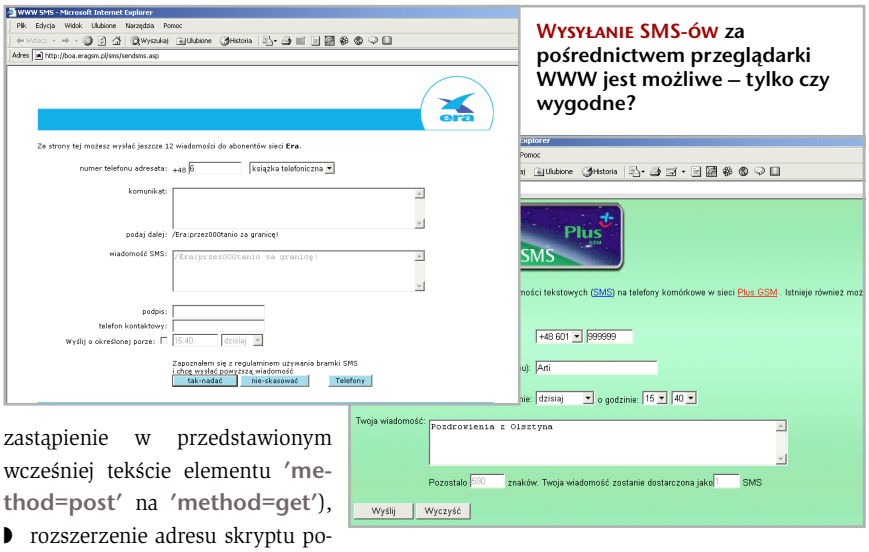
```
<FORM name=form onsubmit='return
 kontrola(this)' action=sendSMS.php
 method=post>
```

Kolejnymi czynnościami są:

- zmiana metody POST na GET (zwyczajne zastąpienie w przedstawionym wcześniej tekście elementu **'method=post'** na **'method=get'**),
- rozszerzenie adresu skryptu podawanego przez parametr **'action'** nazwą protokołu – **action=http://sendSMS.php**;
- zapisanie zmodyfikowanej wersji pliku.

Wczytanie tak przygotowanego dokumentu HTML do przeglądarki internetowej będzie skutkowało pojawianiem się wszystkich poszukiwanych parametrów przy każdej próbie wysłania SMS-a na pasku adresu przeglądarki, np.: **...sendSMS.php?tprefix=601&numer=999999&odkogo=J23&dzien=2001-10-30&godz=14&min=05&tekst='Pozdrowienia z Olsztyna'**.

SMS oczywiście nie zostanie nadany, ponieważ w podanym adresie brakuje jeszcze dokładnej nazwy serwera oraz lokalizacji skryptu „sendSMS.php" na tymże serwerze. Opisana metoda służy jedynie poznaniu przekazywanych do serwerowego skryptu parametrów (dane po tekście **'sendSMS.php?'**). Oddzielane są one znakami „&", a ich wartości pochodzą z elementów wypełnianego formularza. Wynika z tego, że do wysłania SMS-a do abonenta Plusa wystarczy jedynie przekazanie linii odpowiednio zmodyfikowanych parametrów skryptowi znajdującemu się pod adresem http://www.text.plusgsm.pl/sms/sendsms.php. Podobnie wysyłane są informację w sieci Idea. Ich samodzielne „rozgryzienie" lub przeanalizowanie załączonego na płycie CD kodu źródłowego pozostawiam Czytelnikom.



WYSYŁANIE SMS-ÓW za pośrednictwem przeglądarki WWW jest możliwe – tylko czy wygodne?

Era SMS-ów, SMS-y Ery...

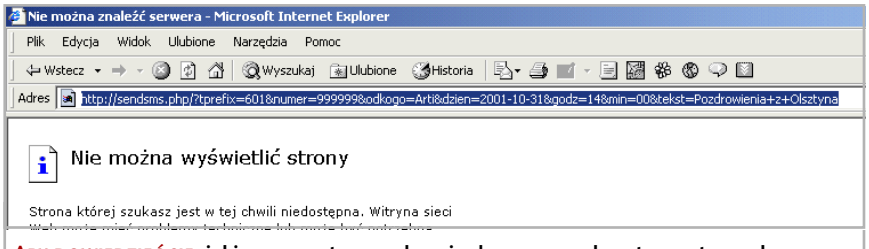
Zajmę się teraz trochę dokładniej przypadkiem bramki Ery. Po zastosowaniu przedstawionego sposobu poznawania przesyłanych parametrów oraz ich wartości zaobserwowałem, że wśród nich znajduje się również jakiś „dziwny", bo niemający swojego odzwierciedlenia w polach wypełnianego formularza, atrybut o nazwie **'code'**:

```
http://boa.eragsm.com.pl/sms/send
 sms.asp?bookopen=&numer=600000000'
 &ksiazka=&message=Pozdrowienia&
 podpis=J23&kontakt=&Code=234&
 Nadaj=++tak-nada%E6
```

Okazuje się, że jego poprawna wartość, za każdym razem inna, wraz z załączanym ciasteczkiem (cookie), stanowi dla skryptu http://boa.eragsm.com.pl/SMS/sendSMS.asp „zapewnienie", że został on wywołany przez naciśnięcie odpowiedniego przycisku na stronie Ery, a nie przez zwykłe wysłanie parametrów np. z programu GateWay.

Jak sobie z tym poradzić, tzn. jak pobrać wspomniany kod? Niestety, rozwiązanie wymaga wydłużenia czasu połączenia internetowego, koniecznego do wysłania wiadomości. Nasz program musi bowiem pobrać całą stronę z formularzem do wysyłania SMS-ów (jest ona wywoływana przez przekazanie do skryptu sendSMS.asp parametru SMS=1) spod adresu **http://boa.eragsm.com.pl/SMS/sendSMS.asp?SMS=1**.

Serwer zwróci odpowiedź (w postaci dokumentu HTML), w której treści należy odnaleźć „tajny kod" i użyć go do wysłania SMS-a, korzystając z adresu: **http://boa.eragsm.com.pl/SMS/sendSMS.asp?bookopen=&numer=600000000&ksiazka=&message=Pozdrowienia**



ABY DOWIEDZIEĆ SIĘ, jakie parametry przekazują do serwera skrypty na stronach operatorów, zapiszemy i uruchomimy lokalnie dokument PHP.

Kod źródłowy programu

» Fragment odpowiedzialny za wysyłanie SMS-a

Poniżej znajduje się fragment procedury **TForm1.ERA()** (patrz: kod źródłowy na CD), bezpośrednio odpowiedzialny za wysłanie formularza z SMS-em do bramki Ery:

```
if ok then
 begin
  GetMem(Bufor,length(Tekst.text));
  Tekst.GetTextBuf(Bufor,length(Tekst.text));
  for i:=0 to length(Tekst.text) do
   if bufor[i]=' ' then bufor[i]:='+';
    nadanie:=TStringStream.Create('');
    odbior:=TStringStream.Create('');
    nadanie.WriteString('?bookopen=&numer='+Numer.Text+
     '&ksiazka=&message='+StrPas(bufor)+
     '&podpis='+Nadawca.Text+
     '&kontakt=&Code='+s+'&Nadaj=++tak-nada%E6++');
    FreeMem(Bufor,length(Tekst.text));
    IdHTTP1.Request.Accept:='*/*';
    IdHTTP1.Request.Host:='boa.eragsm.com.pl';
    IdHTTP1.Request.referer:='boa.eragsm.com.pl';
    IdHTTP1.Request.contenttype:='application/x-www-form-urlencoded';
    try
     IdHTTP1.Post('http://boa.eragsm.com.pl/SMS/sendSMS.asp',
      nadanie,odbior);
    except on Exception do ShowMessage('Problemy z połączeniem!!!');
   end;
   if pos('Wiadomoć dla numeru +48'+Numer.Text+' została wysłana!',
    odbior.DataString)<>0 then ShowMessage('SMS DOSTARCZONY')
   else ShowMessage('SMS NIEDOSTARCZONY');
  nadanie.free;
  odbior.free;
  IdHTTP1.Disconnect;
 end;
end;
```

&podpis=J23&kontakt=&Code=234&Nadaj=++tak-nada%E6++

Jaki komponent?

Skoro znamy już nieco teorii, przejdziemy do implementacji programu. Standardowo dostępny w Delphi 5 komponent do obsługi protokołu HTTP firmy NetMaster, znajdujący się w zakładce FastNet, w przypadku metody POST niestety zawodzi (przynajmniej w moim przypadku). Dlatego też postanowiłem skorzystać z freeware'owego pakietu komponentów Indy 9.0. Można go pobrać ze strony podanej w ramce „Info", a w Delphi 6 jest on dostępny standardowo. Komponentem wykonującym większość pracy naszego programu, tj. łączenie się z odpowiednimi serwerami WWW operatorów, będzie **TIdHTTP**. Po utworzeniu nowego projektu umieszczamy wspomniany komponent na formularzu. Następnie dodajemy do projektu okna programu kolejne potrzebne elementy (wymienione zgodnie ze schematem: nazwa – typ):

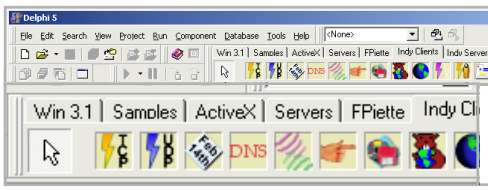
- Numer, Nadawca – TEdit,
- Label1, Label2 – TLabel,
- OK, Przerwij – TBitBtn,
- StatusBar1 – TStatusBar,
- Czysc – TButton,
- StatusBar2 – TStatusBar,
- IdAntiFreeze1 – TIdAntiFreeze,
- Tekst – Tmemo.

Wysyłamy SMS-y

Najpierw utworzymy procedurę wysyłającą SMS-a do abonenta Plusa. Może ona wyglądać następująco:

```
procedure TForm1.PLUS();
var
odbior,nadanie: TStringStream;
begin
 odbior:=TStringStream.Create('');
 nadanie:=TStringStream.Create('');
 nadanie.WriteString('tprefix='+
   copy(Numer.Text,0,3)+
 '&numer='+copy(Numer.Text,4,6)+
 '&odkogo='+Nadawca.Text+
 '&dzien=2001-10-30&godz=14&min=05
   &tekst='+Tekst.text);
 IdHTTP1.Request.contenttype:=
   'application/x-www-form-
   urlencoded';
 try
  dHTTP1.Post('http://www.text.
   plusgsm.pl/SMS/sendSMS.php',
   nadanie,odbior);
 except on Exception do ShowMessage
   ('Problemy z połączeniem!!!');
 end;
 if pos('Twoja wiadomość została
   wysłana',odbior.DataString)<>0
 then
   ShowMessage('SMS DOSTARCZONY')
 else
   ShowMessage('SMS NIE DOSZEDŁ!');
 nadanie.free;
 odbior.free;
 IdHTTP1.Disconnect;
end;
```

Metoda **Post** komponentu **TIdHTTP** wysyła na adres wskazany pierwszym parametrem dane zawarte w drugim parametrze, a to, co odpowie serwer – umieszcza w trzecim. Jeśli wszystko przebiegnie pomyślnie, ostatni parametr metody powinien zawierać między innymi tekst „Twoja wiadomość została wysłana" – stronę HTML z takim m.in. zdaniem zwraca serwer WWW Plusa, jeśli wiadomość zostaje przyjęta do realizacji. Dlatego też w procedurze obsługującej bramkę Plusa pojawiła się funkcja poszukująca tego tekstu w dokumencie z odpowiedzią



Do napisania własnego programu wysyłającego SMS-y wykorzystamy DARMOWE KOMPONENTY FIRMY INDY.

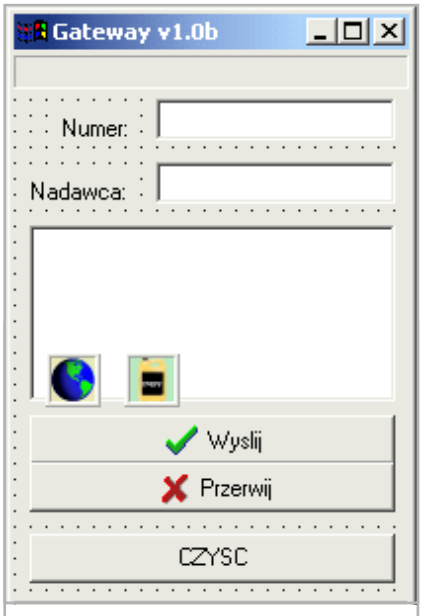
serwera (**if pos('Twoja wiadomość została wysłana',odbior.DataString)<>0**).

Na krótkie wyjaśnienie zasługuje właściwość **Request.Contenttype** komponentu TIdHTTP. Zawiera ona specyfikację typu danych dołączanych do zlecenia wysłanego przez klienta, dla przeglądarek internetowych standardowo równy **'application/x-www-form-urlencoded'**.

Bardzo podobnych jak w przypadku Plusa „zabiegów" związanych z wysłaniem SMS-a wymaga procedura przeznaczona dla abonentów operatora Idea. Z tego też względu zostaną one tu pominięte. Zainteresowanych Czytelników odsyłam do analizy załączonego pliku źródłowego.

Z Erą będzie trochę trudniej, gdyż musimy wykonać dwa wywołania metody POST. Pierwsze w celu poznania wartości parametru **code**:

```
procedure TForm1.ERA();
var
 odbior,nadanie:TStringStream;
 s:string;
 i,i2:integer;
 ok:boolean;
 bufor:PChar;
begin
 ok:=True;
 odbior:=TStringStream.Create('');
 nadanie:=TStringStream.Create('');
 nadanie.WriteString('SMS=1');
 begin
 IdHTTP1.Request.Host:=
   'boa.eragsm.com.pl';
 IdHTTP1.Request.contenttype:=
   'application/x-www-form-
   urlencoded';
 IdHTTP1.Request.referer:=
   'boa.eragsm.com.pl';
 try
   IdHTTP1.Post('http://boa.eragsm.
   com.pl/SMS/sendSMS.asp',nadanie,
   odbior);
 except on Exception do
   begin
     ShowMessage('Problemy z
       połączeniem!!!');
     ok:=False;
   end;
 end;
 s:=odbior.DataString;i:=pos('name=
   'Code' value='',s);
 s:=copy(s,i+19,100);
 i2:=pos(''',s);
 s:=copy(s,0,i2-1);
 nadanie.free;
 odbior.free;
end;
```



PROJEKT FORMULARZA może wyglądać np. tak – ale nic nie stoi na przeszkodzie, żeby to zmienić.

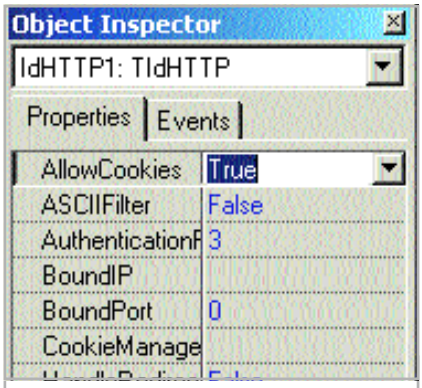
Po wywołaniu metody POST w zmiennej o nazwie odbiór znajduje się zwrócony przez serwer WWW Ery dokument HTML, a przedstawiony poniżej kod poszukuje w nim wyrażenia **'name='Code' value=''**, po którym następuje potrzebna nam wartość parametru code. Należy ją pobrać i wykorzystać przy kolejnym wywołaniu metody POST.

Uważni Czytelnicy zauważyli pewnie w kodzie procedury obsługującej formularze Ery dwie kolejne właściwości komponentu **TIdHTTP**, tj. **Request.Host** oraz **Request.referrer**. Zawierają one nazwy hosta oraz adresu, z którego wykonane zostało odwołanie do skryptu. Skrypt Ery sprawdza te wartości w procesie weryfikacji autentyczności formularzy, dlatego przed odwołaniem się do skryptu należy przypisać tym parametrom wartości: **boa.eragsm.com.pl**.

Na dodatek

SMS-y wysyłane z bramek WWW mają ograniczoną długość, która u każdego operatora jest inna. Dlatego postanowiłem napisać procedurę zapobiegającą przekroczeniu ustalonego limitu znaków, wywoływaną po każdej zmianie zawartości **Tekst: Tmemo**:

```
procedure TForm1.TekstChange
  (Sender: TObject);
begin
 Pokaz_znaki;
end;
```



Najważniejszy jest dla nas KOMPONENT TIdHTTP. Niektóre jego właściwości ustawiamy w Object Inspectorze.

```
function TForm1.Pokaz_znaki:integer;
begin
 result:=znaki-length(Nadawca.Text)
  -length(Tekst.text);
 StatusBar1.SimpleText:='Pozostało
   Ci jeszcze:'+inttostr(result)+'
   znaków';
 if Result<0 then OK.Enabled:=false
   else OK.Enabled:=True;
 if length(Tekst.Text)<1 then
   OK.Enabled:=false;
end;
```

Zmienna **znaki** zawiera maksymalną dopuszczalną liczbę znaków wiadomości (określona dla danego operatora), podaną, na podstawie analizy numeru telefonu odbiorcy, przez procedurę obsługi zdarzenia **OnExit** kontrolki **Numer: TEdit**.

W naszym programie nie pozostało już wiele do zrobienia. Można oczywiście dopracować nieco interfejs użytkownika czy postarać się o prostą książkę adresową. Warto przy tym pamiętać, że przedstawiona w niniejszym tekście technologia automatyzacji obsługi bramek SMS może służyć jedynie celom prywatnym i nie powinna znaleźć zastosowania w masowych i komercyjnych przedsięwzięciach. ■

INFO

KOMPONENTY INDY
http://www.nevrona.com/Indy/
STRONA BORLANDA
http://www.borland.pl/

Na płycie CD dołączonej do numeru, w dziale **Porady | Programowanie w Delphi**, znajduje się kod źródłowy programu opisywanego w artykule i zestaw komponentów Indy w wersji 9.