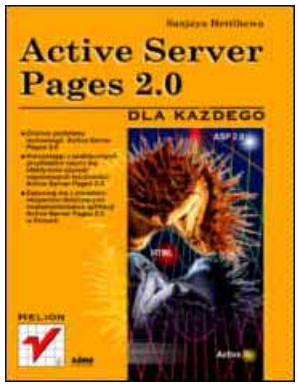
Active Server Pages 2.0 dla każdego



Autor: Sanjaya Hettihewa
Tłumaczenie: Piotr Rajca
ISBN: 83-7197-104-4
Format: B5, 720 stron
Data wydania: 10/1999
Cena książki: 66.00 zł
Przesyłka gratis! Odbiorca pokrywa jedynie koszty pobrania (2,70 zł) w przypadku przesyłki za zaliczeniem pocztowym

Wydawnictwo Helion
ul. Chopina 6, 44-100 Gliwice, POLAND
telefon: (32) 230-98-63, 231-22-19
fax: (32) 230-98-63 w.10
mail: helion@helion.com.pl

Dzięki tej pozycji czytelnik w bardzo krótkim czasie nabędzie umiejętności, które pozwolą rozpocząć efektywną pracę z Active Server Pages 2.0. Książka ukazuje funkcje i działanie ASP 2.0 od podstaw aż do bardziej zaawansowanych funkcji i pojęć. Czytając tę książkę można zrozumieć podstawy tworzenia dynamicznych i interaktywnych stron internetowych, nauczyć się tworzenia aplikacji niezależne od typu przeglądarki internetowej, wkorzystać siłę ActiveX Data Objects (ADO) w tworzeniu internetowych aplikacji baz danych,urozmaicić swoje strony internetowe przy pomocy aplikacji ASP, rozwinąć możliwości ASP dzięki samodzielnemu projektowaniu skryptów i komponentów ASP, opanować bardzo wyszukane techniki interakcji z użytkownikiem, wykorzystujące formularze HTML, Javę, ActiveX, okna dialogowe i pola tekstowe

Zobacz przykładowy rozdział

Spis treści

Jeżeli znasz tę książkę oceń ją

Aktualny cennik książek e-mailem

Książki i "3D" Online

Informacje o nowościach e-mailem

Zamów najnowszy katalog

Rozdział 11.

Tworzenie aplikacji baz danych przy użyciu obiektów danych ActiveX

napisał Duncan Mackenzie

W poprzednim rozdziale zostało przedstawione pojęcie wykorzystywania baz danych w aplikacjach ASP, bazujące na czymś określanym jako ADO. ADO (ActiveX Data Objects – obiekty danych ActiveX) jest zbiorem zewnętrznych obiektów, którymi możesz operować ze swoich stron, dając im tym samym możliwość pracy z bazami danych. Ponieważ zobaczyłeś już przykład sposobu użycia tych obiektów, w tym rozdziale omówimy każdy z nich znacznie bardziej szczegółowo. W tym rozdziale:

Odkryjesz czym jest ADO.

Zapoznasz się z modelem obiektowym ADO.

Użyjesz ADO do stworzenia aplikacji forum użytkowników.

Wykorzystasz zaawansowane możliwości ADO, schematy danych.

Prezentacja ADO

ADO jest grupą obiektów zaprojektowanych w celu dostarczenia prostego interfejsu programistycznego służącego do operowania na bazach danych. W przeszłości stworzono wiele podobnych systemów takich, jak: DAO (Data Access Objects) lub RDO

(Remote Data Objects), które były bardzo szeroko wykorzystywane przez programistów korzystających z języków Visual Basic, Visual C++ oraz osoby używające programu Microsoft Access. Microsoft stwierdził, że te poprzednie systemy były zbyt ograniczone pod względem możliwości operowania na wielu różnych typach danych istniejących w realnym świecie; z tego powodu powstało OLE DB. OLE DB ma wszystkie cechy, których brak DAO – jest szybkie, małe i elastyczne. DAO jest interfejsem Jet – mechanizmu baz danych wykorzystywanym w Access-ie, który nie został zaprojektowany do obsługi innych typów danych. Z drugiej strony, OLE DB może pracować ze wszystkim, włącznie z danymi, które nie są zapisane w tradycyjnym formacie tabelarycznym. Wszystkie te fakty świadczą o tym, że OLE DB jest aktualnie preferowanym narzędziem programistycznym, niestety posiada ono jedną, podstawową wadę, która uniemożliwiła jego powszechne wykorzystanie – OLE DB jest niesłychanie skomplikowane. Aby przezwyciężyć ten problem, Microsoft stworzył ADO, prostą warstwę umieszczoną ponad OLE DB i dostarczającą łatwiejszego interfejsu programistycznego.

Posiadając jedynie trzy główne obiekty oraz kilka pomocniczych kolekcji, ADO jest bardzo proste. Dzięki temu ADO jest łatwe w użyciu i nietrudno się go nauczyć, jednocześnie można przy jego pomocy zrobić niemalże wszystko co można wykonać z bazą danych. Dzięki swojej prostej konstrukcji ADO doskonale nadaje się do wykorzystania wraz z ASP, a zatem stała się ona jedyną metodą obsługi baz danych wykorzystywaną do tworzenia aplikacji ASP. Uważa się także, iż ADO jest lepszym narzędziem do wykorzystania w Visual Basic-u oraz innych środowiskach programistycznych, choć wykorzystywanie ADO poza technologią ASP dopiero teraz zyskuje sobie popularność.

Model obiektowy ADO

ADO posiada jedynie sześć obiektów i dwie kolekcje, jednak spośród tych sześciu obiektów jedynie trzy należy uznać za obiekty główne, których działanie będziesz musiał w pełni zrozumieć. Obiektami tymi są:

`Connection`,

`Command`,

`Recordset`.

Każdy z tych głównych obiektów zostanie wyjaśniony w dalszych częściach tego rozdziału, gdzie zostaną podane listy wszystkich ich właściwości i metod. Przykłady wykorzystania tych obiektów zostaną podane w dalszej części rozdziału, jako część systemu Forum Użytkowników. Zobaczysz je także w następnym rozdziale.

Obiekt Connection

Zgodnie z tym czego dowiedziałeś się w poprzednim rozdziale, każde wykorzystanie bazy danych wymaga obiektu `Connection`. Reprezentuje on rzeczywistą sesję nawiązaną z bazą danych. Zazwyczaj jedynymi metodami tego obiektu, wykorzystywanymi w skryptach ASP, są metody `Open()` i `Close()`. Listing 11.1 przedstawia, w jaki sposób można wykorzystać obiekt `Connection`.

Listing 11.1. *Wykorzystanie obiektu Connection*

```
 1.
 2. <%
 3. Dim Conn             'Nasz obiekt Connection
 4.
 5. Set Conn = Server.CreateObject( "ADODB.Connection" )
 6.
 7. Conn.Open "DSN=WEBSQL;UID=sa;pwd=;"
 8.
 9. ' Tu przychodzi inny kod
10.
11. Conn.Close
12. %>
```

W przedstawionym powyżej kodzie w kilku krokach jest tworzony, wykorzystywany i zamykany obiekt `Connection`. W linii 3. jest deklarowana zmienna `Conn`, w której zostanie zapisany obiekt `Connection`. Takie deklarowanie zmiennej nie jest konieczne, lecz uważa się je za lepszą praktykę programistyczną. W linii 5. przy użyciu metody `Server.CreateObject` jest tworzony faktyczny egzemplarz obiektu `Connection`. Czynność ta musi zostać wykonana przed wywołaniem jakiejkolwiek metody lub właściwości obiektu. Po stworzeniu obiektu, w linii 7., jest otwierana sesja z bazą danych, a jako parametr wywołania metody Open jest używany łańcuch znaków połączenia ze źródłem danych. Ten łańcuch znaków składa się z trzech oddzielnych informacji – nazwy DSN, identyfikatora użytkownika oraz hasła. Wszystkie te informacje można podać oddzielnie, jak pokazano na poniższym przykładzie:

```
Conn.Open "WEBSQL", "sa", ""
```

Dowolny kod operujący na połączeniu powinien zostać umieszczony pomiędzy linią 9. i linią 11.

Obiekt Connection posiada siedem metod przedstawionych w tabeli 11.1.

Tabela 11.1.
Metody obiektu Connection

Metoda	Opis
`Open`	Tworzy sesję z bazą danych. Jest to zazwyczaj pierwsza i niewątpliwie najważniejsza wywoływana metoda.
`Close`	Zamyka sesję, jeśli sesja jest otwarta. Jeśli nie ma otwartej sesji, to metoda zwraca błąd.
`Execute`	Wykonuje polecenie SQL w bazie danych. Metoda ta jest użyteczna przy wykonywaniu czynności, które nie zwracają w wyniku rekordów danych, takich jak usunięcie niektórych rekordów tablicy.
`BaginTrans`	Rozpoczyna transakcję z bazą danych. Więcej informacji na temat transakcji i sposobów ich używania znajdziesz w rozdziale 9., pt. „Używanie komponentów ActiveX stworzonych dla ASP”.

Tabela 11.1.
Metody obiektu Connection – ciąg dalszy

Metoda	Opis
`CommitTrans`	Jedna z dwóch metoda zakończenia transakcji z bazą danych; ta metoda informuje bazę o tym, że cała operacja została zakończona pomyślnie, a wszystkie modyfikacje danych powinny zostać zastosowane.
`RollbackTrans`	Inna metoda zakończenia transakcji; informuje ona bazę danych o wystąpieniu błędów i oznacza, że wszystkie operacje wykonane na bazie od momentu wywołania metody `BeginTrans` powinne zostać wycofane.
`OpenSchema`	Ta zaawansowana metoda jest wykorzystywana raczej rzadko. Zwraca ona zbiór rekordów, którego zawartość zależy od pierwszego argumentu wywołania metody. Pełna lista argumentów metody oraz odpowiadających im wartości nie mogła zostać przedstawiona, jednak przykład jej wykorzystania został podany w sekcji „Praca ze schematami baz danych" znajdującej się w dalszej części tego rozdziału.

Obiekt `Connection` posiada także dziewięć właściwości; te najbardziej popularne i użyteczne zostały przedstawione w tabeli 11.2.

Tabela 11.2.
Popularne i przydatne właściwości obiektu Connection

Właściwość	Opis
`ConnectionString`	Ta wartość zawiera informacje potrzebne do stworzenia sesji z bazą danych; może ona zostać przekazana bezpośrednio do wywołania metody `Open` (i tak się zazwyczaj dzieje).
`ConnectionTimeout`	Określa, jak długo metoda `Open` będzie oczekiwała na otrzymanie danych, zanim zgłosi błąd.
`CommandTimeout`	Ta wartość jest używana do określenia, jak długo polecenie wydane sesji z bazą danych będzie wykonywane, zanim zostanie wygenerowany błąd przekroczenia limitu czasu.
`State`	Ta właściwość zwraca jedną z dwóch wartości, określającą czy połączenie z bazą danych jest otwarte (`1`) czy też zamknięte (`0`).
`Attributes`	Właściwość zaawansowana; zwracana przez nią wartość określa możliwości udostępniane przez bazę danych (jak na przykład obsługę transakcji). Nie jest ona wymagana przy tworzeniu prostych systemów.

Za pomocą obiektu `Connection` można także uzyskać dostęp do dwóch kolekcji, przedstawionych w tabeli 11.3.

Obiekt `Connection` jest tworzony i otwierany na samym początku pracy z bazą danych, jednak przeważnie nie jest on używany w pojedynkę. Do faktycznego manipulowania bazą danych jest wykorzystywany przeważnie przynajmniej jeden z przedstawionych poniżej obiektów. Obiekt Command wykorzystuje obiekt `Connection` poprzez swoją właściwość o nazwie `ActiveConnection`.

Tabela 11.3.
Kolekcje obiektu Connection

Kolekcja	Opis
`Errors`	Jest to kolekcja obiektów `Error`, z których każdy odpowiada jednemu błędowi wygenerowanemu w czasie trwania tej sesji z bazą danych.
`Properties`	Tę kolekcję posiada każdy obiekt ADO; udostępnia ona pod postacią kolekcji wszystkie właściwości obiektu przedstawione w tabeli 11.2.

Obiekt Command

Obiekt `Command` reprezentuje jedno polecenie wydane połączeniu z bazą danych takie jak polecenie SQL lub procedura osadzona (ang. stored procedure). Obiekt został zaprojektowany do lepszej obsługi wykonywania poleceń tego typu, niż umożliwia to metoda `Execute` obiektu `Connection`. Obiekt `Command` posiada grupę właściwości, metod i kolekcji, które zostały przedstawione w tabelach 11.4, 11.5 i 11.6.

Tabela 11.4.
Właściwości obiektu Command

Właściwość	Opis
`ActiveConnection`	Każdy obiekt `Command` pracuje na skojarzonym z nim połączeniu z bazą danych. Ta właściwość pobiera lub ustawia obiekt `Connection` wykorzystywany przez polecenie.
`CommandText`	Ta właściwość przechowuje polecenie SQL lub procedurę osadzoną, która ma zostać wykonana przez polecenie.
`CommandTimeout`	Ta właściwość ma takie samo przeznaczenie co analogiczna właściwość obiektu `Connection`.
`CommandType`	Ta wartość informuje źródło danych o naturze tekstu polecenia, jaki został mu przekazany. Właściwość może posiadać jedną z wielu różnych wartości, jednak najczęściej spotykane są `Text` (`1`) (tekst) oraz `Stored Procedure` (`4`) (procedura osadzona).
`Prepared`	Właściwość kontroluje, czy przed wykonaniem polecenia zostanie stworzona jego skompilowana wersja.

Tabela 11.5.
Metody obiektu Command

Metoda	Opis
`CreateParameter`	Każdy obiekt `Command` posiada kolekcję obiektów `Parameter`. Ta metoda służy do dodania nowego obiektu do tej kolekcji. Więcej informacji na ten temat znajdziesz w tabeli 11.6, gdzie została przedstawiona kolekcja `Parameters`.
`Execute`	Podstawowa metoda obiektu `Command`. Powoduje ona wykonanie polecenia w oparciu o dostarczyciela informacji. Metoda ta może zwrócić obiekt typu `Recordset`.

Tabela 11.6.
Kolekcje obiektu Command

Kolekcja	Opis
`Properties`	Jak w każdym obiekcie ADO, także i tutaj kolekcja `Properties` zawiera wszystkie właściwości obiektu oraz odpowiadające im wartości.
`Parameters`	Jest to kolekcja obiektów `Parameter`. Odpowiadają one parametrom istniejącym w poleceniu SQL bądź w procedurze osadzonej reprezentowanej przez dany obiekt `Command`.

Ogólnie rzecz biorąc, obiekt `Command` używany jest do wykonywania bardziej złożonych operacji na bazach danych; proste zapytania oraz operacje na rekordach są zazwyczaj realizowane przy użyciu, opisanego w następnej sekcji, obiektu `Recordset`.

Obiekt Recordset

Każda grupa rekordów – niezależnie od tego czy jest ona wynikiem wykonania zapytania czy też stanowi całą zawartość tabeli – jest reprezentowana przez obiekt `Recordset`. Tego obiektu będziesz używał niemal do wszystkich operacji na bazach danych, jest to także najbardziej złożony obiekt ADO. Jednak w większości wypadków jest on wykorzystywany w bardzo prosty sposób. Listing 11.2 demonstruje popularne wykorzystanie obiektu `Recordset` – pobieranie i wyświetlanie wyników zapytania.

Listing 11.2. *Stosowane zbiorów rekordów i obiektu Recordset*

```
 1. <%
 2. Dim Conn
 3. Dim RSAutorzy
 4. Dim SQL
 5.
 6. Set Conn = Server.CreateObject( "ADODB.Connection" )
 7. Set RSAutorzy = Server.CreateObject( "ADODB.Recordset" )
 8. Conn.Open "WEBSQL;UID=sa;PWD=;"
 9. SQL = "SELECT * FROM Autorzy"
10. RSAutorzy.Open SQL, Conn
11.
12. Do While Not RSAutorzy.EOF
13.   Response.Write "<P>" & RSAutorzy("Au_Imie") & "</P>"
14.   Response.Write vbcrlf
15.   RSAutorzy.MoveNext
16. Loop
17.
18. RSAutorzy.Close
19. Conn.Close
20. %>
```

W przedstawionym powyżej fragmencie kodu, w liniach 2. – 4. definiowane są wykorzystywane zmienne. Jeszcze raz powtórzę, że nie jest to konieczne, lecz stanowi dobry zwyczaj programistyczny. W linii 6. jest tworzony egzemplarz obiektu `Connection`, a w linii 7. egzemplarz obiektu `Recordset`; oba te obiekty będą wykorzysty-

wane w dalszej części kodu. Następnie, w linii 8., jest otwierany obiekt `Connection`, który będzie potrzebny do otwarcia obiektu `Recordset`. W linii 9. polecenie SQL zostaje zapisane w zmiennej, która jest następnie użyta w linii 10., w wywołaniu metody `Open`. W wywołaniu tej metody są podawane dwa argumenty – polecenie SQL oraz obiekt `Connection`. Po otworzeniu zbioru rekordów, w liniach 12. – 16. jest przedstawiony najbardziej popularny sposób operowania na nim. Do przetworzenia całej zawartości zbioru rekordów wystarczy jedna prosta pętla. W tym wypadku, w linii 12. umieszczony został warunek pętli – `RSAutorzy.EOF` – wartość tej właściwości przyjmie wartość `TRUE` (prawda) gdy zostanie pobrany ostatni rekord zbioru rekordów. Linie 12. i 13. tworzą linię kodu HTML zawierającą wartość pola `Au_Imie` zbioru rekordów. Na samym końcu znajduje się niesłychanie ważna linia – `RSAutorzy.MoveNext` – która przesuwa wskaźnik rekordów na kolejny rekord, zapobiegając tym samym powstaniu nieskończonej pętli.

Oprócz kilku prostych właściwości i metod przedstawionych w powyższym przykładzie, obiekt `Recordset` zawiera wiele innych użytecznych właściwości i metod, o których powinieneś wiedzieć. Metody, właściwości i kolekcje obiektu `Recordset` zostały przedstawione w tabelach 11.7, 11.8 oraz 11.9.

Tabela 11.7.
Metody obiektu Recordset

Metoda	Opis
`Open`	Jest to metoda, którą zazwyczaj będziesz wywoływał w pierwszej kolejności; pozwala ona na zainicjalizowanie obiektu i wypełnienie go poszukiwanymi wartościami. Możesz podać obiekt `Connection`, który będzie wykorzystywany z tym zbiorem rekordów lub parametry połączenia. Jeśli nie podasz obiektu Connection, to dla każdego nowo otworzonego zbioru rekordów będzie tworzone nowe połączenie.
`Close`	Jest to metoda, którą zazwyczaj będziesz wywoływał na samym końcu; niszczy ona wszystkie dane przechowywane w obiekcie i pozwala na ponowne wywołanie metody `Open`.
`AddNew`	Tworzy nowy, pusty rekord.
`Update`	Zapisuje w bazie danych zmiany, jakie wprowadziłeś w aktualnym rekordzie.
`CancelUpdate`	Usuwa zmiany, jakie zostały wprowadzone w aktualnym rekordzie, jeśli nie została wcześniej wywołana metoda `Update`.
`UpdateBatch`	Niektórzy dostarczyciele danych pozwalają na wykorzystanie zbiorów rekordów wykonujących wiele operacji aktualizacji w tym samym czasie, zapisując poszczególne modyfikacje dokonane do momentu wywołania tej metody.
`CancelBatch`	Usuwa oczekujące na obsługę wsadowe zadania aktualizacji. Domyślnie są usuwane wszystkie oczekujące zadania aktualizacji, jednak możesz podać argument, który ograniczy usuwanie do aktualnego rekordu lub rekordów zgodnych z podanym filtrem.

Tabela 11.7.
Metody obiektu Recordset – ciąg dalszy

Metoda	Opis
`Move`	Ta metoda pozwala na przesunięcie się o określoną ilość rekordów do przodu lub do tyłu w zbiorze rekordów. Przesunięcie może odbywać się względem aktualnego rekordu (jeśli nie zostanie podany drugi parametr wywołania metody) lub względem dowolnego innego rekordu zbioru rekordów.
`MoveNext`	Metoda powoduje przejście do kolejnego rekordu w zbiorze rekordów. W momencie, gdy wywołasz tę metodę, znajdując się poza ostatnim rekordem zbioru (gdy wartość właściwości `EOF` wynosi `true`), jest generowany błąd.
`MovePrevious`	Przeciwieństwo metody `MoveNext`; powoduje przejście do rekordu poprzedzającego aktualny rekord. Jeśli aktualnym rekordem jest pierwszy rekord zbioru (gdy wartość właściwości `BOF` wynosi `true`), generowany jest błąd.
`MoveLast`	Ta metoda powoduje przejście do ostatniego rekordu zbioru. Jeśli aktualnym rekordem jest właśnie ostatni rekord zbioru, to wykonanie tej metody nie daje żadnego rezultatu.
`MoveFirst`	Jest to metoda podobna do metody `MoveLast`; powoduje ona przejście do pierwszego rekordu zbioru, jednak nie daje żadnych efektów jeśli aktualnym rekordem jest pierwszy rekord zbioru.
`NextRecordset`	Możliwe jest wydanie polecenia, które będzie zwracało więcej niż jeden zbiór rekordów. Może się to zdarzyć przy wywoływaniu procedur osadzonych lub przy korzystaniu ze złożonych poleceń SQL takich jak to przedstawione poniżej: `SELECT ID FROM Goscie; SELECT * FROM KartaZakupowa` Ta metoda powoduje wykonanie następnego polecenia w grupie i zwraca jego wyniki.
`Save`	Ta metoda pobiera jako argument nazwę pliku, a następnie zapisuje w nim aktualny zbiór rekordów. Rekordy te mogą zostać załadowane w przyszłości poprzez wywołanie metody `Open` i podanie pliku jako źródła danych.
`Requery`	Ta metoda jest ekwiwalentem zamknięcia i ponownego otworzenia zbioru rekordów – powoduje ona ponowne wykonanie polecenia. Jeśli aktualnie podane kryteria spełniają więcej rekordów niż poprzednio, to nowe rekordy także będą dostępne w zbiorze.
`Resync`	Nie należy mylić tej metody z metodą `Requery`. Ta metoda jedynie porównuje dane znajdujące się w rekordach z danymi przechowywanymi na serwerze i uaktualnia wartości w zbiorze rekordów. Argument opcjonalny pozwala na synchronizację wyłącznie aktualnego rekordu lub rekordów spełniających podane kryteria.
`Delete`	Ta metoda usuwa aktualny rekord.
`GetRows`	Ta metoda zwraca dwuwymiarową tablicę wartości typu variant, zawierającą pary pole-wartość. Dodatkowe argumenty pozwalają na określenie ile rekordów ma zostać pobranych w taki sposób oraz od którego rekordu należy rozpocząć pobieranie. Domyślnie są pobierane wszystkie rekordy z aktualnego zbioru.

Tabela 11.7.
Metody obiektu Recordset – ciąg dalszy

Metoda	Opis
`Clone`	Ta metoda tworzy zbiór rekordów zawierający kopię aktualnych danych.
`Supports`	Ta metoda pozwala Ci na określenie, jakie możliwości funkcjonalne są udostępniane przez zbiór rekordów, chodzi tu o możliwości takie jak: dodawanie nowego rekordu, przechodzenie do poprzedniego rekordu, itp.

Tabela 11.8.
Właściwości obiektu Recordset

Właściwość	Opis
`EOF`	Koniec pliku; pozwala na określenie, czy aktualny rekord znajduje się poza ostatnim rekordem zbioru rekordów. Ta właściwość jest powszechnie stosowana do określenia, kiedy należy zakończyć pętlę obsługującą rekordy zbioru rekordów.
`BOF`	Początek pliku; właściwość podobna do EOF zwraca jednak wartość `true`, jeśli aktualny rekord znajduje się przed pierwszym rekordem zbioru. Jeśli obie właściwości – `EOF` i `BOF` – mają wartość `true`, to oznacza to, że zbiór rekordów jest pusty.
`MaxRecords`	Ta właściwość kontroluje, ile rekordów jest zwracanych jako wynik zapytania. Jej wartość można określić wyłącznie przed wywołaniem metody `Open`, kiedy zbiór rekordów jest zamknięty.
`AbsolutePosition`	Tę właściwość można wykorzystać do określenia pozycji aktualnego rekordu względem początku pliku.
`PageSize`	Ta właściwość współpracuje z właściwościami `AbsolutePage` oraz `PageCount` i wraz z nimi umożliwia podzielenie zbioru rekordów na logiczne strony. Ta właściwość zwraca bądź ustawia ilość rekordów mieszczących się w jednej logicznej stronie.
`PageCount`	Ta właściwość zwraca ilość logicznych stron w zbiorze rekordów. Jest ona przeznaczona tylko do odczytu, lecz jej wartość zmienia się wraz ze zmianą wartości właściwości `PageSize`.
`AbsolutePage`	Ta właściwość określa logiczną stronę, na której znajduje się aktualny rekord. Właściwość tę można ustawić, co powoduje przeniesienie aktualnego rekordu na pozycję pierwszego rekordu podanej strony.
`CursorType`	Ta właściwość zwraca jedną z kilku wartości: `Forward Only` (`0`), `Keyset` (`1`), `Dynamic` (`2`) bądź `Static` (`3`). Jej wartość można określić wyłącznie w momencie, gdy zbiór rekordów jest zamknięty, a służy ona do określania sposobu, w jaki zbiór rekordów ma zostać otworzony. Jej wartość można także podać jako argument wywołania metody `Open`. Opis każdego typu kursora oraz jego znaczenie zostało wyjaśnione w dalszej części rozdziału.

Tabela 11.8.
Właściwości obiektu Recordset – ciąg dalszy

Właściwość	Opis
`CacheSize`	Ta właściwość określa ilość rekordów, która jest jednorazowo pobierana od dostarczyciela danych. Jej domyślna wartość wynosi `1`, co oznacza, że tylko aktualny rekord jest przechowywany w pamięci podręcznej. Jeśli wartość właściwości będzie wyższa, to rekordy przechowywane w pamięci podręcznej niekoniecznie muszą odpowiadać rzeczywistym danym przechowywanym w bazie, można je jednak odświeżyć za pomocą metody `Resync`.
`Recordcount`	Ta właściwość jest powszechnie wykorzystywana, jednak nie zawsze jej wartość jest poprawna lub w ogóle dostępna. Jeśli zbiór rekordów „nie wie", ile rekordów w nim znajduje się lub nie jest w stanie określić ich ilości, to wartość tej właściwości wynosi `-1`. Ogólnie rzecz biorąc, wartości tej właściwości nie można uznać za pewną, aż do momentu gdy zostanie odwiedzony ostatni rekord zbioru (`MoveLast`).
`Source`	Ta właściwość zawiera tekst polecenia SQL, jakie ma zostać wykonane (lub nazwę procedury osadzonej); jej wartość można podać bezpośrednio przed otworzeniem zbioru rekordów lub jako argument wywołania metody `Open`.
`EditMode`	Określa stan aktualnego rekordu i może przyjmować jedną z następujących wartości: `None` (`0`) (brak), `Edit In Progress` (`1`) (edycja w toku), `Add In Progress` (`2`) (dodawanie w toku) bądź `Deleted` (`3`) (usunięty).
`ActiveConnection`	Ta właściwość służy od zwracania lub ustawiania obiektu `Connection` wykorzystywanego przez aktualny zbiór rekordów.
`LockType`	Określa, jaki rodzaj blokowania został narzucony na aktualnie edytowany rekord. Wartość może posiadać jedną z następujących wartości: `Read-Only` (`1`) (tylko do odczytu), `Pessimistic` (`2`) (pesymistyczna), `Optimistic` (`3`) (optymistyczna), `Batch Optimistic` (`4`) (wsadowa optymistyczna). Znaczenia poszczególnych typów blokad zostały podane w dalszej części rozdziału.
`Bookmark`	Jeśli zbiór rekordów umożliwia wykorzystywanie zakładek (co możesz sprawdzić za pomocą metody `Supports`), to będziesz mógł zapisać wartość tej właściwości w zmiennej. W przyszłości będziesz mógł wrócić bezpośrednio do tego samego rekordu, przypisując tej właściwości wartość zapisaną w zmiennej. Wartość właściwości jest unikalna dla danego zbioru rekordów. Dwa identyczne rekordy w różnych zbiorach nie mogą mieć tej samej wartości właściwości `Bookmark`, jednak wszystkie rekordy obiektu `Recordset` stworzonego za pomocą metody `Clone` będą miały taką samą wartość tej właściwości jak ich oryginały.
`Filter`	Ta właściwość określa warunek, jaki będzie sprawdzany dla wszystkich rekordów aktualnie znajdujących się w zbiorze. W skład zbioru rekordów wejdą tylko te rekordy, które spełnią podane kryteria. Kryteria te mogą przybierać postać *nazwaPola = wartość*, np.: `("IDGoscia = 123")`.
`Status`	Zwraca aktualny stan rekordu z uwzględnieniem wszelkich procedur masowych oraz procedur osadzonych.

Tabela 11.8.
Właściwości obiektu Recordset – ciąg dalszy

Właściwość	Opis
`CursorLocation`	Ta wartość nie ma zastosowania w ASP, gdyż określa ona czy kursor został stworzony na serwerze, czy też na komputerze użytkownika. Przy wykorzystaniu ADO w ASP kursor zawsze jest tworzony na serwerze.
`MarshalOptions`	Podobnie jak w przypadku właściwości CursorLocation, także i ta właściwość ma wpływ na sposób w jaki zbiór rekordów jest przenoszony pomiędzy serwerem a komputerem użytkownika. W przypadku ASP nie jest ona wykorzystywana.

Tabela 11.9.
Kolekcje obiektu Recordset

Kolekcja	Opis
`Properties`	Jak w każdym obiekcie ADO, także i tutaj kolekcja `Properties` zawiera wszystkie właściwości obiektu oraz odpowiadające im wartości.
`Fields`	Zawiera grupę obiektów `Field`, po jednym dla każdej kolumny zbioru rekordów. Ta kolekcja jest przydatna do określania wszystkich kolumn zbioru rekordów, gdy nie są znane ich nazwy i położenie.

Wykorzystanie ADO do stworzenia aplikacji Forum Użytkowników

Aby pokazać, w jaki sposób należy używać obiektów omówionych w poprzedniej części rozdziału, w tej sekcji zostanie przedstawiony kompletny kod forum użytkowników. Ten system będzie wykorzystywać bazę danych Microsoft Access, jednak ze względu na to, iż połączenie z danymi jest realizowane poprzez ODBC, wykorzystanie innej bazy danych (jak na przykład SQL Server) zajmuje jedynie kilka minut. System zostanie stworzony w sposób jak najprostszy, bez wykorzystania żadnych obrazków ani wymyślnych sposobów formatowania. Taki projekt systemu został podyktowany chęcią zapewnienia jak największej prostoty kodu, a nie dlatego, że nie jest on wymagany.

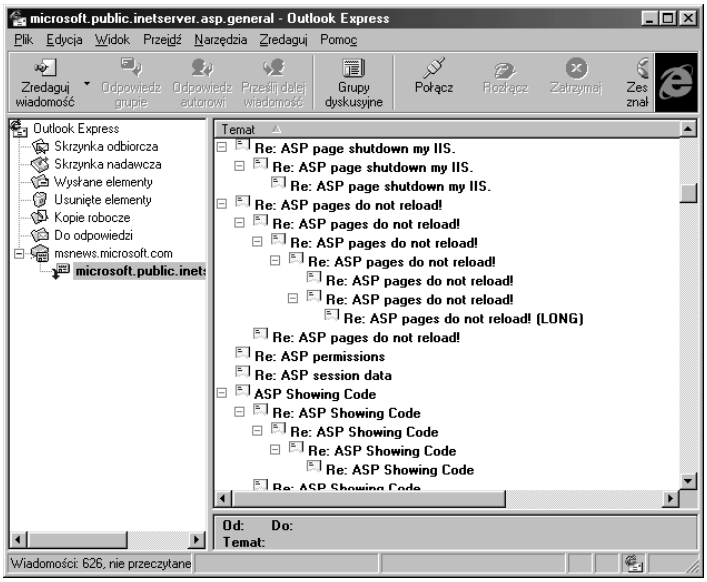
Bazę danych oraz cały kod ASP możesz znaleźć na witrynie WWW Wydawnictwa HELION, pod adresem:

`ftp://ftp.helion.com.pl/przyklady/asp.zip`.

Po skopiowaniu pliku archiwum, rozpakuj go, a następnie umieść na serwerze całą zawartość kartoteki `Rozdzial-11`.

Prezentacja systemu

Aplikacje forum użytkowników są dostępne już od dawna – istnieją dłużej niż Internet. Początkowo istniały pod postacią BBS-ów – *Bulletin Boards Services* – miejsc, do których mogłeś przesyłać pytania, czytać inne pytania, a nawet czasami, otrzymywać odpowiedzi. BBS-y istnieją cały czas, jednak niemal w całości zostały zastąpione odpowiednikami internetowymi – grupami dyskusyjnymi. Te systemy, których przykład został przedstawiony na rysunku 11.1, są niemal identyczne. Użytkownicy, za pomocą specjalnego oprogramowania takiego jak Outlook Express, przeglądają wiadomości i tworzą własne.



Rysunek 11.1. *Grupy dyskusyjne są popularną częścią Internetu*

Grupa dyskusyjna przedstawiona na rysunku 11.1. istnieje w rzeczywistości i jest bardzo przydatna dla programistów ASP. Można ją znaleźć na serwerze `msnews.microsoft.com`, a uczestnictwo w niej nie wymaga podawania jakiegokolwiek hasła ani identyfikatora użytkownika.

System, który stworzysz za pomocą ASP, będzie musiał dysponować tymi samymi możliwościami co istniejące grupy dyskusyjne. Lista funkcji systemu pomoże Ci zaprojektować konieczne strony oraz strukturę bazy danych:

Użytkownicy muszą być w stanie tworzyć nowe wiadomości.

Użytkownicy muszą być w stanie przeglądać wiadomości utworzone przez nich samych oraz przez innych użytkowników.

Użytkownicy muszą być w stanie odpowiadać na wiadomości, a ich odpowiedzi powinny być prezentowane w taki sposób, aby było wiadomo, iż są one związane z oryginalną wiadomością.

Użytkownicy powinni być w stanie usunąć wiadomość, którą stworzyli, jednak nie mogą dysponować możliwością usuwania wiadomości stworzonych przez innych użytkowników.

Użytkownicy muszą być w jakiś sposób identyfikowani, aby ich wiadomości mogły być oznaczane.

Te funkcje niemal bezpośrednio reprezentują to, co system musi robić, jednak dodatkowo stwarzają kilka niejawnych wymagań. Mianowicie system musi przechowywać wiadomości oraz listę użytkowników i ich informacji identyfikacyjnych.

Baza danych

Pierwszym etapem będzie stworzenie bazy danych spełniającej podane wymagania. Konieczne będzie wykorzystanie jedynie dwóch tabel: jednej do przechowywania listy użytkowników i drugiej do przechowywania wiadomości. Pola tabel oraz ich typy zostały przedstawione w tabelach 11.10 oraz 11.11.

Tabela 11.10.
Tabela User

Nazwa pola	Typ pola	Długość pola
`ID`	AutoNumer	niedostępna
`FirstName`	Tekst	30
`LastName`	Tekst	50
`Email`	Tekst	50
`LastVisit`	Data/Godzina	ni dostępna
`UserID`	Tekst	20
`Password`	Tekst	20

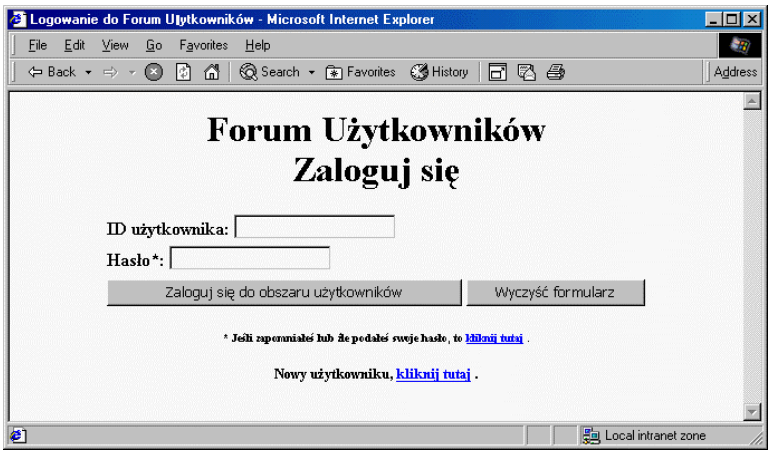
Tabela 11.11.
Tabela Message

Nazwa pola	Typ pola	Długość pola
ID	AutoNumer	niedostępna
UserID	Liczba	Liczba całkowita długa
Parent	Liczba	Liczba całkowita długa
Subject	Tekst	50
Date	Data/Godzina	niedostępna
Message	Memo	niedostępna

Aby umożliwić aplikacji ASP korzystanie z tej bazy danych, będziesz musiał umieścić ją na serwerze WWW i stworzyć dla niej nazwę źródła danych ODBC (plikową lub systemową). W przedstawionych dalej przykładach założyłem, że to zrobiłeś oraz że nadałeś nazwie źródła danych nazwę „Forum". Więcej informacji dotyczących tworzenia nazw źródeł danych znajdziesz w poprzednim rozdziale.

Identyfikacja użytkowników

Z punktu widzenia użytkowników pierwszym krokiem przy korzystaniu z forum będzie zalogowanie się do systemu. W tym właśnie miejscu powinien zaczynać się Twój kod. Kod przedstawiony na listingu 11.3 tworzy stronę logowania, którą pokazano na rysunku 11.2.



Rysunek 11.2. *Użytkownicy są identyfikowani za pomocą tej strony logowania*

Listing 11.3. *Ta strona WWW pozwala użytkownikom na zalogowanie się do systemu (Logowanie.htm)*

```
<HTML>
<HEAD>
  <TITLE>Logowanie do Forum Użytkowników</TITLE>
</HEAD>

<BODY BACKGROUND="">

<H1 ALIGN="CENTER">Forum Użytkowników<BR>Zaloguj się</H1>

<FORM ACTION="ObslugaLogowania.asp" METHOD="POST" TARGET="_parent">
  <DIV ALIGN="CENTER">
  <CENTER>
    <TABLE BORDER="0">
    <TR>
      <TD><STRONG>ID użytkownika:</STRONG>
      <INPUT TYPE="TEXT" SIZE="20" NAME="UserName"></TD>
    </TR>
    <TR>
      <TD><STRONG>Hasło*:</STRONG>
      <INPUT TYPE="PASSWORD" SIZE="20" NAME="Password"</TD>
    </TR>
```

```
22.     <TR>
23.     </TR>
24.     <TR>
25.       <TD>
26.         <INPUT TYPE="SUBMIT" NAME="Login" VALUE="Zaloguj się do
            ⇨obszaru użytkowników">
27.         <INPUT TYPE="RESET" NAME="Czyść" VALUE="Wyczyść formularz">
28.       </TD>
29.     </TR>
30.     </TABLE>
31.   </CENTER>
32.   </DIV>
33.   <DIV ALIGN="CENTER">
34.   <CENTER>
35.     <H6>* Jeśli zapomniałeś lub źle podałeś swoje hasło, to
36.     <A HREF="ZapomnianeHaslo.asp" TARGET="_parent">kliknij
        ⇨tutaj</A>
37.     .</H6>
38.   </CENTER>
39.   </DIV>
40.   <DIV ALIGN="CENTER">
41.   <CENTER>
42.     <H6><BIG>Nowy użytkowniku, <a HREF="NowyUzytkownik.asp">kliknij
        ⇨tutaj</A>
43.     .</BIG></H6>
44.   </CENTER>
45.   </DIV>
46. </FORM>
47. </BODY>
48. </HTML>
```

analiza Większość strony przedstawionej na listingu 11.3 jest bardzo prostym kodem HTML – nie jest to strona ASP. Poniżej wyjaśnione zostały lini, które mogą zainteresować Ciebie jako programistę. Linia 10., zawierająca znacznik otwierający formularz HTML, określa plik, jaki zostanie wykonany po kliknięciu na przycisku Submit w celu obsłużenie danych wpisanych w formularzu. W tym przypadku do obsługi danych służy plik `ObslugaLogowania.asp`, podawanie ścieżki nie jest konieczne, gdyż plik jest przechowywany w tej samej kartotece co formularz. W liniach 16 i 20 zostały umieszczone pola służące odpowiednio do podania nazwy użytkownika i hasła. Ważnym szczegółem, na jaki należy zwrócić uwagę w tych liniach, są nazwy nadane obu polom, gdyż posłużą one do pobrania wartości tych pól z kolekcji `Request.Form` w programie obsługującym formularz.

Kod przedstawiony na listingu 11.3 tworzy prosty formularz. Strona, która w rzeczywistości łączy się z bazą danych i sprawdza poprawność informacji o tożsamości użytkownika, jest określona jako „cel" formularza, jest to strona `ObslugaLogowania.asp`. Inne formularze HTML i odpowiadające im skrypty ASP zostały stworzone dla obsługi rejestracji nowych użytkowników oraz pomocy użytkownikom, którzy zapomnieli swojego hasła.

Miejscem, w którym są sprawdzane informacje podane przez użytkownika, jest procedura obsługi strony logowania przedstawiona na listingu 11.4. Bazując na poprawności podanego identyfikatora użytkownika oraz hasła, ta strona musi zdecydować, jak obsługiwać poprawne oraz niepoprawne próby logowania.

Listing 11.4. *Weryfikacja formularza logowania z listingu 11.3*

```
<HTML>
<HEAD>
       <TITLE>Logowanie użytkownika do systemu Forum
       ⇨Użytkowników</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<%
Dim Conn
Dim RSUser
Dim SQL

Set Conn = Server.CreateObject("ADODB.Connection")
Set RSUser = Server.CreateObject("ADODB.Recordset")

Conn.Open "Forum"

SQL = "SELECT * FROM User WHERE UserID='" & _
       Request.Form("UserName") & "'"
RSuser.Open SQL,Conn,1,2

If RSUser.EOF Then
%>
<H2 ALIGN="CENTER">Użytkownik nie istnieje</H2>
<H3 ALIGN="CENTER"><%= Request.Form("UserName") %></H3>
<H3 ALIGN="CENTER"><A HREF="Logowanie.htm" TARGET="_parent">Kliknij
⇨tutaj, aby zalogować się ponownie</A></H3>
<P> </P>
<H3 ALIGN="CENTER"> LUB </H3>
<P> </P>
<H3 ALIGN="CENTER"><A HREF="NowyUzytkownik.asp"
 ⇨TARGET="_parent">Kliknij tutaj, aby dołączyć do Forum</A></H3>
<%
Else
  If RSUser("Password") <> Request.Form("Password") Then
%>
<H2 ALIGN="CENTER">Podano nieprawidłowe hasło dla</H2>
<H3 ALIGN="CENTER"><%= Request.Form("UserName") %></H3>
<H3 ALIGN="CENTER"><A HREF="Logowanie.htm" TARGET="_parent">Kliknij
⇨tutaj, aby zalogować się ponownie</A></H3>
<P> </P>
<H3 ALIGN="CENTER"> LUB </H3>
<P> </P>
<H3 ALIGN="CENTER"><A HREF="ZapomnianeHaslo.asp">Kliknij tutaj, aby
przesłać Ci hasło za pomocą poczty elektronicznej.</A></H3>
<%
  Else
    Session("User") = "Yes"
    Session("User_ID") = RSUser("ID")
    Session("UserName") = RSUser("FirstName") & " " &
    ⇨RSUser("LastName")
%>
<H2 ALIGN="CENTER">Logowanie pomyślne</H2>
<H3 ALIGN="CENTER">Witamy ponownie <%= RSUser("FirstName") %></H3>
<H3 ALIGN="CENTER">Ostatnio odwiedziłeś system <%=
⇨RSUser("LastVisit") %></H3>
<H3 ALIGN="CENTER">
  <A HREF="Forum.asp" TARGET="_parent">Kliknij tutaj, aby wejść na
  ⇨ Forum Użytkowników</A>
```

```
52. </H3>
53. <%
54.     RSUser("LastVisit") = Now
55.     RSUser.Update
56.   End If
57. End If
58.
59. %>
60. </BODY>
61. </HTML>
```

Przedstawiony powyżej skrypt próbuje pobrać rekord bazy danych dla podanego identyfikatora użytkownika (linie 7. do 18.). Jeśli taki rekord nie istnieje, oznacza to, że nie podano poprawnego identyfikatora i skrypt wyświetla odpowiedni komunikat (linie 22. – 28.). Jeśli rekord istnieje, to kolejny fragment kodu (linie 31. – 40.) sprawdza czy podano poprawne hasło. Jeśli nie, to wyświetlany jest kolejny komunikat. W przypadku, gdy podano poprawne hasło, jest wykonywany kod (linie 43. – 45.), który zapisuje wybrane wartości w obiekcie `Session`, dzięki czemu będzie można z nich skorzystać w przyszłości. Następnie jest wyświetlany komunikat witający użytkownika (linie 43. – 47. – 51.). Na końcu jest uaktualniana wartość pola `LastVisit`, w której jest zapisywana data aktualnej wizyty. Jest to ostatnia czynność, po której wykonywanie strony zostanie zakończone.

Powyższy kod przedstawia kilka funkcji ADO, każda z nich została bardziej szczegółowo opisana w następnych sekcjach rozdziału.

Otwieranie połączenia

Linie 11. i 14. listingu 11.4 tworzą i otwierają połączenie z bazą danych. Nowy, początkowo zamknięty obiekt Connection jest tworzony za pomocą metody `Server.CreateObject`. Identyfikator klasy (Class ID) ADODB.Connection jest konieczny, gdyż informuje on metodę o tym, jaki obiekt należy stworzyć. Następnie, w linii 14., jest wywoływana metoda `Open` nowego połączenia; jako argument jej wywołania jest podawana nazwa źródła danych wykorzystywanej bazy danych. Jej pierwszy parametr jest odpowiednikiem właściwości `ConnectionString` obiektu `Connection` i może zostać określony przed wywołaniem metody `Open`, jak pokazano na poniższym przykładzie:

```
Set Conn = Server.CreateObject("ADODB.Connection")
Conn.ConnectionString = "Forum"
Conn.Open
```

Powyższy kod jest całkowicie równoważny kodowi przedstawionemu na listingu 11.4 – wykonuje te same czynności, choć w nieco inny sposób. W tym przypadku, właściwość `ConnectionString` zawiera po prostu nazwę źródła danych, jednak może ona zawierać także inne informacje takie jak, identyfikator użytkownika i hasło dostępu do bazy danych. Jeśli, dla przykładu, dostęp do bazy danych wymagałby użycia identyfikatora użytkownika o wartości `"sa"` oraz pustego hasła, to mógłbyś użyć właściwość `ConnectionString` o wartości:

```
"DSN=Forum;userid=sa;pwd=;"
```

Pamiętaj, że metoda Open pobiera także dwa opcjonalne parametry (oprócz `ConnectionString`) – identyfikator użytkownika oraz hasło. Jeśli chcesz, to możesz podać te wartości jako oddzielne argumenty:

```
Conn.Open "Forum", "sa", ""
```

Niezależnie od metody jakiej użyjesz, przed wykorzystaniem obiektu `Connection`, będziesz go musiał stworzyć i otworzyć.

Tworzenie i otwieranie zbioru rekordów

Linie 12. i 18. listingu 11.4 tworzą i otwierają zbiór rekordów (obiekt `Recordset`). W wywołaniu metody `Open` została użyta grupa parametrów. Podobnie jak w przypadku omówionej wcześniej metody `Open` obiektu `Connection`, także i teraz każdy z argumentów może zostać określony przy wykorzystaniu właściwości, której wartość jest podawana przed wywołaniem metody `Open`. Kod przedstawiony poniżej, z funkcjonalnego punktu widzenia, odpowiada kodowi z listingu 11.4.

```
RSUser.Source = SQL
Set RSUser.ActiveConnection = Conn
RSUser.CursorType = 1   'Keyset
RSUser.LockType = 2     'Pesymistyczny
RSUser.Open
```

Musisz pamiętać o podaniu tych wartości, niezależnie od metody jaką do tego wykorzystasz, gdyż w przeciwnym wypadku zostaną zastosowane wartości domyślne. Jeśli nie dysponujesz stworzonym i otworzonym obiektem `Connection`, jednak pomimo tego chcesz otworzyć zbiór rekordów, to możesz zastąpić argument `Conn` faktycznym łańcuchem znaków używanym do nawiązania połączenia z bazą danych:

```
RSUser.Open SQL, "Forum", 1,2
```

Wydaje się, że taka metoda daje dokładnie identyczne rezultaty jak poprzednia, tak jednak nie jest. Za każdym razem, gdy zbiór rekordów jest tworzony przy wykorzystaniu tej metody określania połączenia, jest tworzony nowy obiekt `Connection`. Przedstawiony poniżej kod powoduje stworzenie trzech obiektów `Connection`:

```
Set RSPrzykalad1 = Server.CreateObject("ADODB.Recordset")
Set RSPrzykalad2 = Server.CreateObject("ADODB.Recordset")
Set RSPrzykalad3 = Server.CreateObject("ADODB.Recordset")
RSPrzyklad1.Open SQL, "Forum", 1, 2
RSPrzyklad2.Open SQL, "Forum", 1, 2
RSPrzyklad3.Open SQL, "Forum", 1, 2
```

Jednak kod przedstawiony poniżej wykorzystuje tylko jeden obiekt `Connection` (choć jego wykonanie daje identyczne rezultaty):

```
Set RSPrzykalad1 = Server.CreateObject("ADODB.Recordset")
Set RSPrzykalad2 = Server.CreateObject("ADODB.Recordset")
Set RSPrzykalad3 = Server.CreateObject("ADODB.Recordset")
Set Conn = Server.CreateObject("ADODB.Connection")
Conn.Open "Forum"
RSPrzyklad1.Open SQL, Conn, 1, 2
RSPrzyklad2.Open SQL, Conn, 1, 2
RSPrzyklad3.Open SQL, Conn, 1, 2
```

Wykorzystanie tylko jednego obiektu Connection ogranicza obciążenie serwera bazy danych do jednego połączenia na każdy wykonywany współbieżnie skrypt. Pozwala to na otrzymanie maksymalnej liczby równoczesnych połączeń, gdyż jedno połączenie jest minimalną liczbą połączeń jaką można stworzyć na jednej stronie. W następnej sekcji zostanie przedstawiona metod określania czy zbiór rekordów zawiera jakiekolwiek dane, niezależnie od metody jaka została wykorzystana do jego stworzenia.

Sprawdzanie czy zbiór rekordów jest pusty

Zanim zrobisz cokolwiek ze zbiorem rekordów otworzonym w linii 18. listingu 11.4, należy sprawdzić czy nie jest on pusty. Do tego właśnie służy kod umieszczony w linii 20. Jeśli zbiór rekordów nie jest pusty, to po jego otworzeniu wskaźnik rekordu będzie umieszczony na pierwszym rekordzie, a właściwość `EOF` będzie miała wartość `false`. Właściwość ta, niezwłocznie po otworzeniu zbioru rekordów, będzie miała wartość `true` tylko i wyłącznie wtedy, gdy zbiór rekordów będzie pusty. Jeśli spróbujesz wywołać jakąkolwiek metodę obiektu `Recordset`, jeśli wskaźnik rekordów nie znajduje się na poprawnym rekordzie (na przykład, gdy właściwość `BOF` lub `EOF` ma wartość `true`), to zostanie wygenerowany błąd.

> Sprawdzenie wyłącznie właściwości `EOF` będzie wystarczające tylko wtedy, gdy zostanie wykonane tuż po otworzeniu zbioru rekordów. Jeśli istnieje możliwość, że przed wykonaniem testu były realizowane jakieś operacje na zbiorze rekordów, to powinieneś sprawdzać zarówno właściwość `EOF` jak i `BOF`.

Pobieranie wartości ze zbioru rekordów

Jeśli zbiór rekordów nie jest pusty, to program będzie mógł skorzystać z pól zbioru. Poniżej została podana składnia używana do pobierania wartości pól zbioru rekordów:

```
<NazwaZbioruRekordów>("<NazwaPola>")
```

To polecenie może zostać wykorzystane do pobrania lub określenia wartości dowolnego pola. W rzeczywistości jest ono odpowiednikiem poniższego polecenia:

```
RSUser.Fields.Item("UserID").Value
```

Dzieje się tak, gdyż kolekcja `Fields` jest domyślną właściwością obiektu `Recordset`, właściwość `Item` – domyślną właściwością kolekcji `Fields`, a właściwość `Value` – domyślną właściwością obiektu `Field`. Właśnie dlatego oba polecenia są sobie równoważne. To ważna informacja, którą warto zapamiętać – po co bowiem używać bardziej skomplikowanej składni.

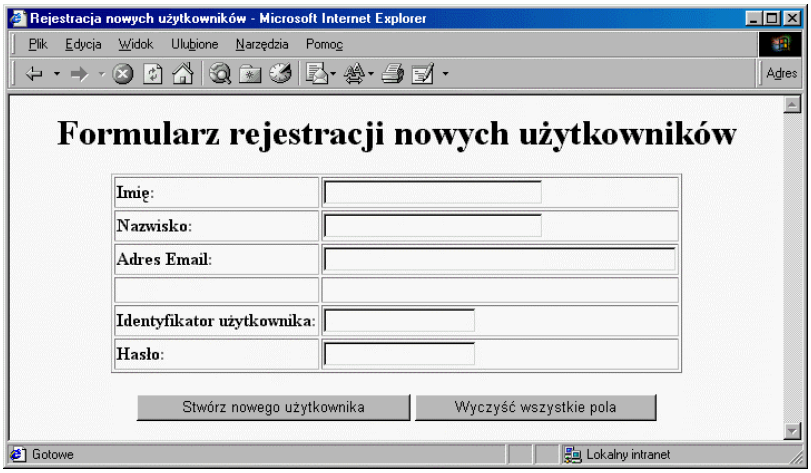
Modyfikowanie wartości w zbiorze rekordów

Jeśli nie otworzyłeś zbioru rekordów jako przeznaczonego tylko do odczytu (patrz omówione wcześniej właściwości `CursorType` oraz `LockType`), to będziesz mógł modyfikować wartości pól, używając do tego dwuetapowego procesu. W pierwszej kolejności,

przy użyciu normalnego operatora przypisania, umieścisz wartości w wybranych polach (patrz linia 54. listingu 11.4), a następnie wywołasz metodę `Upadate`. Jeśli zmodyfikujesz wartości pól, a następnie zamkniesz zbiór rekordów lub przejdziesz do innego rekordu, to wprowadzone modyfikacje zostaną utracone. Zbiory rekordów używane w niektórych innych modelach danych, takich jak DAO, wymagają, aby przed modyfikacją wartości pól została wywołana metoda `Edit`. Metoda ta nie istnieje jednak w ADO, a zatem jej użycie spowoduje wygenerowanie błędu („Object does not support this method").

Tworzenie nowego użytkownika

W poprzedniej sekcji została przedstawiona strona służąca do podawania identyfikatora użytkownika oraz hasła, jednak skąd biorą się te wartości? W systemie musi znajdować się strona pozwalająca na rejestrację nowych użytkowników, co pozwoli im na korzystanie z forum. Ta strona została już połączona ze stroną służącą do logowania użytkowników, jej kod został przedstawiony na listingu 11.5. Rysunek 11.3 przedstawia formularz rejestracyjny nowych użytkowników, tworzony przez kod z listingu 11.5.



Rysunek 11.3. *Formularz rejestracji nowych użytkowników pozwala użytkownikom na podanie informacji, które zostaną zapisane w bazie danych*

Listing 11.5. *Ten formularz HTML prosi użytkownika o podanie informacji koniecznych do stworzenia nowego rekordu w tabeli User (NowyUzytkownik.asp)*

```
 1. <HTML>
 2. <HEAD>
 3.
 4.         <TITLE>Rejestracja nowych użytkowników</TITLE>
 5. </HEAD>
 6.
 7. <BODY BGCOLOR="#FFFFFF" TEXT="#000000">
 8.
 9. <H1 ALIGN="CENTER">Formularz rejestracji nowych użytkowników</H1>
10.
11. <FORM METHOD="POST" ACTION="StworzUzytkownika.asp">
12.   <DIV ALIGN="CENTER"><CENTER><TABLE BORDER="1">
13.     <TR>
```

```
14.       <TD><STRONG>Imię:</STRONG></TD>
15.       <TD><INPUT TYPE="TEXT" NAME="txtFirstName" SIZE="30"></TD>
16.     </TR>
17.     <TR>
18.       <TD><STRONG>Nazwisko:</STRONG></TD>
19.       <TD><INPUT TYPE="TEXT" NAME="txtLastName" SIZE="30"></TD>
20.     </TR>
21.     <TR>
22.       <TD><STRONG>Adres Email:</STRONG></TD>
23.       <TD><INPUT TYPE="TEXT" NAME="txtEmailAddress" SIZE="50"></TD>
24.     </TR>
25.     <TR>
26.       <TD><STRONG> </STRONG></TD>
27.       <TD> </TD>
28.     </TR>
29.     <TR>
30.       <TD><STRONG>Identyfikator użytkownika:</STRONG></TD>
31.       <TD><INPUT TYPE="TEXT" NAME="txtUserID" SIZE="20"></TD>
32.     </TR>
33.     <TR>
34.       <TD><STRONG>Hasło:</STRONG></TD>
35.       <TD><INPUT TYPE="PASSWORD" NAME="txtPassword" SIZE="20"></TD>
36.     </TR>
37.   </TABLE>
38.   </CENTER></DIV>
39.   <DIV ALIGN="CENTER">
40.   <CENTER>
41.   <P><INPUT TYPE="SUBMIT" VALUE="Stwórz nowego użytkownika"
      ⇨NAME="B1">
42.   <INPUT TYPE="RESET" VALUE="Wyczyść wszystkie pola" NAME="B2"></P>
43.   </CENTER>
44.   </DIV>
45. </FORM>
46. </BODY>
47. </HTML>
```

Listing 11.5 przedstawia kod formularza HTML. Sam formularz zawiera kod, którego w większości nie musisz pisać sam. Przedstawiony formularz można stworzyć w dowolnym edytorze HTML (jednym z lepszych, choć nie jedynym, może być FrontPage). W rzeczywistości sam układ formularza nie jest ważny; dla programisty znaczenie posiada jedynie fakt istnienia pól i przycisków formularza oraz ich nazwy. Nazwy pól oraz przycisków formularza zdefiniowane zostały w liniach 15., 19., 23., 31. i 35.; są one wykorzystywane w skrypcie, który obsługuje wyniki wykonania tej strony. Nazwa skryptu obsługującego formularz została podana w linii 11., a jego kod można znaleźć na listingu 11.6.

Listing 11.6. *Ten skrypt przetwarza wartości wprowadzone do formularza na stronie NowyUzytkownik.asp i tworzy nowego użytkownika w tabeli User (StworzUzytkownika.asp)*

```
1. <HTML>
2. <HEAD>
3.   <TITLE>Stwórz nowego użytkownika</TITLE>
4. </HEAD>
5.
6. <BODY BGCOLOR="#FFFFFF">
7. <%
8. Dim Conn
```

```
 9. Dim RSUser
10. Dim SQL
11.
12. Set Conn = Server.CreateObject("ADODB.Connection")
13. Set RSUser = Server.CreateObject("ADODB.Recordset")
14.
15. Conn.Open "Forum"
16.
17. SQL = "SELECT * FROM User WHERE UserID='" &
    ⇨Request.Form("txtUserid") & "'"
18.
19. RSUser.Open SQL,Conn,1,2
20.
21. If Not RSUser.EOF Then
22. %>
23.
24. <H2 ALIGN="CENTER">Użytkownik już istnieje.</H2>
25.
26. <H3 ALIGN="CENTER"> <%= Request.Form("txtUserId") %></H3>
27.
28. <H3 ALIGN="CENTER"><A HREF="NowyUzytkownik.asp" TARGET="_parent">
29. Kliknij tutaj, aby podać inny identyfikator użytkownika</A></H3>
30.
31. <P> </P>
32.
33. <H3 ALIGN="CENTER">LUB</H3>
34.
35. <P> </P>
36.
37. <H3 ALIGN="CENTER"><A HREF="ZapomnianeHaslo.asp" TARGET="_parent">
38. Kliknij tutaj, jeśli chcesz, aby przesłać Ci hasło pocztą
39. elektroniczną,jeśli zapomniałeś go, a wcześniej zarejestrowałeś się
    ⇨przy użyciu tego identyfikatora.</A></H3>
40. <%
41. Else
42.
43.   RSUser.AddNew
44.   RSUser("FirstName") = Request.Form("txtFirstName")
45.   RSUser("LastName") = Request.Form("txtLastName")
46.   RSUser("UserID") = Request.Form("txtUserID")
47.   RSUser("Password") = Request.Form("txtPassword")
48.   RSUser("Email") = Request.Form("txtEmail")
49.   RSUser("LastVisit") = Now()
50.   RSUser.Update
51. %>
52.
53. <H2 ALIGN="CENTER">Stworzono nowego użytkownika</H2>
54.
55. <H3 ALIGN="CENTER">
56. Witamy na Forum Użytkowników <%= Request.Form("txtFirstName") %>
57. </H3>
58.
59. <H3 ALIGN="CENTER">
60. <A HREF="Logowanie.htm" TARGET="_parent">Kliknij tutaj, aby się
    ⇨zalogować</A></H3>
61. <%
62. End If
63. %>
64. </BODY>
65. </HTML>
```

Ten skrypt przeprowadza tylko jedną kontrolę poprawności danych (umieszczone w liniach 17. – 21.), sprawdzając, czy podany identyfikator użytkownika już istnieje. Zupełnie puste wartości zostałyby zaakceptowane, jednak jest to do przyjęcia w przypadku testowego systemu ADO. Zadaniem, jakie miał ilustrować ten konkretny skrypt, jest dodawanie nowego rekordu do zbioru rekordów; zadanie to zostało szczegółowiej omówione w kolejnych sekcjach.

Dodawanie rekordów do zbioru rekordów

Po sprawdzeniu, że wprowadzony identyfikator użytkownika nie występuje już w bazie danych (linia 41. listingu 11.6) nadszedł czas, aby dodać użytkownika do tabeli. Zbiór rekordów musi zostać wcześniej otworzony na tabeli, do której chcesz dodawać rekordy. W powyższym skrypcie zbiór rekordów został otworzony na tabeli `User` przy użyciu kryteriów, które nie zostały spełnione. Do otwarcia zbioru rekordów posłużyło polecenie SQL umieszczone w linii 17. Nawet pomimo tego, że nie zostały zwrócone żadne rekordy, zbiór rekordów cały czas wskazuje na tabelę User, a zatem dodawane rekordy będą umieszczane właśnie w niej. Jeśli będziesz potrzebował otworzyć zbiór rekordów wyłącznie w celu dodawania nowych rekordów, to w poleceniu SQL powinieneś podać takie kryteria, które spowodują, że nie zostaną zwrócone żadne rekordy. Takim poleceniem może być:

```
SELECT * FROM User WHERE 1=2
```

Nie ma żadnego sensu w zużywaniu pamięci i zasobów procesora na pełny zbiór rekordów, jeśli nie jest on do niczego potrzebny.

Kiedy będziesz już dysponował zbiorem rekordów, to dodanie faktycznego rekordu jest łatwym, trójetapowym procesem:

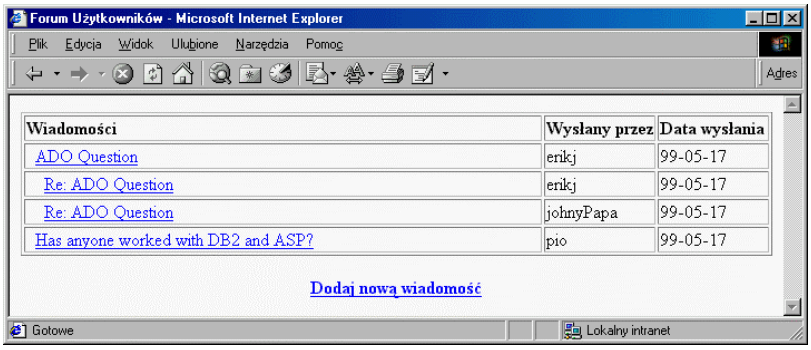
1. Wywołaj metodę `AddNew` (linia 43.).
2. Przypisz polom odpowiednie wartości (linie 44. – 49.).
3. Wywołaj metodę `Update` (linia 50.).

Zauważ, że nie musisz określać wartości pola ID (identyfikatora), gdyż jest ona określana automatycznie, poprzez inkrementację wartości tego samego pola poprzedniego rekordu (jest to pole auto-inkrementujące). Wszelka weryfikacja na poziomie rekordu, w tym weryfikacja indeksów, jest wykonywana w momencie wywołania metody `Update`, a zatem zazwyczaj właśnie w tym miejscu pojawiają się błędy. Błędy powodowane przez niezgodność typów lub wielkości pól są generowane w liniach, w których znajduje się przypisanie wartości do pola.

Wyświetlanie Forum

Główną stroną tworzonego systemu jest `Forum.asp` przedstawiona na listingu 11.7. Służy ona do wyświetlania wiadomości pogrupowanych według prowadzonych rozmów. Na tej stronie, użytkownik może kliknąć na dowolnej wiadomości, co spowoduje wyświetlenie jej tekstu lub dodać nową wiadomość. Na rysunku 11.4 została przedsta-

wiona kompletna strona wyświetlona w przeglądarce. Funkcja odpowiadania na wiadomości jest dostępna tylko na stronie prezentującej treść wiadomości.



Rysunek 11.4. *Wyniki wykonania Forum.asp przedstawiają listę wszystkich wiadomości zapisanych w bazie danych*

Listing 11.7. *Ta strona wyświetla wszystkie wiadomości przechowywane w tabeli Messages (Forum.asp)*

```
<HTML>
<SCRIPT LANGUAGE="VBSCRIPT" RUNAT="SERVER">
Sub WyswietlWiadomosciPotomne( Parent, Level )

  Dim RSMessages
  Dim SQL
  Dim i

  Set RSMessages = Server.CreateObject("ADODB.Recordset")

  SQL = "SELECT Message.ID, " & _
  "Message.Subject, Message.Date, " & _
  "User.UserID FROM Message, User " & _
  "WHERE Message.UserID = User.ID AND " & _
  "Message.Parent=" & Parent

  RSMessages.Open SQL,Conn,1,2

  Do While Not RSMessages.EOF

    Response.Write "<TR>"
    Response.Write "<TD>"

    For i = 1 to level
      Respnse.Write "  "
    Next

    Response.Write "<A HREF='Wiadomosc.asp?id="
    Response.Write RSMessages("ID") & "'>"
    Response.Write RSMessages("Subject")
    Response.Write "</A>"
    Response.Write "</TD><TD>"
    Response.Write RSMessages("UserID")
    Response.Write "</TD><TD>"
    Response.Write FormatDateTime( RSMessages("Date"),2)
    Response.Write "</TD></TR>"

    WyswietlWiadomosciPotomne RSMessages("ID"), Level+1
    RSMessages.MoveNext

```

```
41. Loop
42.
43. End Sub
44. </SCRIPT>
45.
46. <HEAD>
47.         <TITLE>Forum Użytkowników</TITLE>
48. </HEAD>
49. <BODY>
50.
51. <TABLE BORDER="1" WIDTH="100%">
52.   <TR>
53.     <TD WIDTH="70%"><STRONG>Wiadomości</STRONG></TD>
54.     <TD WIDTH="15%"><STRONG>Wysłany przez</STRONG></TD>
55.     <TD WIDTH="15%"><STRONG>Data wysłania</STRONG></TD>
56.   </TR>
57.
58. <%
59.
60. Dim Conn
61. Set Conn = Server.CreateObject("ADODB.Connection")
62.
63. Conn.Open "Forum"
64.
65. WyswietlWiadomosciPotomne 0,1
66.
67. %>
68. </TABLE>
69.
70. <H4 ALIGN="CENTER"><A HREF="NowaWiadomosc.asp">Dodaj nową
    ⇨wiadomość</A></H4>
71. </BODY>
72. </HTML>
```

Strona przedstawiona na listingu 11.7 działa dzięki wykorzystaniu *procedury rekurencyjnej*, co oznacza, że procedura ta wywołuje sama siebie. Procedura ta nosi nazwę `WyswietlWiadomosciPotomne` i została zapisana w liniach 3. – 43. Do stworzenia całej strony konieczne jest podanie tylko dwóch parametrów. Pierwszy z nich określa dla jakiej wiadomości głównej powinny zostać wyświetlone wiadomości potomne; wartość 0 tego parametru oznacza, że powinien zostać wyświetlony najwyższy poziom wiadomości, czyli te, które nie mają wiadomości nadrzędnych. Ten argument jest wykorzystywany jako część polecenia SQL używanego do pobierania wiadomości z bazy (parz linie 11. – 15.). Dla każdej wiadomości odnalezionej „poniżej" danej wiadomości nadrzędnej, funkcja `WyswietlWiadomosciPotomne` jest wywoływana ponownie, jednak tym razem, w jej wywołaniu, jako wiadomość nadrzędna jest podawany identyfikator aktualnej wiadomości (linia 38.). W ten sposób powstaje bardzo niewielki fragment kodu, który jest w stanie obsłużyć dowolną ilość wiadomości. Drugi argument funkcji `WyswietlWiadomosciPotomne` służy do kontrolowania wcinania wiadomości i określa na jakim poziomie ogólnego drzewa aktualna wiadomość jest wyświetlana.

Przechodzenie pomiędzy rekordami

Procedura podana na listingu 11.7 przedstawia sposób wykorzystania metod obiektu Recordset służących do poruszania się pomiędzy rekordami. Metody te pozwalają na

pobranie po kolei każdego rekordu znajdującego się w zbiorze rekordów i wykonanie na nim serii operacji. W liniach 19. – 41. jest umieszczona pętla, która będzie wykonywana aż do osiągnięcia końca zbioru rekordów (`EOF`). W każdej iteracji pętli wywoływana, jest metoda `MoveNext` obiektu `Recordset`, co powoduje przesunięcie wskaźnika do następnego rekordu. Jeśli to polecenie zostałoby zapomniane, powstałaby nieskończona pętla, gdyż nigdy nie zostałby osiągnięty koniec zbioru rekordów.

Tworzenie URL-i na podstawie informacji z bazy danych

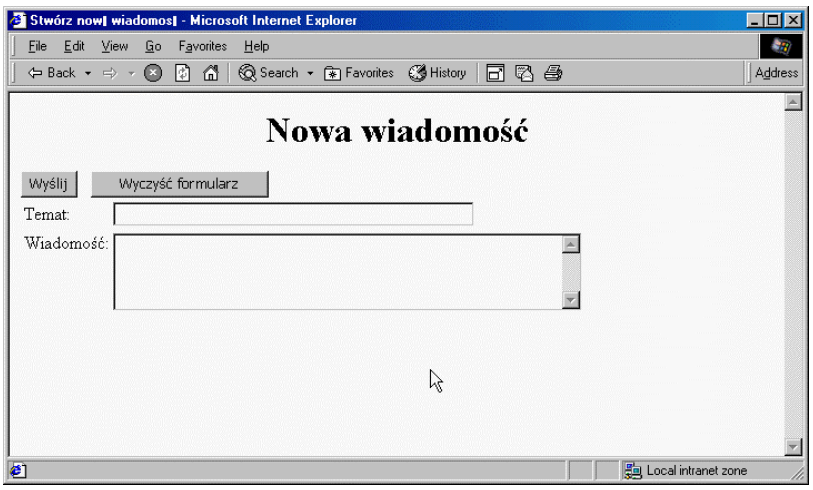
Funkcjonalna możliwość kliknięcia na konkretnej wiadomości jest implementowana przez stworzenie dla każdej wiadomości URL-a przekazującego jej identyfikator do pliku `Wiadomosc.asp`. Zadanie to jest wykonywane dzięki użyciu *podstawienia*, wstawiającemu wartość ze zbioru rekordów do kodu HTML każdego połączenia. Zaleca się zachowywanie jak najprostszej postaci URL-li i przekazywanie do nich jedynie identyfikatora lub słowa kluczowego. Przedstawiona powyżej strona w całości bazuje na wiadomościach zapisanych w bazie danych systemu, jednak musi istnieć jakaś metoda dodawania wiadomości do tej bazy. Metoda ta jest dostarczana przez stronę `NowaWiadomosc.asp` opisaną w następnej sekcji; kod tej strony przedstawiony został na listingu 11.8.

Listing 11.8. *Strona służąca do tworzenia nowych wiadomości (NowaWiadomosc.asp)*

```
 1. <HTML>
 2. <HEAD>
 3.         <TITLE>Stwórz nową wiadomość</TITLE>
 4. </HEAD>
 5. <BODY BGCOLOR="#FFFFFF" TEXT="#000000">
 6.
 7. <% If Request.QueryString("Type") = "Reply" Then %>
 8. <H1 ALIGN="CENTER">Odpowiedz na wiadomość</H1>
 9.
10. <FORM ACTION="DodajNowaWiadomosc.asp?ID=<%=
    ⇨Request.QueryString('ID')%>"
11.  METHOD="POST" TARGET="_parent">
12. <% Else %>
13.   <DIV ALIGN="CENTER"><CENTER><H1>Nowa wiadomość</H1>
14.   </CENTER></DIV>
15.
16.
17. <FORM ACTION="DodajNowaWiadomosc.asp" METHOD="POST"
    ⇨TARGET="_parent">
18. <% End If %>
19.   <TABLE BORDER="0">
20.     <TR>
21.       <TD>Temat:</TD>
22.       <TD><INPUT ID="txtSubject" NAME="txtSubject" SIZE="50"
23.        VALUE="<%= Request.QueryString("Subject") %>" ></TD>
24.     </TR>
25.     <TR>
26.       <TD VALIGN="TOP">Wiadomość:</TD>
27.       <TD><TEXTAREA ID="txtMessage" NAME="txtMessage" ROWS="4"
          ⇨COLS="50">
28.
29. </TEXTAREA></TD>
30.     </TR>
```

```
31.      <P><INPUT NAME="cmdSubmit" TYPE="SUBMIT"
         ⇨VALUE="Wyślij">  
32.      <INPUT NAME="cmdReset" TYPE="RESET" VALUE="Wyczyść
         ⇨formularz"></P>
33.    </TABLE>
34. </FORM>
35. </BODY>
36. </HTML>
```

Kod przedstawiony na listingu 11.8 tworzy prosty formularz HTML, który można zobaczyć na rysunku 11.5. Powyższa strona zawiera także niewielki fragment kodu ASP, który pozwala jej funkcjonować zarówno jako formularz służący do tworzenia nowych wiadomości, jak i formularz służący do odpowiadania na wiadomości przesłane do forum. Podczas odpowiadania na wiadomość, linie 7. – 18. modyfikują URL umieszczony w parametrze `action` formularza, dodając do niego identyfikator wiadomości. Ten parametr, wraz z zawartością samego formularza, przesyłany jest do programu obsługi (`DodajNowaWiadomosc.asp`), którego kod przedstawiono na listingu 11.9.



Rysunek 11.5. *Ten formularz HTML gromadzi informacje, które posłużą do stworzenia nowej wiadomości forum użytkowników*

Skrypt obsługujący otrzymuje wyniki z formularza przedstawionego na rysunku 11.5 i musi zapisać je w bazie danych. Czynności wykonywane przy dodawaniu nowych wiadomości zostały opisane w następnej sekcji rozdziału.

Przetwarzanie formularza nowej wiadomości

Strona `DodajNowaWiadomosc.asp`, której kod został przedstawiony na listingu 11.9,. dodaje nową wiadomość do systemu, zachowując się przy tym w różny sposób, w zależności od tego czy zapisywana wiadomość jest zupełnie nowa, czy też stanowi odpowiedź. Listing 11.9 zawiera kod używany do przetworzenia wiadomości – czyli do dodania nowego rekordu do bazy danych.

Listing 11.9. *Stworzenie nowej wiadomości lub odpowiedzi na podstawie wartości przekazanych parametrów (DodajNowaWiadomosc.asp)*

```
 1. <%@ LANGUAGE=VBScript %>
 2. <HTML>
 3. <%
 4. Dim Conn
 5. Dim RSAddMessage
 6. Dim SQL
 7.
 8. Set Conn = Server.CreateObject("ADODB.Connection")
 9. Set RSAddMessage = Server.CreateObject("ADODB.Recordset")
10.
11. Conn.Open "Forum"
12.
13. SQL = "SELECT * FROM Message Where 1=2"
14.
15. RSAddMessage.Open SQL, Conn, 1, 2
16.
17. RSAddMessage.AddNew
18. RSAddMessage("Subject") = Request.Form("txtSubject")
19. RSAddMessage("UserID") = Session("User_ID")
20. RSAddMessage("Date") = Now()
21. If Request.QueryString("ID") > 0 Then
22.   RSAddMessage("Parent") = Request.QueryString("ID")
23. Else
24.   RSAddMessage("Parent") = 0
25. End If
26.
27. RSAddMessage("Message") = Request.Form("txtMessages")
28. RSAddMessage.Update
29.
30. %>
31.
32. <HEAD>
33.         <TITLE>Dodaj wiadomość</TITLE>
34. </HEAD>
35.
36. <BODY BGCOLOR="#FFFFFF" TEXT="#000000">
37.
38. <H1 ALIGN="CENTER">Twoja wiadomość została dodana!</H1>
39.
40. <P ALIGN="CENTER"><A HREF="Forum.asp">
41. Kliknij tutaj, aby powrócić na Forum użytkowników</A>
42. </BODY>
43. </HTML>
```

Najważniejszy skrypt tej strony znajduje się w liniach 17. – 28. Ten skrypt tworzy nowy rekord, po kolei zapisuje wartości jego pól, a następnie, w zależności czy wiadomość jest odpowiedzią czy nie, polu wiadomości nadrzędnej jest przypisywany identyfikator wiadomości, na którą odpowiadamy lub wartość `0`. Po zapisaniu wszystkich wiadomości jest wywoływana metoda `Update` (w linii 28.). Dodana wiadomość pojawi się na liście wyświetlanej przez stronę `Forum.asp` zaraz po jej odświeżeniu.

Teraz dysponujemy już większością możliwości funkcjonalnych tworzonego forum. Użytkownik może się zalogować, dodać wiadomość i wyświetlić listę wszystkich dostępnych wiadomości; cały czas brakuje jednak kilku elementów. Podstawowym z nich

jest możliwość wyświetlenia pojedynczej wiadomości. Do listy wiadomości dodaliśmy już URL-e, które mają umożliwić wyświetlenie wiadomości (patrz linie 28. – 29. listingu 11.7), jednak, jak na razie, nie został stworzony żaden kod, który by je obsługiwał. Tek kod jest stosunkowo ważny, dlatego też zostanie omówiony w następnej sekcji.

Przeglądanie pojedynczych wiadomości

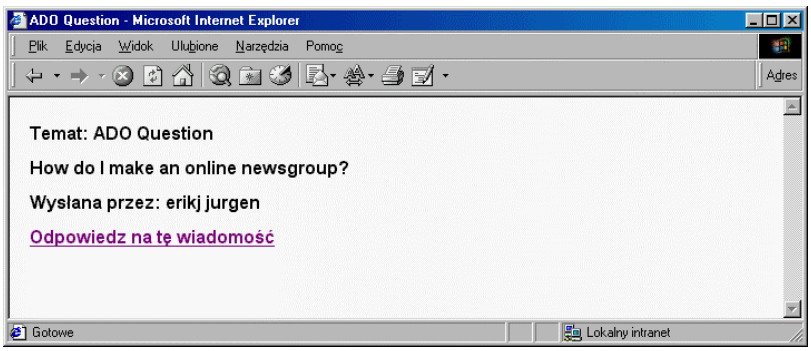
Na listingu 11.10 został przedstawiony skrypt służący do wyświetlania pojedynczych wiadomości. Czynności realizowane za pomocą ADO na tej stronie są takie same, jak na poprzednich stronach. Bez ADO cały system nie mógłby wiele zdziałać. Listing 11.10 zawiera kod strony `Wiadomosc.asp` używanej do wyświetlania każdej wybranej wiadomości.

Listing 11.10. *Ta strona wyświetla wiadomości zapisane w systemie na podstawie przekazanego do niej identyfikatora (Wiadomosc.asp)*

```
 1. <HTML>
 2. <%
 3. Dim Conn
 4. Dim RSMessage
 5. Dim TheSubject
 6. Dim TheMessage
 7. Dim SQL
 8.
 9. Set Conn = Server.CreateObject("ADODB.Connection")
10. Set RSMessage = Server.CreateObject("ADODB.Recordset")
11.
12. Conn.Open "Forum"
13.
14. SQL = "SELECT Message.ID, " & _
15. "Message.Subject, Message.Date, Message.Message, " & _
16. "User.UserID, User.FirstName, User.LastName, FROM Message, User " & _
17. "WHERE Message.UserID = User.ID AND " & _
18. "Message.ID =" & Request.QueryString("ID")
19.
20. RSMessage.Open SQL, Conn, 1, 2
21.
22. If Not RSMessage.EOF Then
23.
24.   TheSubject = Server.HTMLEncode(RSMessage("Subject"))
25.   TheMessage = Server.HTMLEncode(RSMessage("Message"))
26.   ThePoster = RSMessage("FirstName") & " " & RSMessage("LastName")
27.   QueryString = "Type=Reply&ID=" & RSMessage("ID") & "&Subject=" & _
28.   Server.URLEncode(RSMessage("Subject"))
29.
30. End If
31. %>
32. <HEAD>
33.        <TITLE><%= TheSubject %></TITLE>
34. </HEAD>
35.
36. <BODY>
37.
38. <TABLE BORDER="0" WIDTH="100%" CELLSPACING="4" CELLPADDING="4">
39.   <TR>
40.     <TD WIDTH="100%"><STRONG><FONT FACE="ARIAL">
```

```
41.     Temat: <%= TheSubject %></FONT></STRONG></TD>
42.   </TR>
43.   <TR>
44.     <TD WIDTH="100%"><STRONG><FONT FACE="ARIAL">
45.     <%= TheMessage %></FONT></STRONG></TD>
46.   </TR>
47.   <TR>
48.     <TD WIDTH="100%"><STRONG><FONT FACE="ARIAL">
49.     Wysłana przez: <%= ThePoster %></FONT></STRONG></TD>
50.   </TR>
51.   <TR>
52.     <TD WIDTH="100%"><STRONG><FONT FACE="ARIAL">
53.     <A HREF="NowaWiadomosc.asp?<%= QueryString %>"Odpowiedz na tę
        ⇨wiadomość></A>
54.     </FONT></STRONG></TD>
55.   </TR>
56. </TABLE>
57.
58. <P> </P>
59. </BODY>
60. </HTML>
```

Kod wyświetlający wiadomość jest prosty; prezentuje on wiadomość w sposób czytelny (patrz rysunek 11.6) i zapewnia, że żadne znaki specjalne nie zaburzą jej wyglądu (linie 24. – 26.). Zmienne pośrednie są używane wyłącznie do uproszczenia kodu strony i nie są konieczne. W linii 53., na samym końcu tabeli służącej do przedstawienia wiadomości, jest umieszczany URL do strony `NowaWiadomosc.asp`, omówionej we wcześniejszej części rozdziału. W tym wypadku, ze względu na to, że chcesz odpowiedzieć na oglądaną wiadomość, do adresu URL jest dodawana grupa parametrów. Przesyłany jest identyfikator wiadomości, na którą odpowiadamy, jej temat oraz parametr `Type` używany przez skrypt `NowaWiadomosc.asp` (listing 11.8) do określenia sposobu zapisania nowej wiadomości.



Rysunek 11.6. *Skrypt Wiadomosc.asp wyświetla wybraną wiadomość i pozwala użytkownikowi na odpowiedzenie na nią*

Podsumowanie systemu Forum Użytkowników

Stworzyłeś już znaczną większość aplikacji forum użytkowników. System jest już prawie kompletny, nie licząc kilku mniej znaczących szczegółów, które zostaną przedstawione jako ćwiczenia pod koniec tego rozdziału.

Praca ze schematami baz danych

W tej części rozdziału zajmiemy się przykładem bardziej zaawansowanych możliwości ADO – metodą `Connection.OpenSchema`.

Korzystając z ADO lub jakiegokolwiek innego narzędzia baz danych, przed wykonaniem kodu będziesz musiał być świadomy struktury bazy. Zapisanie używanych poleceń SQL oraz innych rozkazów wymaga znajomości nazw tabel oraz innych informacji; co jednak zrobić, jeśli będziesz chciał pracować na bazie, której struktura jest dynamiczna – której tabele i pola są co jakiś czas dodawane, usuwane bądź zmieniane? ADO posiada kilka użytecznych możliwości, pozwalających na określenie struktury bazy danych w trakcie wykonywania programu. Możliwość pobrania zbioru rekordów zawierającego wszystkie tabele bazy danych pozwala na stworzenie, dla przykładu, doraźnego systemu wyszukiwawczego, bez konieczności kodowania czegokolwiek na stałe. Strona przedstawiona na listingu 11.11 tworzy serię tabel HTML, po jednej na każdą tabelę bazy danych, dostępnej za pomocą nazwy źródła danych ODBC. Nazwa źródła danych została zakodowana na stałe w linii 22., jednak równie dobrze można by ją pobierać z kolekcji `Request.Form` lub `Request.QueryString`.

Listing 11.11. *Przykład.asp tworzy listę wszystkich tabel bazy danych oraz listę pól każdej z tych tabel*

```
 1. <HTML>
 2.
 3. <HEAD>
 4.   <TITLE>Przykład otwierania schematu</TITLE>
 5. </HEAD>
 6.
 7. <BODY>
 8. <%
 9. Dim Conn
10. Dim RSTables
11. Dim RSCurrentTable
12. Dim adSchemaTables
13. Dim Field
14. Dim SQL
15.
16. adSchemaTables = 20
17.
18. Set Conn = Server.CreateObject("ADODB.Connection")
19. Set RSTables = Server.CreateObject("ADODB.Recordset")
20. Set RSCurrentTable = Server.CreateObject("ADODB.Recordset")
21.
22. Conn.Open "Forum"
23.
24. Set RSTables = Conn.OpenSchema( adSchemaTables )
25.
26. RSTables.Filter = "Table_Type='Table'"
27.
28. Do While Not RSTables.EOF
29. %>
30.
31. <H2><%= RSTables("Table_Name" %></H2>
32. <%
33.   SQL = "SELECT * FROM [" & RSTables("Table_Name") & "] WHERE 1=2"
```

```
34.
35.   RSCurrentTable.Open SQL, Conn, 1, 2
36.
37.   For Each Field in RSCurrentTable.Fields
38.
39.     Response.Write "<H4>   "
40.     Response.Write Field.Name & "</H4>" & vbcrlf
41.
42.   Next
43.
44.   RSCurrentTable.Close
45.
46.   RSTables.MoveNext
47. Loop
48.
49. RSTables.Close
50. Conn.Close
51. %>
52. </BODY>
53. </HTML>
```

Kod aplikacji `Przykład.asp` wykorzystuje kilka interesujących sztuczek dostępnych w ADO. Pierwszą z nich, jest sposób pobrania listy tabel za pomocą metody `OpenSchema()`, której wywołanie zostało umieszczone w linii 24. Ta metoda pobiera argument, którym może być dowolna spośród znacznej grupy wartości i w wyniku zwraca zbiór rekordów. Istniej także możliwość podania drugiego argumentu, który pozwala na filtrowanie zwracanych rezultatów. Zwracane mogą być bardzo różne informacje, począwszy od indeksów, a skończywszy na więzach integralności; niemniej jednak w tym przykładzie skoncentrujemy uwagę wyłącznie na zwróceniu tabel. Wartość przekazana jako argument – `20`, zapisana w zmiennej `adSchemaTables` – powoduje, że metoda `OpenSchema()` zawróci zbiór rekordów zawierający wszystkie tabele bazy. Ze względu na fakt, że zwrócone zostaną także informacje o wielu tabelach systemowych, które w większości wypadków nie będą potrzebne, linia 26. wykorzystuje filtr ograniczający zwracany zbiór rekordów, tylko do standardowych tabel bazy danych. To przefiltrowanie wyników mogłoby zostać zrealizowane także poprzez podanie drugiego argumentu w wywołaniu metody `OpenSchema()`.

Po przefiltrowaniu zbioru rekordów, pętla (linie 28. – 47.) analizuje każdy z rekordów, a następnie otwiera zbiór rekordów (linia 35.) wykorzystując przy tym nazwę tabeli umieszczoną w `RSTables`. Następnie pobierane są po kolei wszystkie elementy kolekcji `Fields` obiektu `Recordset`, na podstawie których jest tworzona lista kolumn danej tabeli. Przedstawiony kod może zostać z łatwością zmodyfikowany i wykorzystany do różnych celów.

Podsumowanie

Celem ASP jest niemal wyłącznie przeniesienie informacji z baz danych na strony WWW; ten cel jest realizowany za pomocą ADO. Co więcej, ADO bardzo szybko staje się popularnym narzędziem obsługi baz danych wykorzystywanym w wielu innych śro-

dowiskach programistycznych, w tym także w Visual Basicu. W tym rozdziale nauczyłeś się wykorzystywać ADO i pracować z bazami danych, tworząc przy okazji system forum użytkowników. Tworząc ten przykład, poznałeś obiekty `Connection`, `Command` oraz `Recordset`.

Pytania i odpowiedzi

Dlaczego niektóre z przykładów przedstawionych w tym rozdziale używają kryteriów, które nigdy nie zostaną spełnione (takich jak `WHERE 1=2`)?

Dodając rekordy do tabeli przy wykorzystaniu metody AddNew, zbiór rekordów musi zostać otworzony na danej tabeli. Zadanie to mogłoby spełnić każde zapytanie bazujące na wybranej tabeli, jednak nie chcesz, aby zapytanie zwracało jakiekolwiek wyniki, gdyż nie są one potrzebne. Używając warunku takiego jak `1=2`, możesz być pewny, że nie zostaną zwrócone żadne wyniki, gdyż warunek taki nigdy nie zostanie spełniony. Niemniej jednak, zbiór rekordów zostanie otworzony na wybranej tabeli.

Kiedy można używać kolekcji `Fields` obiektu `Recordset`?

Kolekcja Fields pozwala na operowanie na wszystkich polach tabeli jako elementach grupy. Możesz jej używać, jeśli chcesz pobrać po kolei wszystkie jej elementy i wykonać na nich takie same operacje.

Dlaczego w skryptach ASP nie można używać DAO?

W rzeczywistości można. ADO nie jest jedynym narzędziem obsługi baz danych jakiego możesz używać w skryptach ASP. Wykorzystywać można niemal wszystkie narzędzia przystosowane do pracy w Visual Basic-iem, jednak w większości wypadków nie będą one przystosowane do pracy w środowisku skryptowym działającym po stronie serwera. DAO wykorzystywane w aplikacjach ASP będzie działać poprawnie, jednak efektywność działania będzie niższa niż w przypadku wykorzystania ADO, a obciążenie serwera – większe.

Argumenty wywołania metody `Recordset.Open` można określić poprzez przypisanie wartości odpowiednim właściwościom obiektu `Recordset`. Która z tych metod jest lepsza?

Obie metody są równie dobre. Teoretycznie więcej czasu może zająć przypisywanie wartości właściwościom, jednak jest to wyłącznie kwestia efektywności działania. Określanie poszczególnych właściwości jest rzadziej stosowane, lecz bardziej czytelne.

Warsztat

Pytania kwizowe oraz ćwiczenia mają służyć poprawieniu zrozumienia omawianych zagadnień. Odpowiedzi na pytania znaleźć można w dodatku A, pt. „Odpowiedzi na pytania”. Odpowiedzi na ćwiczenia można znaleźć na witrynie WWW Wydawnictwa HELION.

Kwiz

1. Które z poniższych metod są metodami obiektu Recordset?

   a. `Open`

   b. `Execute`

   c. `Seve`

   d. `Close`

2. Jaki obiekt udostępnia kolekcję `Parameters`?

3. Jaka jest rola kolekcji `Properties`?

4. Prawda czy fałsz: Właściwość `Recordset.Recordcount` zawsze zwraca poprawną ilość rekordów.

5. Jakie dwie rzeczy powinieneś sprawdzić, aby upewnić się, że zbiór rekordów jest całkowicie pusty?

6. Jakiej metody obiektu `Connection` możesz użyć, aby wykonać na bazie danych polecenie SQL?

7. Co powoduje poniższa linia kodu, przy założeniu, że Conn jest obiektem typu Connection:

```
Conn.Open "DNS=FooBar;uid=Fredzio;pwd=Bolo"
```

8. Prawda czy fałsz: Jeśli otworzysz trzy zbiory rekordów, za każdym razem używając do tego tych samych informacji o połączeniu (zamiast obiektu `Connection`), to zbiory te użyją tego samego połączenia.

Ćwiczenia

1. Do strony `Forum.asp` dodaj skrypt, który zmusi użytkowników do zalogowania się do systemu, jeśli jeszcze tego nie zrobili.

2. Przykład systemu forum użytkowników nigdy nie udostępnił użytkownikom możliwości usuwania swoich własnych wiadomości. Dodaj taką możliwość do strony `Wiadomosc.asp`, zapewniając jednocześnie, aby wiadomość mógł usunąć tylko ten użytkownik, który ją stworzył.

3. Pytanie kwizowe numer 7 zawiera wywołanie metody `Open` obiektu `Connection`. Zapisz ten sam kod na dwa inne sposoby.