helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

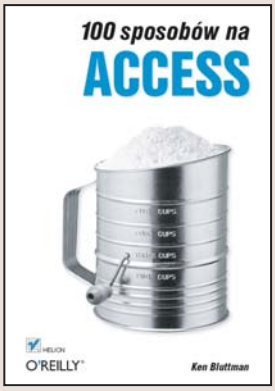
KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

100 sposobów na Access



Autor: Ken Bluttman
Tłumaczenie: Paweł Koronkiewicz
ISBN: 83-246-0058-2
Tytuł oryginału: Access Hacks
Format: B5, stron: 352

Odkryj nieznane możliwości Accessa

- Wyświetl pliki multimedialne w oknach Accessa
- Skorzystaj z funkcji obsługi plików XML
- Połącz Accessa z innymi bazami danych

Microsoft Access to najpopularniejsza w zastosowaniach biurowych baza danych. Dzięki swej prostocie, dużym możliwościom oraz zintegrowanym mechanizmom służącym do szybkiego tworzenia aplikacji i interfejsów użytkownika. Access jest stosowany wszędzie tam, gdzie pożądana jest minimalna ilość czynności związanych z konserwacją wdrożonego rozwiązania. Może również pełnić funkcję interfejsu użytkownika dla innych baz danych, takich jak Oracle czy MySQL. Poza możliwościami widocznymi na pierwszy rzut oka i powszechnie stosowanymi Access oferuje również wiele mniej znanych funkcji, które mogą okazać się niezwykle przydatne w jego codziennym użytkowaniu.

„100 sposobów na Access" to podręcznik przedstawiający wszystkie funkcje tej bazy danych. Czytając tę książkę, dowiesz się, jak budować złożone kwerendy, odtwarzać pliki wideo i wyświetlać strony WWW w formularzach oraz wykorzystywać funkcje Excela w Accessie. Nauczysz się tworzyć wydajne aplikacje, stosować Accessa w połączeniu z innymi bazami danych oraz korzystać z funkcji pozwalających na obsługę plików XML.

- Dostosowywanie interfejsu aplikacji
- Porządkowanie biblioteki makropoleceń
- Szybkie kopiowanie danych pomiędzy tabelami
- Optymalizacja i usprawnianie działania formularzy
- Stosowanie formatowania warunkowego
- Budowanie tabel Accessa za pomocą MS SQL Servera
- Łączenie Accessa z innymi aplikacjami pakietu MS Office
- Tworzenie i stosowanie aplikacji dodatkowych

Dzięki tej książce poznasz nowe sposoby pracy z Accessem

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

Spis treści

ROZDZIAŁ TRZECI

Wprowadzanie danych i nawigacja

Sposoby 19. – 27.

O tym, czy aplikacja zostanie uznana za dobrą i udaną, decyduje często przychylność przeciętnego użytkownika. Pamiętając o tym, łatwo docenimy wartość prostej przyjemności korzystania z niej i wygody użytkowania.

Użytkowa strona aplikacji bywa często zaniedbywana. Programiści spędzają długie godziny na projektowaniu tabel, pól, relacji, kwerend i innych elementów wewnętrznych, najistotniejszych dla poprawnego i wydajnego funkcjonowania aplikacji. Czy jest to zauważalne dla typowego użytkownika? Ani trochę!

Spójrzmy prawdzie w oczy. Access jest nie tylko systemem zarządzania bazami danych. To SZBD z wbudowanymi narzędziami do budowy frontonu aplikacji. Tabele to rdzeń bazy danych, ale formularze i raporty to narzędzia platformy programowania aplikacji. Access łączy oba te elementy, oba więc wymagają dopracowania.

W tym rozdziale zajmiemy się usprawnieniami, które zauważy najmniej „zaawansowany komputerowo" użytkownik. Wprowadzanie danych to zwykle najczęściej wykonywana przez niego czynność. Postarajmy się nieco urozmaicić to monotonne zajęcie.

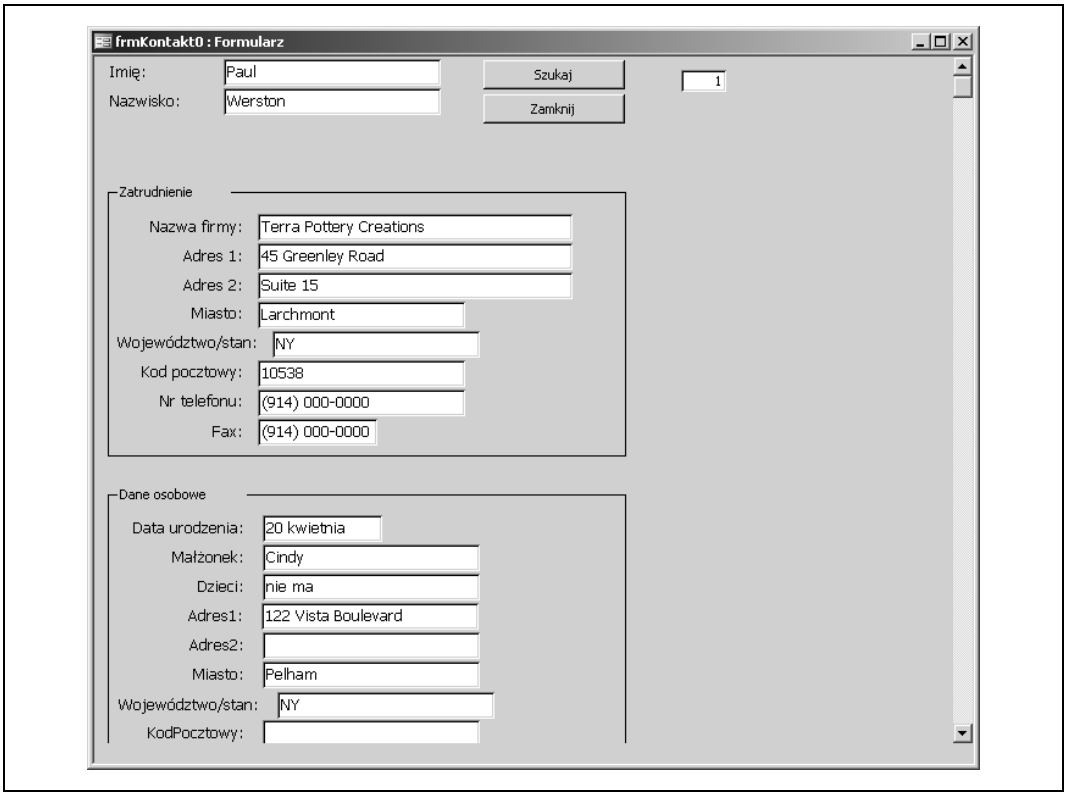
SPOSÓB 19.

Nawigacja w długich formularzach

Używaj formantów podziału strony i przycisków poleceń, aby uniknąć przewijania długich formularzy wprowadzania danych.

Informacja jest ważna. Im więcej wiemy, tym więcej da się zrobić, więcej zaplanować — o ile nie utkniemy na etapie wprowadzania danych. Jeżeli tak się stanie, możemy jedynie zaplanować dużo pisania i klikania.

Rysunek 3.1 przedstawia stosunkowo długi formularz wprowadzania danych. Zawiera on więcej pól, niż można pomieścić w rozsądny sposób na ekranie. Dowodzi tego widoczny z prawej strony pasek przewijania. Osoba korzystająca z takiego formularza będzie musiała przewijać jego zawartość, używając albo paska, albo klawisza *Tab*.

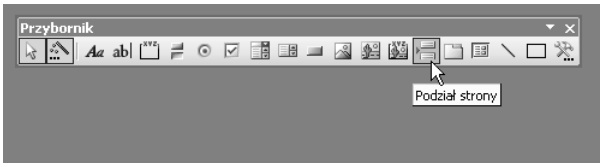


Rysunek 3.1. Formularz wymagający żmudnego wpisywania

Dobrym narzędziem ułatwiającym zapanowanie nad dużą ilością elementów sterujących formularza jest formant Karta. Tutaj nie będziemy o nim pisać. Opieramy się na przykładzie wziętym z praktyki. Autor pracował kiedyś nad projektem, w którym osoby wprowadzające dane przenosiły je z dużych papierowych arkuszy (w amerykańskim formacie Legal, znacznie większym od A4). Jednym z wymagań stawianych aplikacji było to, aby wygląd formularza na ekranie był odbiciem jego papierowego odpowiednika.

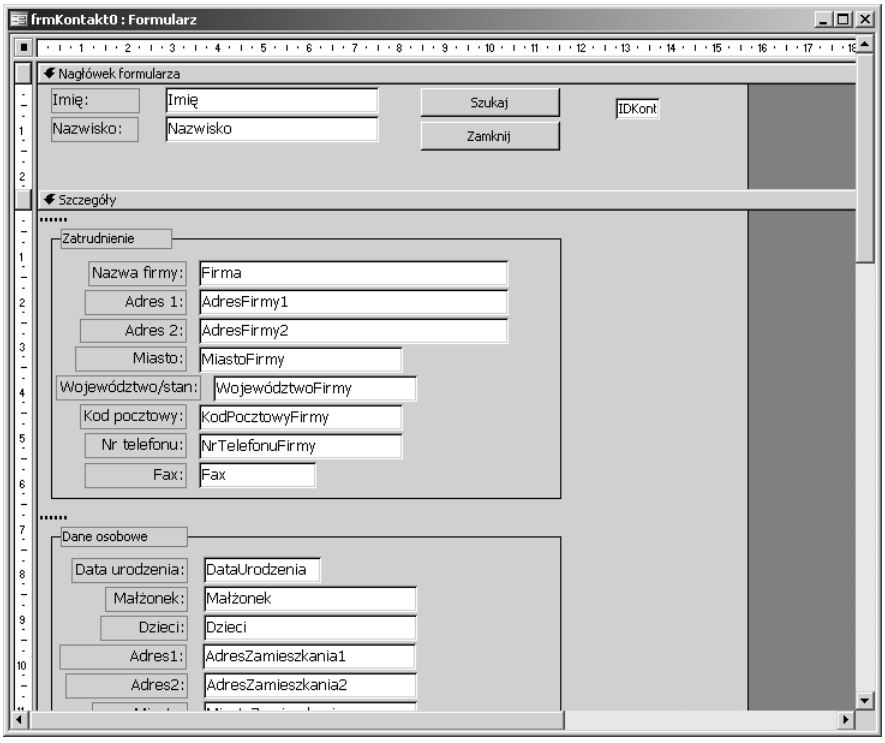
Szybkie przewijanie umożliwiają klawisze *Page Up* i *Page Down*. Nie pozwalają one jednak kontrolować tego, jak daleko zawartość formularza zostanie przesunięta. Szansa, że po wciśnięciu jednego z tych klawiszy formularz znajdzie się we właściwym położeniu jest stosunkowo niewielka.

Na szczęście istnieje dobre rozwiązanie takiego problemu. Jest nim *formant Podział strony*. Gdzie go znajdziemy? Rysunek 3.2 przedstawia Przybornik z zaznaczonym formantem podziału strony. Dobrze rozmieszczone formanty podziału mogą zapewnić, że klawisze *Page Up* i *Page Down* będą zawsze przewijały formularz do właściwej pozycji.



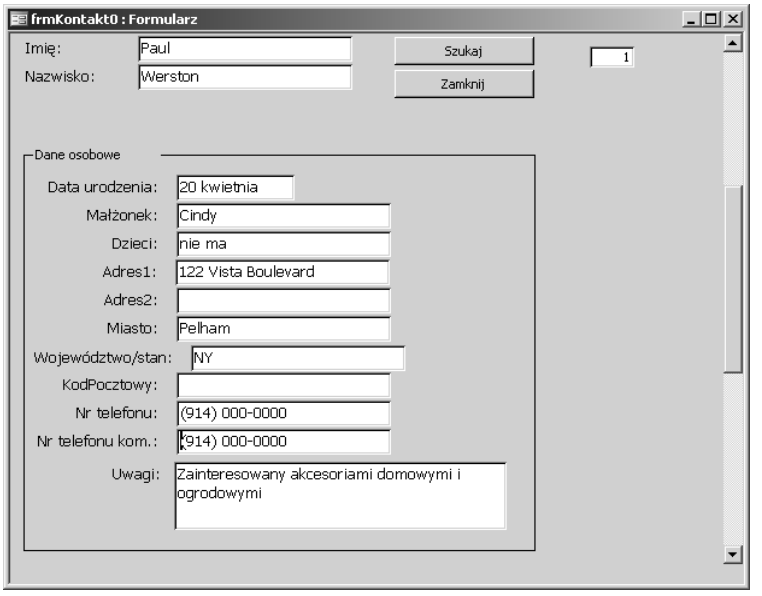
Rysunek 3.2. Ikona formantu podziału strony w Przyborniku

Rysunek 3.3 przedstawia formularz w widoku projektu. Formant podziału strony został umieszczony bezpośrednio nad ramką danych osobowych. Nie jest to formant kłopotliwy. W widoku projektu symbolizuje go krótka kropkowana linia. W widoku formularza nie jest wcale widoczny.



Rysunek 3.3. Dodawanie formantów podziału strony

Po umieszczeniu formantu podziału strony na formularzu klawisz *Page Down* przewija okno dokładnie do miejsca, w którym znajduje się ten formant. Rysunek 3.4 przedstawia wcześniejszy formularz po przewinięciu „o jedną stronę".



Rysunek 3.4. Inteligentne przewijanie formularza

Jeszcze lepsza nawigacja

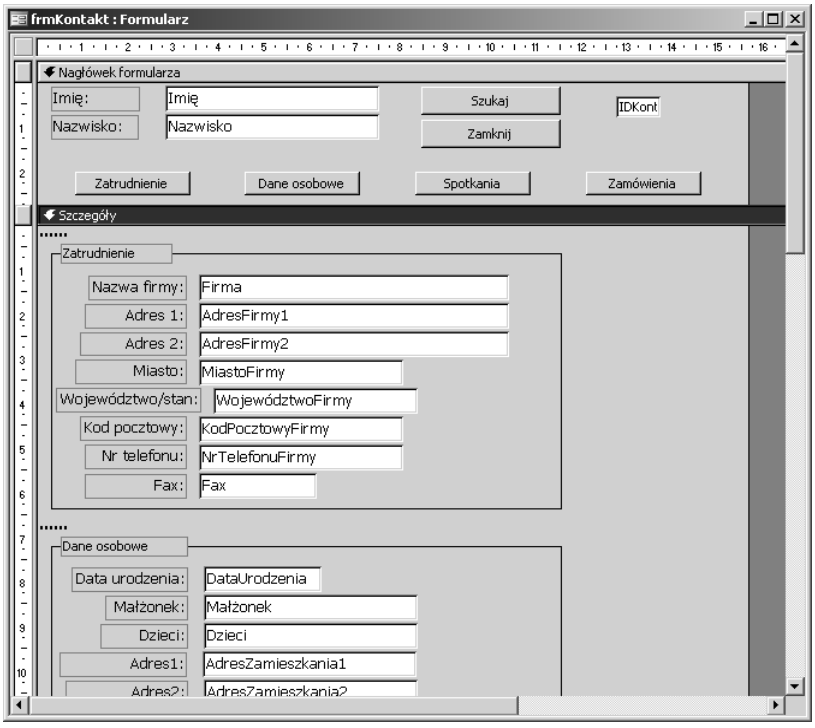
Korzystanie z klawiszy *Page Up* i *Page Down* jest dobrym i intuicyjnym sposobem przewijania formularza. Nie znaczy to, że nie może być jeszcze lepiej. Wyobraźmy sobie formularz z kilkoma ramkami wprowadzania danych (podobnymi do przedstawionych w przykładzie powyżej) i wymagający dostępu do nich w dowolnej kolejności. Wielokrotne wciskanie klawiszy *Page Up* i *Page Down* może być wtedy uciążliwe.

Aby uniknąć takiego problemu, można umieścić w nagłówku lub w stopce formularza grupę przycisków poleceń. Nagłówek i stopka to elementy zawsze widoczne (nie podlegają przewijaniu wraz z sekcją szczegółów). Rysunek 3.5 przedstawia projekt formularza wyposażonego w grupę przycisków umieszczonych w nagłówku. Pozwalają one przewinąć zawartość okna do dowolnego miejsca, odpowiednio do bieżących potrzeb.

Kod

Cała sztuka polega na przypisaniu przyciskom procedur symulujących wciśnięcie klawiszy *Page Up* i *Page Down*. Poniższy kod realizuje to przy użyciu instrukcji `SendKeys`. Każdy z czterech nowych przycisków inicjuje wykonanie serii takich instrukcji:

```
Private Sub cmdWorkInfo_Click()
'
' 4 SendKeys PageUp, 1 SendKeys PageDown
'
  navigate_form 4, 1
End Sub
```



Rysunek 3.5. Przyciski nawigacyjne, które pomagają korzystać z formularza

```
Private Sub cmdPersonalInfo_Click()
'
' 4 SendKeys PageUp, 2 SendKeys PageDown
'
  navigate_form 4, 2
End Sub

Private Sub cmdContactDetails_Click()
'
' 4 SendKeys PageUp, 3 SendKeys PageDown
'
  navigate_form 4, 3
End Sub

Private Sub cmdOrders_Click()
'
' 4 SendKeys PageUp, 4 SendKeys PageDown
'
  navigate_form 4, 4
End Sub

Sub navigate_form(u As Integer, d As Integer)
  For form_up = 1 To u
    SendKeys "{PGUP}"
  Next form_up
  For form_down = 1 To d
    SendKeys "{PGDN}"
  Next form_down
End Sub
```

Każde zdarzenie `Click` (*Przy kliknięciu*) prowadzi do wysłania pewnej liczby „uderzeń klawiszy" *Page Up* i *Page Down*. Wynikiem jest przewinięcie formularza do właściwego miejsca. Symulowane uderzenia klawiszy wykorzystują umieszczone wcześniej w sekcji *Szczegóły* formanty *Podział strony*. Każda procedura zdarzenia rozpoczyna pracę od wysłania czterech kodów *Page Up*. Jest to oczywiście liczba dostosowana do naszego przykładu. Celem tej czynności jest przejście do początku formularza. Aż cztery *Page Up* są potrzebne tylko wtedy, gdy wyświetlany jest sam koniec arkusza wprowadzania danych. Nie sprawia to jednak problemu w innych ustawieniach.

Podstawowym elementem jest odpowiednio dobrana liczba kodów *Page Down*. Dla każdego przycisku jest ona inna, bo każdy przycisk ma przewijać do innego miejsca. Przycisk *Przedstawiciel* wymaga jednego *Page Down*, przycisk *Rozmowy* dwóch itd.

Dzięki takiemu rozwiązaniu w każdym momencie pracy z formularzem można wyświetlić dowolną grupę danych, używając tylko jednego kliknięcia.

SPOSÓB 20.

Łatwe uzupełnianie pól tekstowych

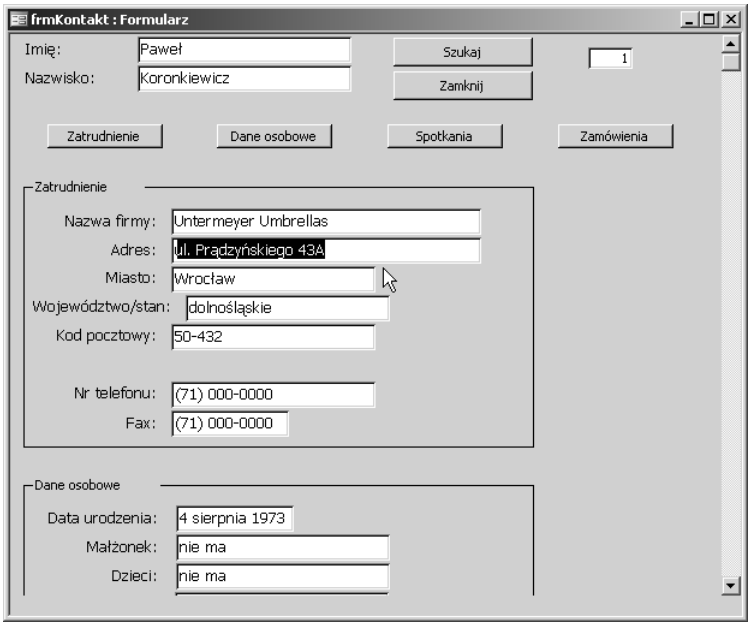
Umieść punkt wstawiania na końcu wpisu w polu tekstowym, aby rozpoczęcie wprowadzania dodatkowych danych nie wymagało dodatkowych czynności.

Metoda, o której będziemy tu pisać, jest ogromnym, a bardzo często zapominanym ułatwieniem. Czy zwróciłeś uwagę na to, że gdy w trakcie pracy z formularzem przejdziesz do kolejnego formantu i tym formantem jest pole tekstowe, zaznaczany jest od razu cały tekst pola? Niestety, to standardowe zachowanie sprawia, że dane pola stają się bardzo podatne na przypadkowe usunięcie. Rysunek 3.6 przedstawia pole tekstowe adresu, bezpośrednio po przejściu do niego wciśnięciem klawisza *Tab*. Jeżeli naszym zamiarem jest *dołączenie* tekstu, a nie jego *zastąpienie*, musimy umieścić wskaźnik myszy na końcu wpisu i kliknąć, aby usunąć zaznaczenie i odpowiednio umiejscowić punkt wstawiania (lub zrobić to samo przy użyciu klawiatury).

Czy nie byłoby prościej, gdyby użytkownik nie musiał klikać, aby usunąć zaznaczenie? Aby tak było, potrzebna jest naprawdę minimalna ilość kodu.

Wiele formantów, w tym pola tekstowe, dysponuje zdarzeniem `Enter`, które następuje w momencie kliknięcia formantu lub przejścia do niego klawiszem *Tab*. Właśnie w procedurze obsługi tego zdarzenia możemy umieścić kod, który przeniesie kursor na koniec tekstu, jeszcze zanim użytkownik będzie mógł rozpocząć wpisywanie danych. Oto przykład takiego kodu, obsługującego zdarzenie wejścia formantu o nazwie `CompanyAddress1`:

```
Private Sub CompanyAddress1_Enter()
  Dim text_length As Integer
  text_length = Len(Me.CompanyAddress1)
  Me.CompanyAddress1.SelStart = text_length
End Sub
```



Rysunek 3.6. Automatycznie zaznaczone dane są zagrożone przypadkowym usunięciem lub zamianą

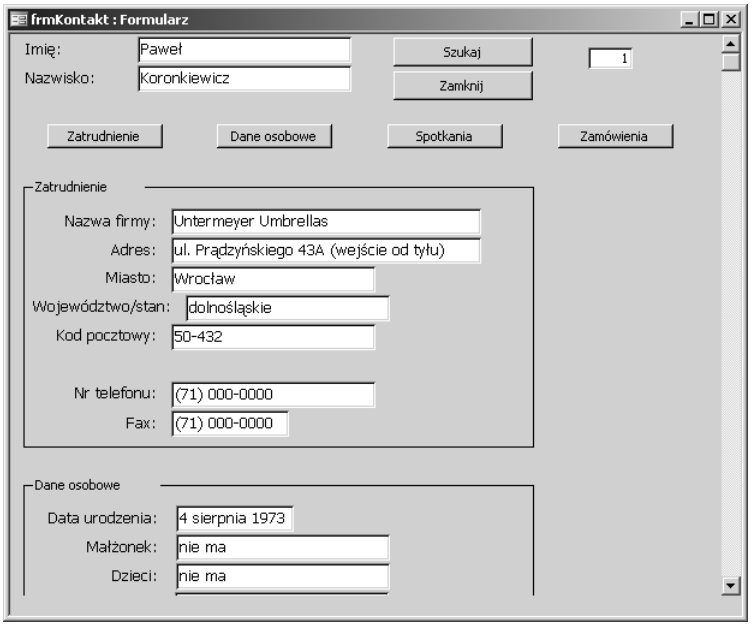
Ustalamy długość tekstu funkcją `Len`, a następnie przypisujemy uzyskaną wartość właściwości `SelStart`. To wszystko.

Jeżeli to potrzebne, w podobną procedurę można wyposażyć wszystkie pola tekstowe formularza. Musimy się zastanowić: czy dane są zazwyczaj zastępowane, czy uzupełniane? Zależy to od budowanej aplikacji. W naszym przykładzie dodajemy do adresu dodatkowe dane. Ilustruje to rysunek 3.7.

Z właściwością `SelStart` wiążą się dwie inne: `SelLength` i `SelText`. Odpowiednio je wykorzystując, uzyskujemy niemal pełną kontrolę nad obsługą tekstu w polu tekstowym lub kombi.

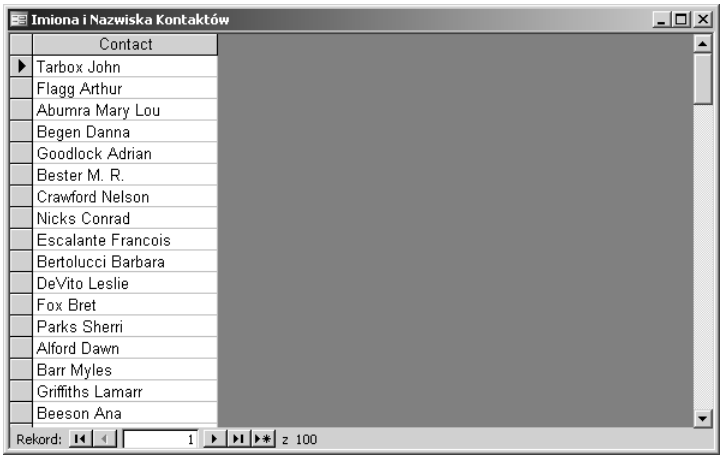
Pokazaliśmy, jak używać właściwości `SelStart` do ustalania pozycji punktu wstawiania. Pójdźmy o krok dalej. Autor spotkał się w swojej pracy z sytuacją, w której użytkownik wymagał prostego sposobu zamieniania kolejności imion i nazwisk, zapisanych w jednym, wspólnym polu. Miała to być operacja wykonywana na całej grupie rekordów. Oczywiście, pojedyncze pole imienia i nazwiska zawiera często drugie imię lub jego inicjał (albo jeszcze inny element). Każdy, kto kiedykolwiek pisał procedurę pracującą z imionami i nazwiskami, wie jak trudno czasem zapewnić obsługę wszystkich możliwości.

Zadanie, o którym mowa, było typowym zadaniem „jednorazowym". Nie miało sensu pisanie długiej, dopracowanej procedury. Wykorzystałem wtedy wymienione wyżej właściwości pól tekstowych i pozostawiłem część pracy samemu użytkownikowi. Oto opis tego rozwiązania.



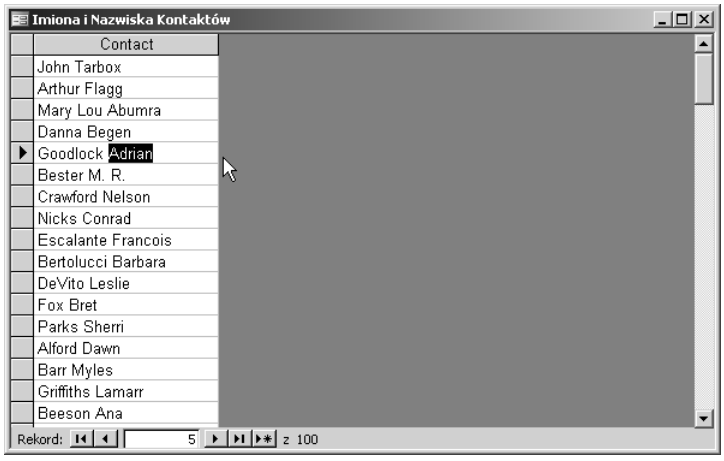
Rysunek 3.7. Nowe dane, które uzupełniły (a nie zastąpiły) wcześniejszy wpis

Użytkownik otrzymał imiona i nazwiska w prostej postaci, takiej jak przedstawiona na rysunku 3.8.



Rysunek 3.8. Lista, która wymaga zmiany kolejności imion i nazwisk

Jego zadaniem było zaznaczenie imienia — albo obu imion, albo imiona i inicjału drugiego imienia itp., — a następnie wciśnięcie klawisza *Tab* lub *Enter*. Ilustruje to rysunek 3.9.



Rysunek 3.9. Użytkownik zaznacza imię

To wszystko! Cały schemat działał dzięki wykorzystaniu właściwości pola w procedurze obsługi zdarzenia `Exit` (*Przy zakończeniu*). W omawianym przykładzie modyfikowane pole miało nazwę `Kontakt`. Istotnym uzupełnieniem była również procedura poprzednio opisana, związana ze zdarzeniem `Enter` (*Przy wejściu*):

```
Private Sub Contact_Enter()
'
'usuwa zaznaczenie
'i umieszcza punkt wstawiania na końcu
'
  Dim text_length As Integer
  text_length = Len(Contact)
  Contact.SelStart = text_length
End Sub

Private Sub Contact_Exit(Cancel As Integer)
'
'jeżeli tekst został zaznaczony, a zaznaczenie
'nie obejmuje całego tekstu, to
'jeżeli zaznaczenie zaczyna się za pierwszym znakiem tekstu, to
'przenosi zaznaczony tekst na początek
'
  Dim new_text As String
  If Contact.SelLength > 0 And _
    Contact.SelLength < Len(Contact) Then
      If Contact.SelStart > 1 Then
        new_text = Contact.SelText & " " & _
          Left(Contact, Len(Contact) - Contact.SelLength)
        Contact.Text = Trim(new_text)
      End If
  End If
End Sub
```

Użytkownik zaznacza imię i dalsze elementy, które mogą mu towarzyszyć (jak drugie imię) i w dowolny sposób wychodzi poza pole tekstowe (wciśnięciem klawisza *Tab* lub *Enter*). Powoduje to wykonanie procedury obsługi zdarzenia wyjścia. Procedura wykonuje proste

sprawdzenie warunków początkowych — czy tekst nie został zaznaczony w całości, czy w ogóle pewien fragment został zaznaczony i czy zaznaczenie nie rozpoczyna się na początku pola.

Właściwe czynności procedury zdarzenia to zamiana pozycji części zaznaczonej i części niezaznaczonej (imienia i nazwiska) i wstawienie nowego wpisu do pola. Rysunek 3.10 przedstawia wynik takiej operacji.

Rysunek 3.10. Lista po zmianie kolejności imion i nazwisk

Procedura wykorzystuje trzy właściwości pola tekstowego: `SelStart`, `SelLength` i `SelText`. To wszystko, czego potrzebujemy do pracy z zaznaczonymi fragmentami pól tekstowych. Gdy porównamy je z równoważnymi właściwościami i metodami do pracy z tekstem — `Text`, `Len`, `Left`, `Right`, `Mid` itp. — zobaczymy, że zapewniane przez nie możliwości są dość duże.

SPOSÓB 21.

Uzupełnianie standardowych list przez użytkowników

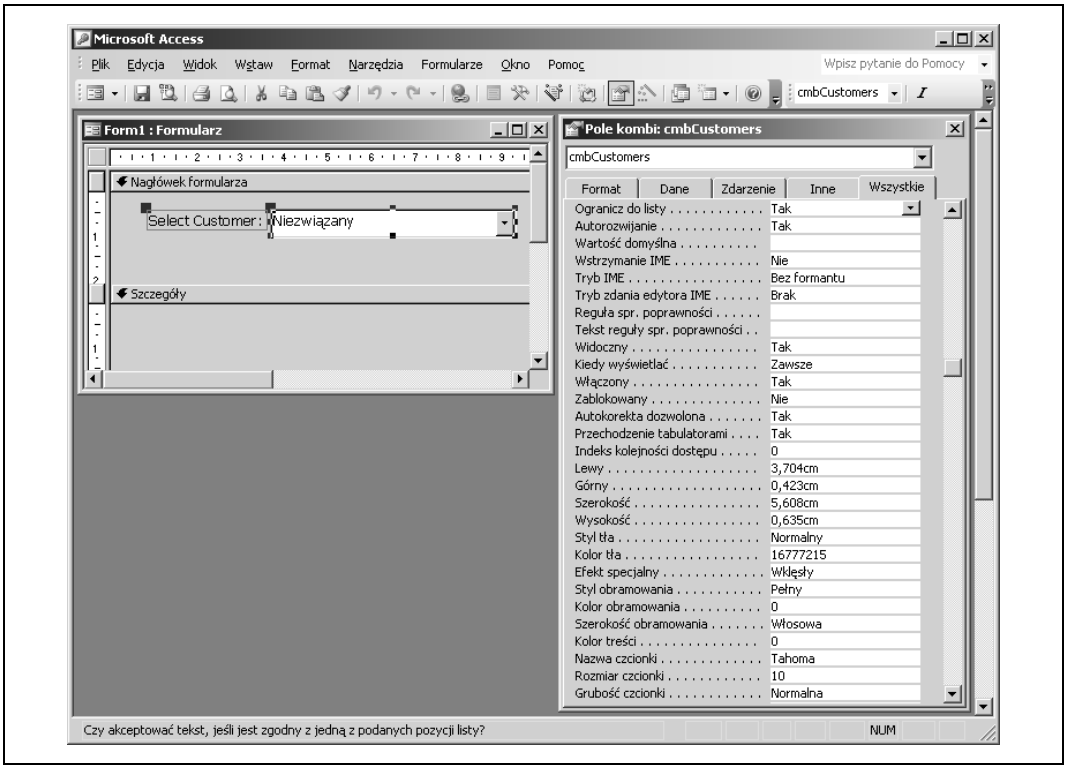
Pozwól użytkownikowi rozbudowywać listę standardowych elementów listy, wprowadzając procedurę obsługującą nowe wartości.

Użytkownicy często wybierają elementy z pewnej listy, korzystając z pola kombi na formularzu. Od czasu do czasu może pojawić się potrzeba wprowadzenia w polu wartości spoza listy. Będzie tak na przykład wtedy, gdy pojawia się nowy klient, którego nie ma jeszcze w tabeli kontrahentów, albo gdy użytkownik wprowadza poprawną wersję elementu listy, który został wpisany z błędem.

Właściwość *Ogranicz do listy* pozwala zdecydować o tym, czy w polu kombi będą dopuszczalne nowe wartości. Jeżeli ma wartość *Nie*, użytkownik może wprowadzać własne wpisy. Rozwiązanie takie ma jedną wadę: jeżeli nowa wartość ma stać się stałym elementem listy, nie zapewni tego samo wpisanie jej w polu kombi.

Jeżeli w aplikacji korzystne będzie umożliwienie użytkownikom trwałego rozbudowywania listy źródłowej pola kombi, musimy zastosować inną metodę. Rozpoczynamy od ustawienia właściwości *Ogranicz do listy* na wartość *Tak* (owszem, takie ustawienie uniemożliwia swobodne wprowadzanie wartości, ale czytajmy dalej…). Sztuka polega na wykorzystaniu zdarzenia `NotInList` (*Przy wartości spoza listy*). Zdarzenie to jest generowane tylko wtedy, gdy właściwość *Ogranicz do listy* ma wartość *Tak*. Właśnie procedura obsługi tego zdarzenia umożliwi automatyczne poszerzenie listy.

Rysunek 3.11 przedstawia formularz w widoku projektu. W formularzu widoczne jest pole kombi. Arkusz właściwości przedstawia właściwości pola kombi. Widać, że właściwość *Ogranicz do listy* ma wartość *Tak*.



Rysunek 3.11. Właściwość Ogranicz do listy ustawiona na wartość Tak

Gdy użytkownik podejmie próbę wprowadzenia w polu nowej wartości, wystąpi zdarzenie `NotInList` (*Przy wartości spoza listy*). Procedura obsługi tego zdarzenia zapewnia dodanie nowej wartości do listy.

Kod rozwiązania

Niezbędny kod nie jest skomplikowany. Wywołanie przekazuje procedurze dwa argumenty: `NewData` (nowe dane) i `Response` (reakcja). Są one elementem standardowej sygnatury procedury:

```
Private Sub cmbCustomers_NotInList(NewData As String, _
        Response As Integer)
  Dim ctl As Control
  Set ctl = Me.cmbCustomers
  Response = acDataErrAdded
  ctl.RowSource = ctl.RowSource & ";" & NewData
End Sub
```

Stała `acDataErrAdded`, przypisana zmiennej `Response`, nakazuje Accessowi zignorowanie ustawienia uniemożliwiającego dodawanie nowych wartości. Po wykonaniu tego przypisania nowe dane (dostępne w zmiennej `NewData`) zostają dołączone do wartości właściwości `RowSource`.

Dalsze usprawnienia

Przedstawione rozwiązanie sprawdza się, o ile wartością właściwości `RowSourceType` (*Typ źródła wierszy*) jest *Lista wartości*. Jeżeli właściwość ta ma wartość *Tabela/Kwerenda*, niezbędna jest procedura umieszczająca nową wartość w wykorzystywanym magazynie danych. Procedura zdarzenia `NotInList` (*Przy wartości spoza listy*) musi więc dołączyć tę wartość do tabeli źródłowej.

Oto przykładowa wersja takiej procedury. Przyjmujemy w niej, że tabela źródłowa ma nazwę tblSposobyDostawy, a wykorzystywane pole to SposóbDostawy:

```
Private Sub cmbShippingMethods_NotInList(NewData As String, _
            Response As Integer)
  Dim new_data As String
  Dim conn As ADODB.Connection
  Set conn = CurrentProject.Connection
  'przed wstawieniem musimy podwoić apostrofy
  new_data = Replace(NewData, "'", "''")
  Response = acDataErrAdded
  conn.Execute "Insert Into " & _
    "tblSposobyDostawy(SposóbDostawy) Values('" & _
      new_data & "')"
End Sub
```

SPOSÓB 22.

Sprawne wypełnianie i sortowanie list

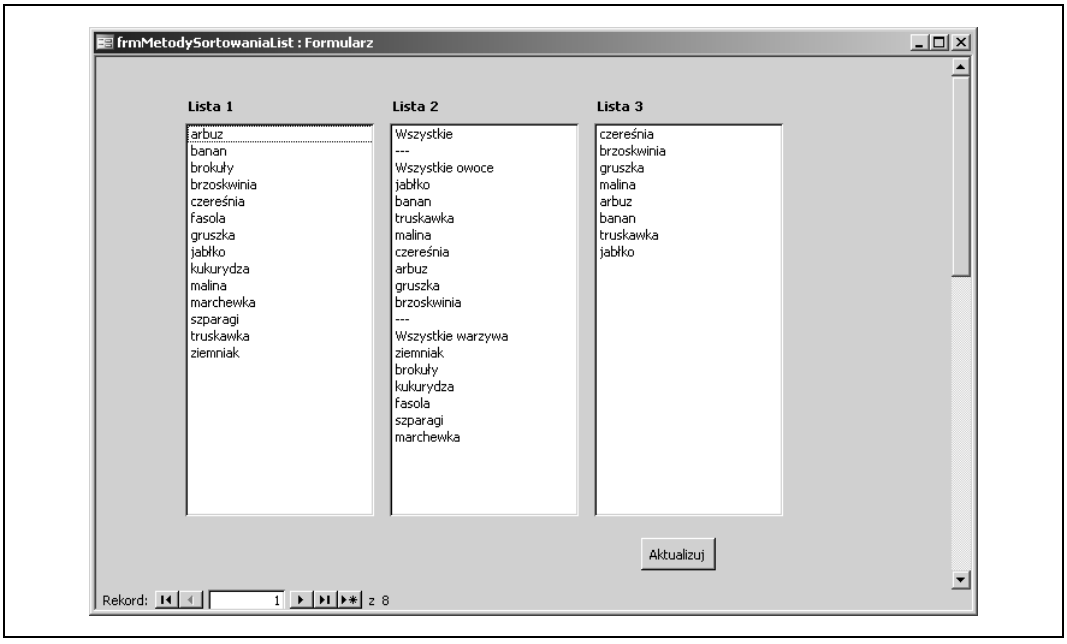
Zastosuj trzy sprytne techniki wypełniania i sortowania formantów listy.

Listy to nieodłączny element formularzy. Oczywiście, nie każdy formularz wymaga listy, ale kiedy można je zastosować, wybór odpowiedniej pozycji jest znacznie prostszy niż wpisywanie wartości. W ten sposób unikamy też drobnych błędów, literówek.

Przedstawimy teraz trzy metody wypełniania i sortowania formantów listy. Kluczowymi elementami wszystkich przykładów są tabele danych i ich struktura. Pokażemy, jak sortować alfabetycznie dane z dwóch źródeł; jak sortować opierając się na wartości klucza; jak sortować, gdy wprowadzamy wartości bezpośrednio w instrukcji SQL, a nawet jak sortować listę według popularności jej elementów! Podstawą naszych działań będzie klauzula języka SQL `Union`.

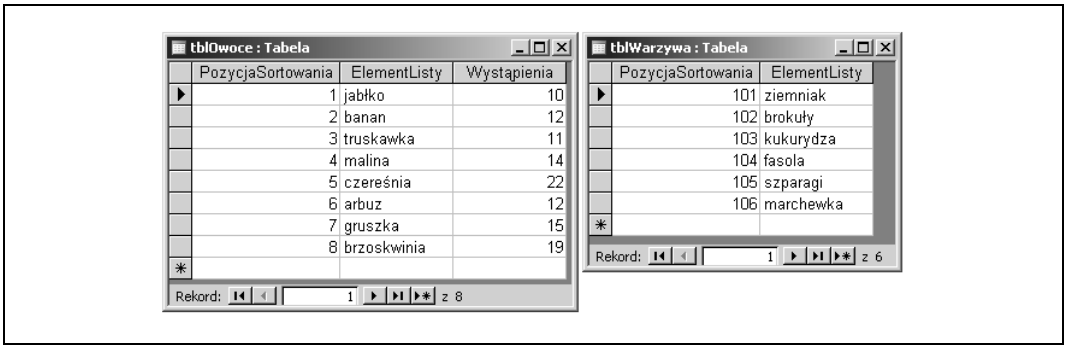
Formularz

Rysunek 3.12 przedstawia formularz z trzema listami, stosownie nazwanymi: Lista 1, Lista 2 i Lista 3.



Rysunek 3.12. Formularz z trzema formantami listy

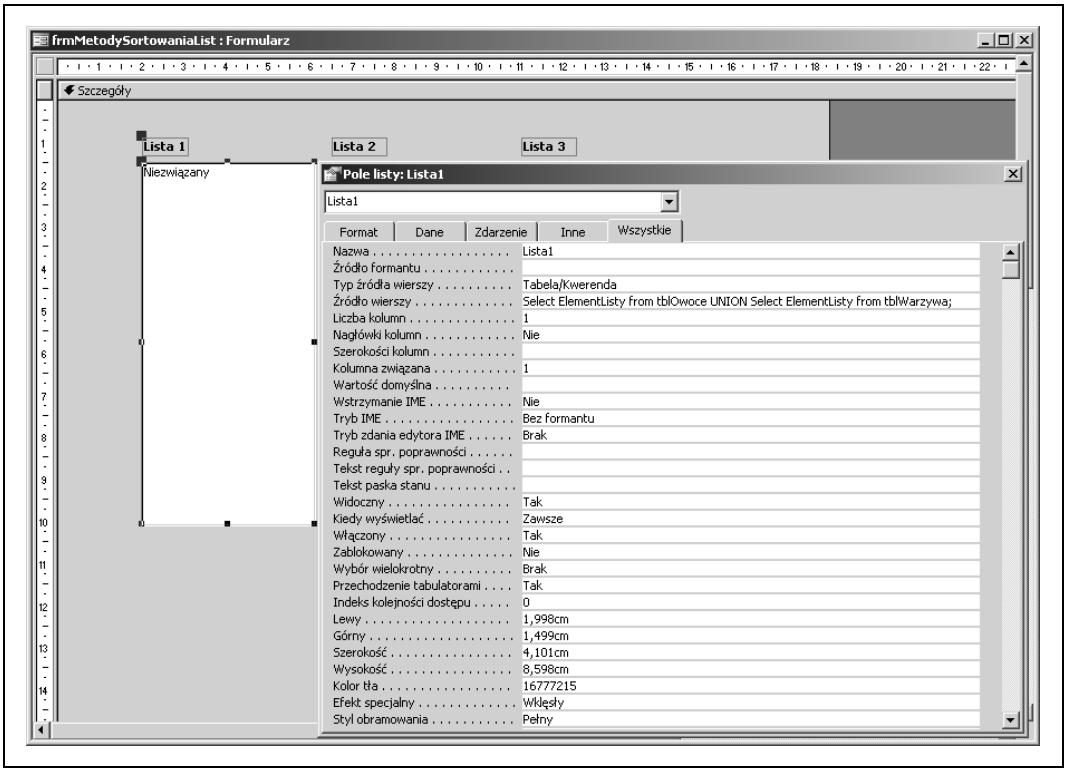
Źródłami danych formantów list są dwie tabele: tblOwoce i tblWarzywa, przedstawione na rysunku 3.13. Zwróćmy uwagę, że mają one dwa wspólne pola: PozycjaSortowania i ElementListy. Jak się przekonamy, będą one miały duże znaczenie.



Rysunek 3.13. Tabele wykorzystywane przez formanty list

Alfabetyczne wypełnianie pola listy danymi z dwóch źródeł

Lista 1 wyświetla wartości z dwóch tabel, posortowane alfabetycznie jako jeden zbiór danych. Formant Lista wykorzystuje jako źródło wierszy obie tabele. Jest to możliwe dzięki połączeniu rekordów obu tabel kwerendą `Union`. Rysunek 3.14 przedstawia formularz w widoku projektu, razem z arkuszem właściwości tej listy.



Rysunek 3.14. Właściwość Źródło wierszy pierwszej listy

Instrukcja SQL zapisana we właściwości *Źródło wierszy* ma następującą postać:

```
Select ElementListy from tblOwoce UNION Select ElementListy from tblWarzywa;
```

Klauzula `Union` powoduje połączenie wartości z obu tabel. Warunkiem jest zgodność ich struktur i typów. Innymi słowy, instrukcja SQL wykorzystuje tylko pole ElementListy obu tabel. Jednakowa liczba pól w każdej instrukcji SQL jest podstawowym wymogiem kwerendy `Union`. Kwerenda taka nie może być wykonana, jeżeli liczba zwracanych przez każdą z kwerend składowych pól jest różna.

W efekcie połączone rekordy zostają posortowane tak, jakby pochodziły z jednego źródła (co, technicznie rzecz biorąc, jest prawdą, bo kwerenda `Union` zwraca jeden zbiór rekordów). Tracimy w ten sposób rozróżnienie między owocami i warzywami — brokuły trafiają na pozycję po bananie, a ziemniak po truskawce.

Jest to prosta metoda prezentacji danych z więcej niż jednego źródła. Jak pokażemy dalej, stosując wiele klauzul `Union` możemy łączyć dane z dowolnej liczby źródeł.

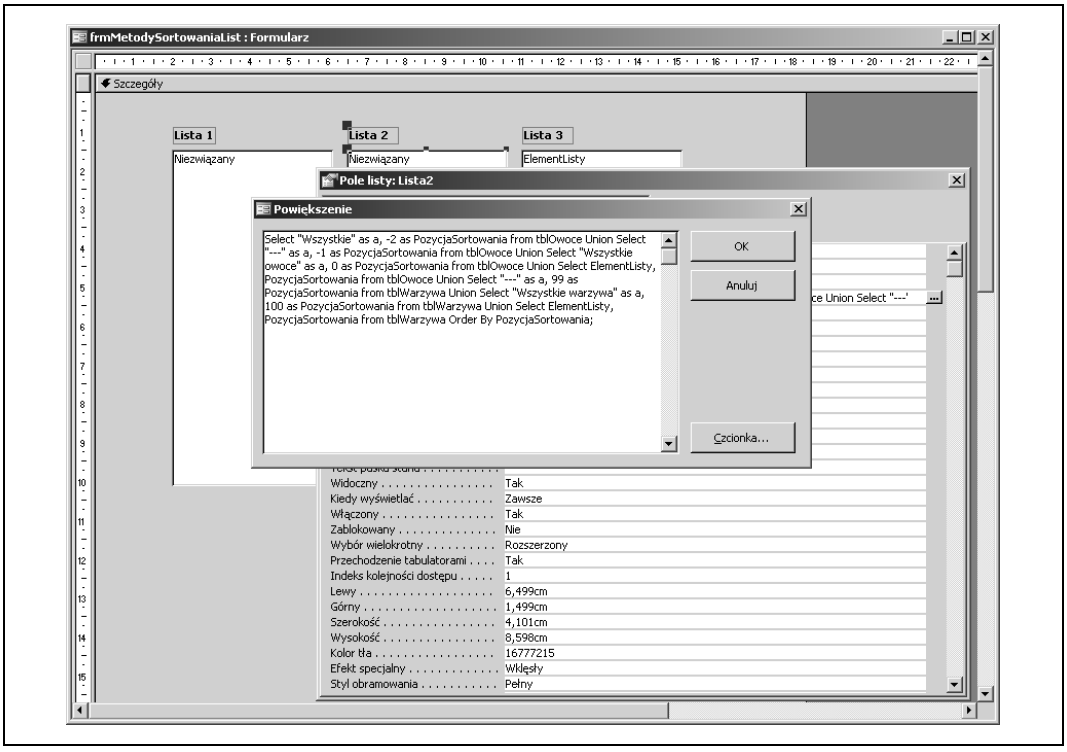
Kontrola sortowania listy wypełnionej danymi z dwóch źródeł

Na rysunku 3.12 Lista 2 przedstawia wynik nieco bardziej wyszukanego sortowania owoców i warzyw. Obie grupy są rozdzielone, a kolejność sortowania każdej z nich nie odpowiada porządkowi alfabetycznemu. Pojawiają się też separatory i wartości, których próżno szukać w tabelach źródłowych: `Wszystkie`, `Wszystkie owoce` i `Wszystkie warzywa`. Jak więc taka lista powstała?

Źródłem wierszy jest ponownie kwerenda `Union`. Wykorzystywane są dwa źródła — tabele tblOwoce i tblWarzywa, — jednak w tym przypadku nie pozwalamy formantowi wymieszać i posortować pozycji alfabetycznie. O kolejności decyduje pole PozycjaSortowania.

Kluczowym czynnikiem są inne zakresy wartości w polu PozycjaSortowania tabeli tblOwoce i w polu PozycjaSortowania tabeli tblWarzywa. Operacja `Union` w rzeczywistości powoduje wymieszanie rekordów z obu źródeł, jednak pole PozycjaSortowania umożliwia powstrzymanie jednolitego sortowania w formancie.

Rysunek 3.15 przedstawia formularz w widoku projektu i arkusz właściwości listy Lista 2. Instrukcja SQL użyta jako właściwość *Źródło wierszy* jest widoczna w oknie *Powiększenie*.



Rysunek 3.15. Właściwość Źródło wierszy drugiej listy

Oto ta sama instrukcja SQL w nieco bardziej uporządkowanej postaci:

```
Select "Wszystkie" as a, -2 as PozycjaSortowania from tblOwoce
Union Select "---" as a, -1 as PozycjaSortowania from tblOwoce
Union Select "Wszystkie owoce" as a, 0 as PozycjaSortowania from tblOwoce
Union Select ElementListy, PozycjaSortowania from tblOwoce
Union Select "---" as a, 99 as PozycjaSortowania from tblWarzywa
Union Select "Wszystkie warzywa" as a,
100 as PozycjaSortowania from tblWarzywa
Union Select ElementListy, PozycjaSortowania from tblWarzywa
Order By PozycjaSortowania;
```

Trochę się tu dzieje. Ogólnie instrukcja SQL łączy elementy z tabel źródłowych z elementami wprowadzonymi bezpośrednio w kodzie SQL. Porządek narzuca pole PozycjaSortowania.

Klauzula `Union` została użyta w tej instrukcji wielokrotnie, z zachowaniem zasady, że każda pojedyncza instrukcja `Select` ma taką samą liczbę pól. W naszym przykładzie ta liczba to 2.

Pierwszym krokiem jest umieszczenie słowa `Wszystkie` na początku listy. Realizuje to instrukcja:

```
Select "All" as a, -2 as PozycjaSortowania
```

To, że słowo `Wszystkie` znajdzie się na początku listy, gwarantuje najniższa wartość pola PozycjaSortowania. Tutaj jest to –2. Ani słowo `All`, ani wartość –2 nie zostały pobrane z tabel. Mimo to ich pozycja jest dopasowana do struktury innych instrukcji `Select` w kodzie SQL. Jest to niezbędne, aby połączyć je z innymi wartościami, pobieranymi z tabel.

Kod SQL wykorzystuje klauzulę `Union` do połączenia danych z tabel z wartościami wprowadzonymi bezpośrednio. Takich wartości jest więcej:

```
Select "Wszystkie" as a, -2 as PozycjaSortowania from tblOwoce
Union Select "---" as a, -1 as PozycjaSortowania from tblOwoce
Union Select "Wszystkie owoce" as a, 0 as PozycjaSortowania from tblOwoce
Union Select "---" as a, 99 as PozycjaSortowania from tblWarzywa
Union Select "Wszystkie warzywa" as a, 100 as PozycjaSortowania from
tblWarzywa
```

Każdy z tych fragmentów prowadzi do wyświetlenia na liście kolejnej wartości: `Wszyst➥kie`, `Wszystkie owoce`, `Wszystkie warzywa` i `---`. Żadna z tych wartości nie pochodzi z tabel. Jednak każda z nich zostaje połączona z pozycją sortowania, która decyduje o ich położeniu w formancie listy.

Przyjrzyjmy się bliżej pozycjom sortowania skojarzonymi z tymi wprowadzonymi bezpośrednio wartościami i pozycjom sortowania elementów w tabelach (rysunek 3.13). Pozycje sortowania warzyw zaczynają się od 101. Stąd też przypisanie pozycji 100 elementowi `Wszystkie warzywa`. Prowadzi to do umieszczenia go bezpośrednio nad wszystkimi nazwami warzyw.

Nie zapominajmy, że lista tego rodzaju, zawierająca różnorodne elementy, wymaga również odpowiednich funkcji, obsługujących potencjalne wybory użytkownika. Jeżeli użytkownik wybierze pojedynczy owoc lub warzywo, można oczekiwać, że dalsza praca aplikacji

będzie naturalna i oczywista. Co się jednak stanie, gdy użytkownik wybierze pozycję `Wszystkie owoce`? Wówczas trzeba uwzględnić w dalszym przetwarzaniu wszystkie wartości z tabeli tblOwoce.

Zwróćmy też uwagę, że wprowadziliśmy separatory (---), zapewniające rozdzielenie fragmentów długiej listy. Jest to bardzo wygodne dla użytkownika przewijającego listę o wielu elementach, bo ułatwia znalezienie potrzebnej podgrupy. Prowadzi jednak do tego, że użytkownik może zaznaczyć sam separator! Musimy więc zadbać o sprawdzanie poprawności dokonanego wyboru i generowanie odpowiednich powiadomień. W typowej aplikacji, jeżeli użytkownik zaznaczy separator, ukazuje się komunikat informujący o tym, że musi wybrać inną pozycję.

Sortowanie elementów listy według popularności

Określenie, które elementy listy użytkownicy będą wybierać najczęściej, nie zawsze jest proste. Można użyć pola PozycjaSortowania do uporządkowania elementów w kolejności, który wydaje się najlepsza. Jest też inny sposób.

Możemy sprawić, że o kolejności elementów będą decydowały czynności użytkownika. Wystarczy, że liczby w polu PozycjaSortowania będą odbiciem „popularności" elementów.

Aby mechanizm taki funkcjonował, aktualizujemy pole sortowania za każdym razem, gdy użytkownik dokonuje wyboru. Rysunek 3.16 przedstawia formularz w widoku projektu i arkusz właściwości listy Lista 3.

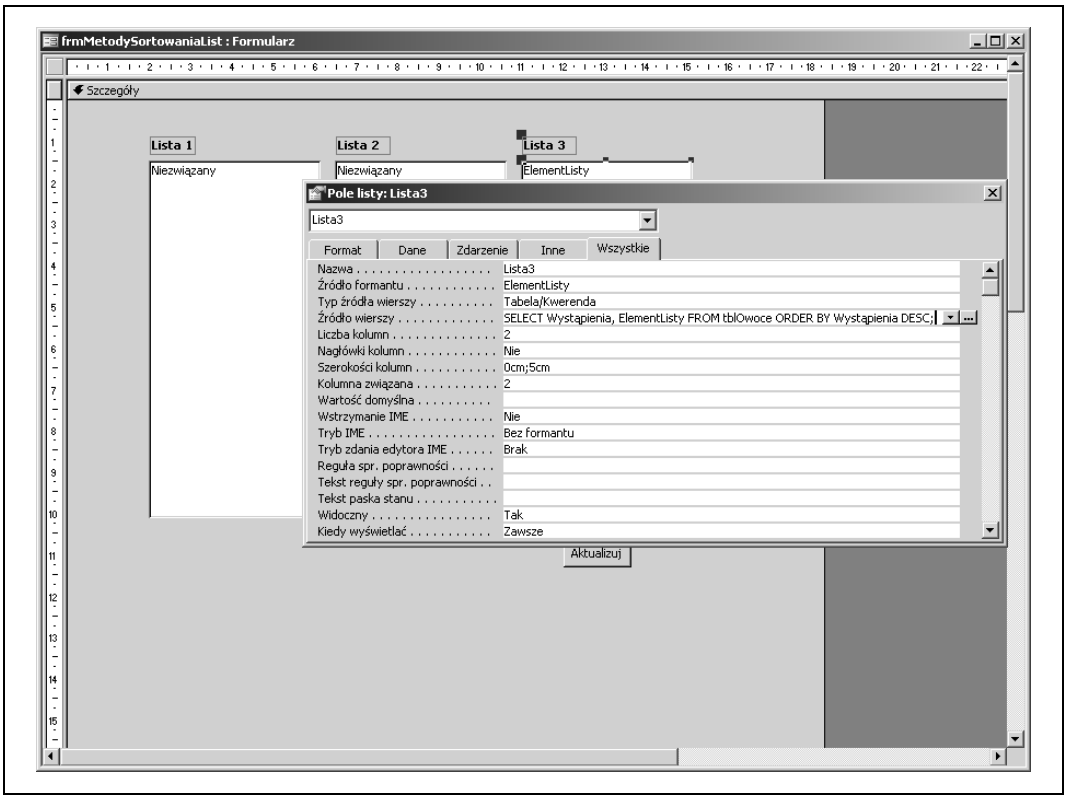
Instrukcja SQL przypisana właściwości *Źródło wierszy* trzeciej listy ma postać:

```
SELECT Wystąpienia, ElementListy FROM tblOwoce ORDER BY Wystąpienia DESC;
```

Źródłem wierszy jest jedna tabela, tblOwoce. Ma ona dodatkowe pole Wystąpienia, wykorzystywane do sortowania. Zwróćmy uwagę, że definicja zbioru rekordów zapewnia sortowanie według tego pola w porządku malejącym.

Aby dodatkowe pole spełniało swoją funkcję, niezbędna jest odpowiednia procedura aktualizowania jego wartości. Aktualizacja będzie wykonywana za każdym razem, gdy przetwarzana jest wybrana z listy wartość. Nasz przykładowy formularz zawiera przycisk. Procedura zdarzenia `Click` tego przycisku odczytuje wartość listy i modyfikuje wartość pola Wystąpienia:

```
Private Sub cmdUpdateCount_Click()
  'pobierz bieżący stan licznika wystąpień
  Dim selected_item_count As Integer
  If Not IsNull(Me.List3) Then
    selected_item_count =
      DLookup("Wystąpienia", "tblOwoce", "ElementListy='" & Me.List3 & "'")
    'zwiększ stan licznika i aktualizuj tabelę
    selected_item_count = selected_item_count + 1
    DoCmd.SetWarnings False
    DoCmd.RunSQL (Update tblOwoce Set Wystąpienia=" &
      selected_item_count & " Where ElementListy='" & Me.List3 & "'")
    Me.List3.Requery
  End If
End Sub
```



Rysunek 3.16. Właściwość Źródło wierszy trzeciej listy

Funkcja `DLookup` wyszukuje bieżącą wartość pola Wystąpienia dla zaznaczonej pozycji listy i zapisuje ją w zmiennej `selected_item_count`. Wartość zmiennej jest zwiększana o 1 i instrukcja SQL zapisuje ją w tabeli. Metoda `Requery` powoduje ponowne wyświetlenie listy na ekranie, w kolejności odpowiadającej zmodyfikowanym danym w kolumnie Wystąpienia.

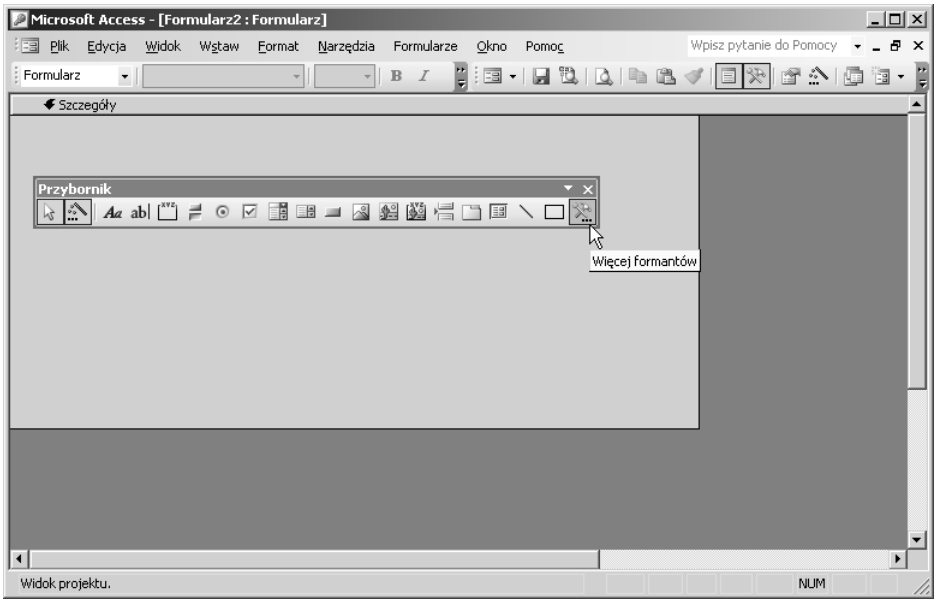
W efekcie częstsze wybieranie pewnych elementów listy będzie powodowało ich stopniowe przesuwanie w górę. Wartości pola Wystąpienia w tabeli tblOwoce na rysunku 3.13 odpowiadają kolejności elementów na rysunku 3.12 — pierwszą pozycję zajmuje czereśnia, ponieważ w polu Wystąpienia jej rekordu jest najwyższa wartość.

SPOSÓB 23.

Dodatkowe formanty formularzy

Odkryj możliwości niestandardowych formantów formularzy.

Projektując aplikacje Accessa mamy okazję dobrze poznać jego standardowy „przybornik". Dostępne w nim formanty są już zapewne Czytelnikowi dobrze znane. Ale czy zainteresowaliśmy się kiedyś ostatnim przyciskiem, tym z etykietą *Więcej formantów*? Widać go na rysunku 3.17.



Rysunek 3.17. Dostęp do dodatkowych formantów

Kliknięciem tego przycisku otwieramy długą listę dodatkowych elementów sterujących. W rzeczywistości większość z nich nie ma praktycznego zastosowania w Accessie. Jednak kilka z nich może być bardzo przydatnych. Warto przede wszystkim zwrócić uwagę na te, których nazwa zaczyna się od „Microsoft". Przyjrzymy się bliżej kilku z nich.

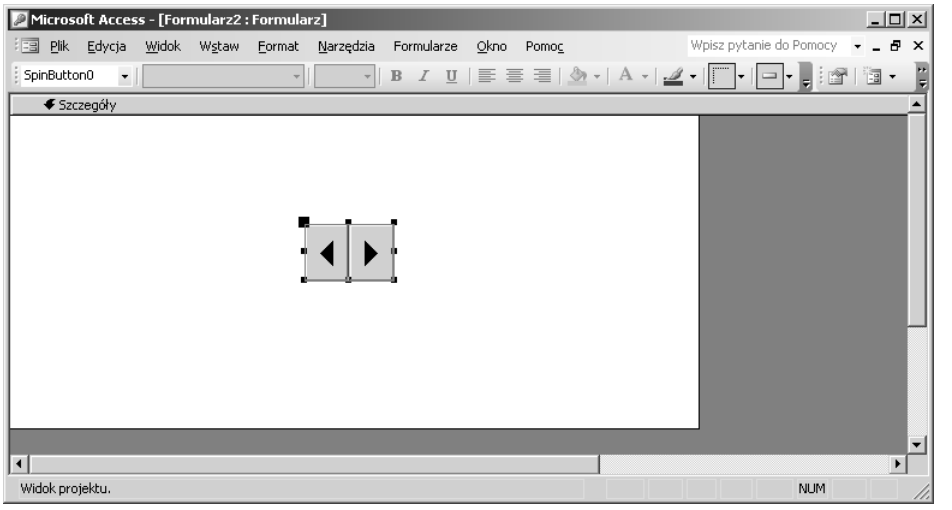
Korzystanie z dodatkowych formantów

Lista formantów zależy od konfiguracji systemu, więc na każdym komputerze jest nieco inna. Można jednak oczekiwać, że zawsze będzie dostępna biblioteka Microsoft Forms (instalowana razem z Microsoft Office).

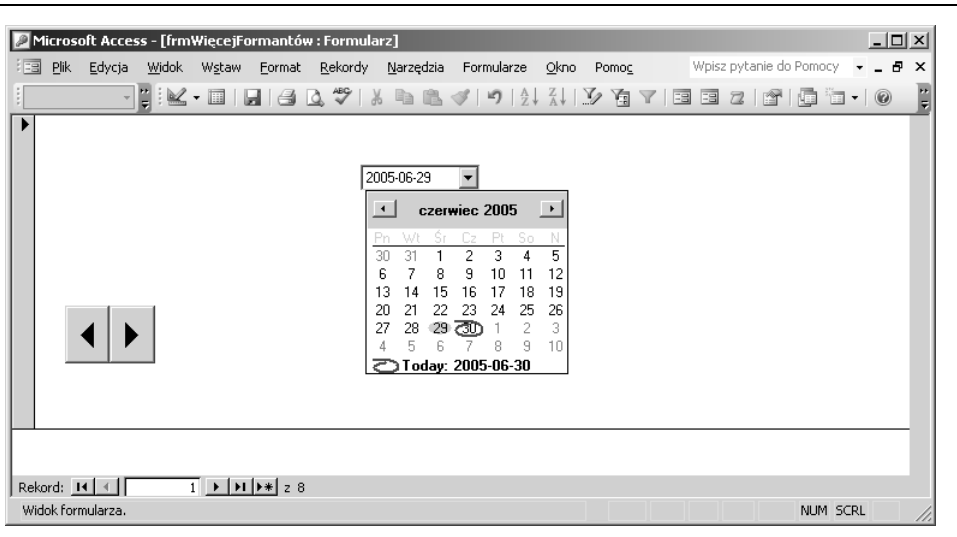
Na rysunku 3.18 widać przykład umieszczonego na formularzu przycisku pokrętła. Wystarczyło tylko wybrać odpowiednią pozycję z listy formantów i zaznaczyć rozmiar przycisku. Przycisk pokrętła to jeden z formantów z biblioteki Microsoft Forms 2.0.

Przycisk pokrętła ma dwie ważne właściwości, wyznaczające początek i koniec zakresu wartości dostępnych użytkownikowi. Pokrętło może więc służyć do ustawiania wartości od 1 do 100 albo od 128 do 133 — odpowiednio do wymagań aplikacji. Wartość ustawianą przez przycisk pokrętła odczytujemy w taki sam sposób jak wartości innych formantów.

Kolejnym przykładem formantu spoza standardowej listy będzie Microsoft Date and Time Picker Control 6.0, przedstawiony na rysunku 3.19. Jest to formant do ustawiania daty i godziny, który ukazuje się w całej okazałości dopiero po kliknięciu pola przypominającego zwiniętą listę. Jest to bardzo wygodne narzędzie. Użytkownicy nie muszą dzięki niemu wprowadzać dat ręcznie, a na formularzu wcale nie ubywa miejsca.



Rysunek 3.18. Przycisk przewijania



Rysunek 3.19. Formant kalendarza

Warto poznawać nowe formanty i wykorzystywać je na równi ze standardowym przybornikiem. Otwierają one drogę do nowych rozwiązań i sprawiają, że korzystanie z aplikacji jest dla użytkownika nie tylko wygodniejsze, ale i przyjemniejsze.

Zobacz również

- „Film w formularzu” **[Sposób 34.]**.
- „Przeglądarka WWW w oknie Accessa” **[Sposób 97.]**.

SPOSÓB 24.

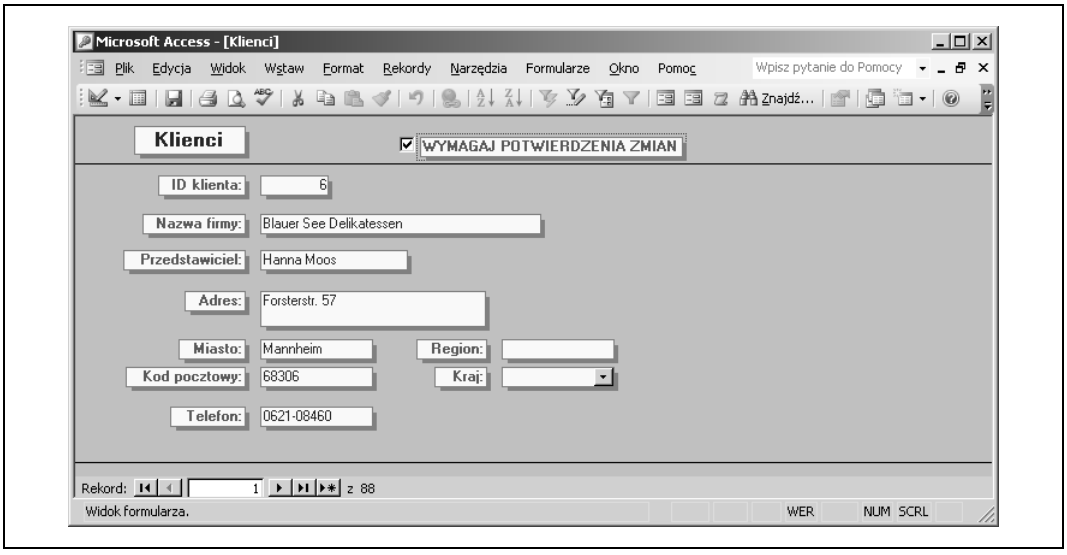
Potwierdzanie zmiany rekordu przed zapisem

Zapewnij użytkownikom możliwość przejrzenia całości wprowadzonych poprawek bezpośrednio przed ich zapisaniem.

Gdy korzystamy z formularza związanego z tabelą, przewijamy listę rekordów, zmieniamy ich zawartość, a każda poprawka jest zapisywana automatycznie. Jest to normalne działanie aplikacji, raczej cenione w codziennej pracy. Bywają jednak przypadki, gdy bardziej wskazane jest odejście od tego schematu i umożliwienie użytkownikowi przejrzenia modyfikacji. Aktualizacja to operacja niszcząca stare dane, warto więc dbać o to, aby nowe były dobrze sprawdzone.

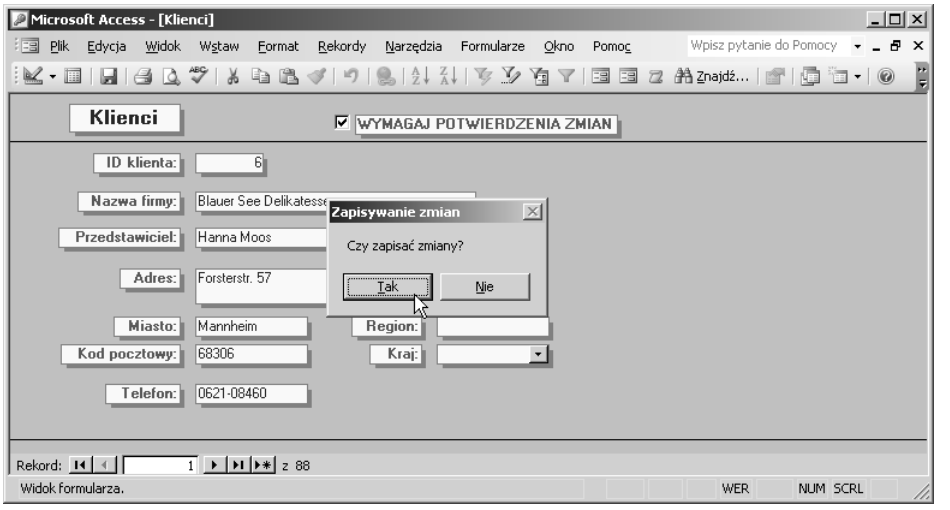
Wprowadzenie takiego dodatkowego kroku znakomicie ułatwia zdarzenie `BeforeUpdate` (*Przed aktualizacją*). Jego procedura może wejść w dodatkowe interakcje z użytkownikiem, prowadzące do potwierdzenia zapisu. Użytkownik może wówczas też odpowiedzieć „nie" i cofnąć zmiany.

Użytkownik powinien mieć możliwość decydowania o tym, czy będzie wyświetlane żądanie potwierdzenia poprawek, ponieważ ostrzeżenia tego rodzaju łatwo mogą stać się irytujące. O tym, czy użytkownik w danym momencie ich oczekuje, czy nie, decydować może wiele czynników. Rysunek 3.20 przedstawia formularz z polem wyboru (w prawym górnym rogu), które pozwala w każdej chwili zmienić decyzję o tym, czy żądania potwierdzenia mają być wyświetlane, czy nie.



Rysunek 3.20. Pole wyboru, które decyduje o tym, czy aktualizacje będą wymagały potwierdzenia

Zdarzenie `BeforeUpdate` następuje tylko wtedy, gdy użytkownik zmienia dane. Sprawdzane jest wówczas najpierw ustawienie pola wyboru. Jeżeli ma wartość `true`, wyświetlane jest okienko dialogowe przedstawione na rysunku 3.21.



Rysunek 3.21. Potwierdzanie aktualizacji

Jeżeli użytkownik kliknie *Tak*, aktualizacja zostanie zatwierdzona. Jeżeli kliknie `Nie`, wykonywane jest polecenie `undo` (*Cofnij*), prowadzące do odrzucenia zmian. Oto kod procedury obsługi zdarzenia:

```
Private Sub Form_BeforeUpdate(Cancel As Integer)
  If Me.chkConfirm = True Then
    proceed = MsgBox("Czy zapisać zmiany?", vbYesNo, _
      "Zapisywanie zmian")
    If proceed = vbNo Then
      DoCmd.RunCommand ac CmdUndo
    End If
  End If
End Sub
```

Najważniejszym elementem jest pozostawienie użytkownikowi decyzji o tym, czy ostrzeżenia mają się pojawiać. Nieustanne żądania potwierdzenia zmian mogą bardzo łatwo prowadzić do zniechęcania. Opcja włączenia potwierdzeń w trakcie pracy z ważnymi danymi i wyłączenia ich przy mniej istotnych czynnościach może być bardzo wygodna.

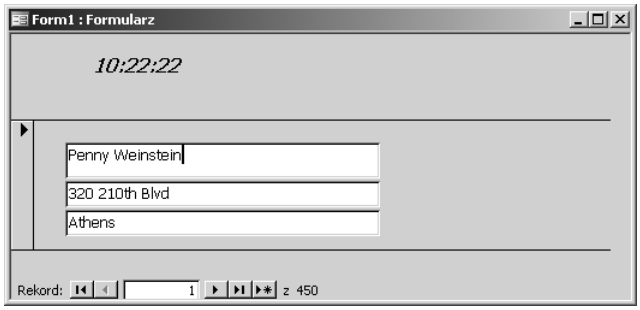
SPOSÓB 25.

Zegar w formularzu

Zapewnij użytkownikowi łatwy dostęp do aktualnej daty i godziny. Najlepiej od razu dla wielu różnych stref czasowych.

Korzystając ze zdarzenia formularza `OnTimer`, możliwości ustawiania czasu oczekiwania na zdarzenie i zegara systemowego, możemy stworzyć na formularzu bardzo praktyczny zegarek.

Rysunek 3.22 przedstawia formularz z zegarem w nagłówku. O tym, że jest on w nagłówku, decyduje fakt, że nie jest to element związany z żadnym rekordem.



Rysunek 3.22. Formularz, który wyświetla informację o godzinie

Gdy zegar nagłówka spokojnie tyka, użytkownik może przeglądać rekordy i zmieniać dane.

Budowanie zegara

Przedstawiony na rysunku zegar utworzymy w bardzo prosty sposób. Rozpoczynamy od umieszczenia na formularzu formantu etykiety. Następnie ustawiamy właściwość *Interwał czasomierza* formularza na 1000 (czyli jedną sekundę). Ostatnią czynnością jest wprowadzenie pojedynczego wiersza kodu w procedurze obsługi zdarzenia `OnTimer` (*Przy cyklu czasomierza*):

```
Me.lblClock.Caption = Format(Now(), "hh:mm:ss")
```

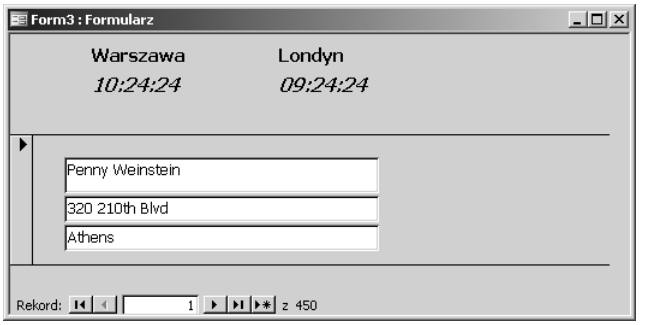
Zakładamy tutaj, że formant etykiety ma nazwę `lblClock`. Funkcja `Now` zwraca bieżącą godzinę, a funkcja `Format` nadaje jej ciągowi znaków pożądaną postać. Określanie formatu nie jest absolutnie niezbędne. Jeżeli ten krok pominiemy, Access wyświetli pełną datę i godzinę. Tutaj zawęziliśmy tę informację do samej godziny: `hh:mm:ss` nakazuje wyświetlenie samej godziny, z dokładnością do sekund.

Dalsze usprawnienia

Wygląd i funkcje zegara można dalej dopracowywać. Jednym z pomysłów może być wyświetlanie godziny dla miast w różnych strefach czasowych. Rysunek 3.23 przedstawia formularz z dwoma zegarami. Jeden wyświetla godzinę w Polsce, drugi — w Wielkiej Brytanii.

W Polsce jest o godzinę później. Aby wprowadzić jednogodzinną różnicę, korzystamy z funkcji `DateAdd`. Oto nowa postać procedury obsługi zdarzenia `OnTimer`:

```
Me.lblClockWarszawa.Caption = Format(Now(), "hh:mm:ss")
Me.lblClockLondyn.Caption = Format(DateAdd("h", -1, Now()), _
    "hh:mm:ss")
```



Rysunek 3.23. Formularz z dwoma zegarami

Funkcja `DateAdd` może dodawać lub odejmować wartość daty. W tym przypadku wartość –1 prowadzi do przesunięcia zegara o godzinę wstecz.

A oto jeszcze jeden pomysł: pozwolić użytkownikowi zmieniać format. Efekt taki osiągniemy stosując zmienną publiczną i zdarzenie `DblClick` (*Przy podwójnym kliknięciu*) formantu etykiety. W momencie otwierania formularza zmienna publiczna, którą nazwiemy `format_type`, otrzyma wartość 1. Jej wartość będzie zwiększana przy każdym podwójnym kliknięciu zegara. Gdy osiągnie 4, powrócimy do ustawienia 1. Procedura zdarzenia `OnTimer` bada wartość tej zmiennej i stosuje odpowiedni format. Oto pełny kod:

```
Option Compare Database
Public format_type As String

Private Sub Form_Open(Cancel As Integer)
  format_type = 1
End Sub

Private Sub Form_Timer()
Select Case format_type
  Case 1
    Me.lblClockCaption = Format(Now(), "hh:mm:ss")
  Case 2
    Me.lblClockCaption = Format(Now(), "hh:mm AMPM")
  Case Else
    Me.lblClockCaption = Format(Now(), "mm/dd hh:mm AMPM")
  End Select
End Sub

Private Sub lblClock_DblClick(Cancel As Integer)
  format_type = format_type + 1
  If format_type = 4 Then format_type = 1
End Sub
```

Teraz użytkownik może podwójnymi kliknięciami zegara dobrać odpowiadający mu sposób wyświetlania godziny. Oczywiście listę dostępnych formantów można rozbudować jeszcze bardziej.