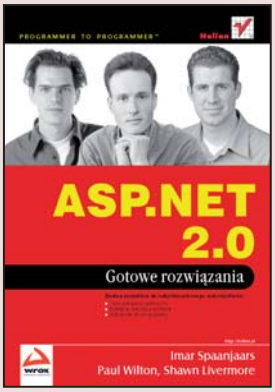
ASP.NET 2.0. Gotowe rozwiązania



Autor: Imar Spaanjaars, Paul Wilton, Shawn Livermore
Tłumaczenie: Michał Dadan
ISBN: 978-83-246-0566-8
Tytuł oryginału: ASP.NET 2.0 Instant Results
Format: 168x237, stron: 488

Zestaw projektów do natychmiastowego wykorzystania!

- Opisy założeń projektowych
- Instrukcje w kwestii instalacji
- Wskazówki dla programistów

Rosnąca popularność platformy .NET 2.0 widoczna jest także w internecie. Coraz więcej aplikacji internetowych powstaje z wykorzystaniem technologii ASP.NET 2.0. Kontrolki i biblioteki udostępniane programistom zdecydowanie ułatwiają budowanie nawet najbardziej złożonych systemów. Biblioteki .NET 2.0 to także zmieniona architektura witryn, nowe mechanizmy dostępu do danych i znacznie większa szybkość działania aplikacji. Jednak tak ogromna liczba zmian wiąże się z koniecznością opanowania przez programistów nowych możliwości i zasad stosowania ich w praktyce. Jednym z najlepszych sposobów nauki jest analiza gotowych projektów i implementowanie ich we własnych systemach.

Książka „ASP.NET 2.0. Gotowe rozwiązania" to przegląd 12 projektów zrealizowanych z wykorzystaniem tej technologii. Znajdziesz tu omówienie najpopularniejszych elementów witryn i aplikacji internetowych – założenia projektowe, kod źródłowy, wykorzystane w projekcie biblioteki i kontrolki, wskazówki dotyczące instalacji i uruchamiania oraz porady związane z samodzielnym modyfikowaniem omawianego modułu. Każdy z projektów przedstawia różne aspekty stosowania technologii ASP.NET 2.0 i stanowi doskonałe źródło wiedzy.

- Internetowy dziennik i kalendarz
- System wymiany plików
- Serwer chatów
- Mechanizm obsługi ankiet
- CMS
- Blog
- Album fotograficzny
- Witryna pomocy dla klienta
- Sklep internetowy
- System rezerwacji online
- Kartki internetowe
- Baza błędów

Poznaj ASP.NET 2.0 na praktycznych przykładach

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

Spis treści

4

Mechanizm obsługi ankiet

Jeżeli kiedykolwiek brałeś udział w internetowej ankiecie, zapewne wiesz, jak ciekawe potrafi być oglądanie wyników, zwłaszcza jeśli przyglądając się im, możemy kiwnąć potakująco głową. Ankiety intrygują gości witryny, ponieważ są one nastawione na zbieranie i kompilowanie informacji od opinii publicznej. Na scenie politycznej toczą się ciągłe spekulacje w oparciu o wyniki ankiet przeprowadzanych na wybranych grupach ludzi. Zmiany w firmach i przepływ inwestycji również uzależnione są od informacji zebranych w ankietach. Na pewno zauważyłeś też, że pytania w ankietach są formułowane tak, aby trafiały do przeciętnego użytkownika. Stanowią one najlepszy sposób pozyskiwania informacji od użytkowników internetu i dlatego można je spotkać w całej sieci. Za ich pośrednictwem dane zbierają zarówno małe, jak i duże firmy. Użytkownikowi przedstawia się proste i zrozumiałe pytania oraz listę kilku możliwych odpowiedzi. Wyniki ankiety są generowane na bieżąco i udostępniane kierownictwu firmy, która ją zorganizowała, co pozwala mu podejmować właściwe decyzje. Tak więc stworzenie internetowej ankiety w celu zebrania informacji o prawdziwym stanie danej grupy lub sektora rynku ma jak najbardziej sens.

Mechanizm obsługi ankiet, który stworzymy, to ciekawa aplikacja. Można ją z powodzeniem stosować wszędzie tam, gdzie mamy do czynienia z szeregiem pytań, na które można udzielić odpowiedzi z określonego zbioru. Po przeprowadzeniu ankiety powinniśmy mieć możliwość obejrzenia zebranych informacji i ułożenia ich w czytelnym formacie. Na podstawie wyników ankiety powinniśmy mieć też możliwość generowania raportów, co ułatwi nam demonstrowanie zdania większości.

Mechanizm obsługi ankiet to wspaniały projekt przykładowy, w oparciu o który można dowiedzieć się naprawdę wiele o ASP.NET 2.0. Strona do zarządzania ankietami, którą stworzymy, daje administratorowi możliwość tworzenia ankiet i monitorowania ich wyników za pośrednictwem przyjaznego interfejsu użytkownika.

Mechanizm obsługi ankiet oferuje wiele przydatnych funkcji, takich jak:

- Tworzenie ankiet na bieżąco
- Przeglądanie listy wyników w zestawieniu procentowym
- Osadzanie ankiet w istniejących witrynach internetowych

Są to główne funkcje, na implementacji których się skupimy. Zostawiają one wiele miejsca na modyfikacje i rozszerzenia.

W tym rozdziale pokażemy, jak łatwo implementuje się niektóre nowe kontrolki i techniki dostępne w ASP.NET 2.0. Powiemy na przykład o kontrolce `ObjectDataSource`, rozszerzonej kontrolce `DataSource` SQL Servera, interfejsie bezpieczeństwa ASP.NET Web Security Interface, stosowaniu motywów z poziomu pliku *.config*, nowych kontrolkach nawigacyjnych, kontrolkach logowania i odzyskiwania hasła oraz stosowaniu szablonów stron.

W podrozdziale „Projekt mechanizmu obsługi ankiet" dokładniej przyjrzymy się projektowi aplikacji. Omówimy sobie strukturę plików w bazie danych, projekty klas, podstawowy model dziedziczenia i luźną architekturę tej aplikacji.

W podrozdziale „Kod i jego objaśnienie" dokonamy metodycznej analizy kodu, zagłębiając się w poszczególne moduły i funkcje i wyjaśniając ich rolę, od interfejsu użytkownika, przez bazę danych i z powrotem. Ponadto przeanalizujemy logikę klas i zasugerujemy możliwe rozszerzenia systemu.

W ostatnim podrozdziale, „Konfiguracja projektu", napiszemy, jak zainstalować mechanizm obsługi ankiet w środowisku programistycznym i przystosować go do swoich potrzeb.

Zacznijmy jednak od początku, to znaczy od ogólnego spojrzenia na działanie mechanizmu obsługi ankiet.

Korzystanie z mechanizmu obsługi ankiet

Posługiwanie się mechanizmem obsługi ankiet jest bardzo proste. Jest to w zasadzie strona internetowa, która ma za zadanie ułatwić firmom lub osobom prywatnym proste i szybkie tworzenie internetowych ankiet. Każdy badany będzie musiał obejrzeć tylko kilka podstron, ponieważ cała ankieta będzie składała się zaledwie z kilku pytań oraz listy odpowiedzi, z których żadna nie jest zła. Sprawia to, że ankiety są idealnym kandydatem do stworzenia komponentu lub modułu wielokrotnego użytku, który będzie je generował. Załóżymy, że na każde pytanie ankiety będą do wyboru dokładnie 4 odpowiedzi.

Jeżeli strona mechanizmu obsługi ankiet została już poprawnie zainstalowana (patrz podrozdział „Konfiguracja projektu" w dalszej części tego rozdziału), możesz obejrzeć ją w przeglądarce internetowej po uruchomieniu w programie Visual Web Developer. Twoim oczom ukaże się strona pokazana na rysunku 4.1.

W górnej części strony domowej dostępne są następujące łącza:

- *Strona główna*
- *O programie*
- *Kontakt*
- *Administracja*

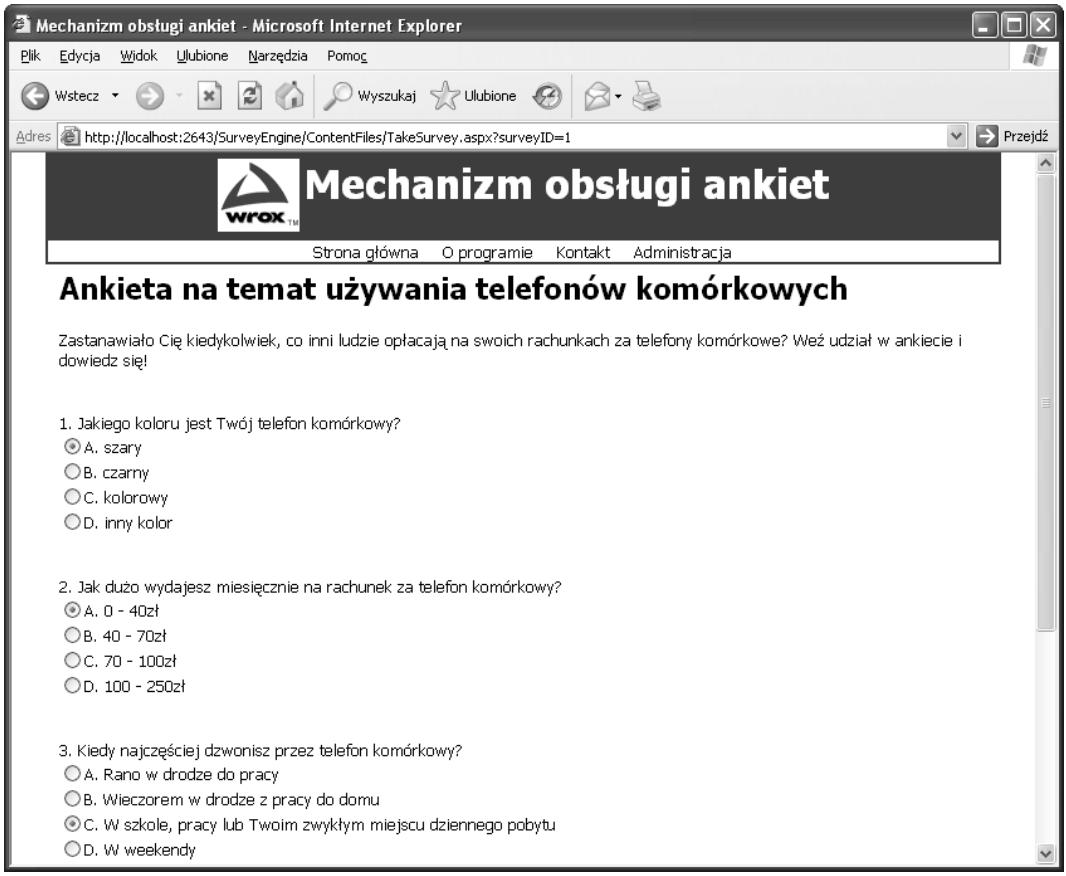
Rysunek 4.1

W dolnej części strony głównej znajdziesz też prostokąt o zaokrąglonych narożnikach, który pełni rolę hiperłącza prowadzącego do gotowej ankiety. Po jego kliknięciu zostaniesz przeniesiony na stronę, na której wyświetlone są wszystkie pytania jednocześnie (patrz rysunek 4.2).

Użytkownikowi pokazywana jest lista czterech odpowiedzi (A, B, C, D) na każde pytanie, z której może wybrać tę, która najbardziej odpowiada jego przekonaniom. Po zaznaczeniu odpowiedzi na wszystkie pytania można kliknąć przycisk *Zobacz wyniki*, co spowoduje wyświetlenie strony pokazanej na rysunku 4.3.

Na tym etapie, po wysłaniu wypełnionej ankiety, będzie można zapoznać się z dotychczas zebranymi wynikami. Oczywiście w miarę jak w badaniu będzie brało udział więcej osób, wyniki będą się zmieniać.

Na tym w zasadzie kończy się opis głównej części naszej aplikacji, z którą będzie się stykać większość użytkowników. Równie ważne jest jednak to, co kryje się „od kuchni” — sekcja administracyjna, która umożliwia szybkie i łatwe tworzenie ankiet i przeglądanie odpowiedzi.



Rysunek 4.2

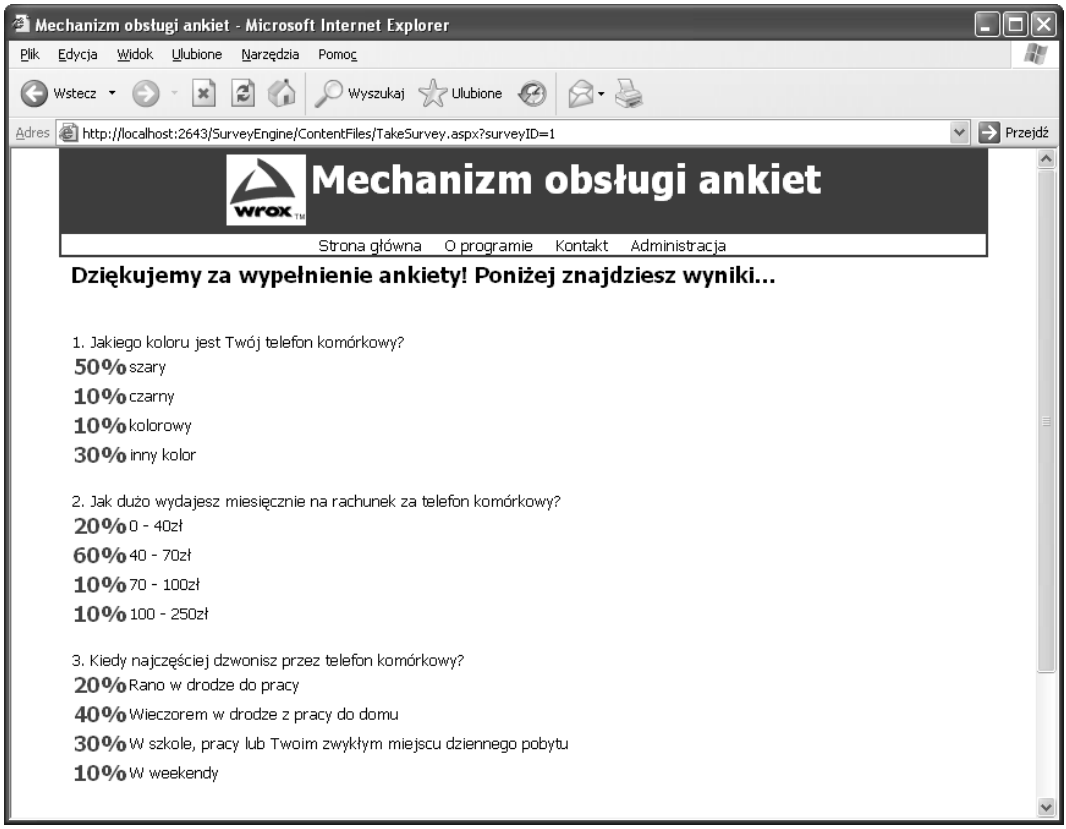
Po kliknięciu hiperłącza *Administracja*, znajdującego się w głównym menu, użytkownik zostaje przeniesiony na ekran logowania, chyba że zdążył się już zalogować i utworzyć sesję. Interfejs logowania pokazaliśmy na rysunku 4.4.

Strona ta udostępnia mechanizm logowania i mechanizm przypominania hasła. Wprowadź nazwę użytkownika `Admin`, hasło `password#` i kliknij przycisk *Zaloguj*.

Po zalogowaniu się w witrynie zostaniesz przeniesiony na stronę administracyjną, pokazaną na rysunku 4.5.

Zawiera ona tabelę ze wszystkimi ankietami, jakie istnieją w systemie, i umożliwia przeprowadzanie czynności administracyjnych, takich jak:

- Tworzenie nowej ankiety
- Edycja pytań lub nazwy istniejącej ankiety
- Dodawanie pytań do istniejącej ankiety



Rysunek 4.3

- Przeglądanie udzielonych odpowiedzi
- Oznaczanie wybranej ankiety jako gotowej do wyświetlenia na stronie

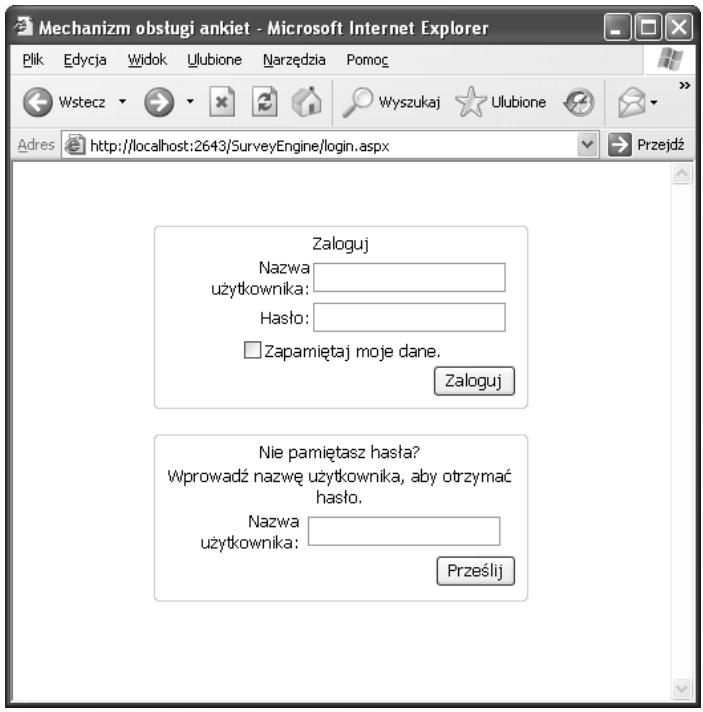
Klikając wybrane hiperłącza w tabeli, przechodzisz na strony, które umożliwiają wykonanie wybranych czynności. Każdą z nich opisaliśmy w kolejnych podrozdziałach.

Dodawanie nowej ankiety

Jedną z pierwszych funkcji, jakie pewnie będziesz chciał wywołać, jest Kreator dodawania ankiety, do którego można się dostać za pośrednictwem hiperłącza widocznego w lewym dolnym rogu strony lub dowolnego łącza Nowa widocznego w tabeli. Po kliknięciu któregoś z tych elementów pojawi się Kreator dodawania ankiety i poprosi o wprowadzenie danych tworzonej ankiety (patrz rysunek 4.6).

Na rysunku pokazaliśmy pierwszy etap kreatora. W prawym dolnym rogu ekranu widać przycisk Dalej, który jest nową kontrolką w ASP.NET 2.0. W następnym podrozdziale, „Projekt mechanizmu obsługi ankiet”, omówimy szczegóły i przypadki użycia.

Rysunek 4.4



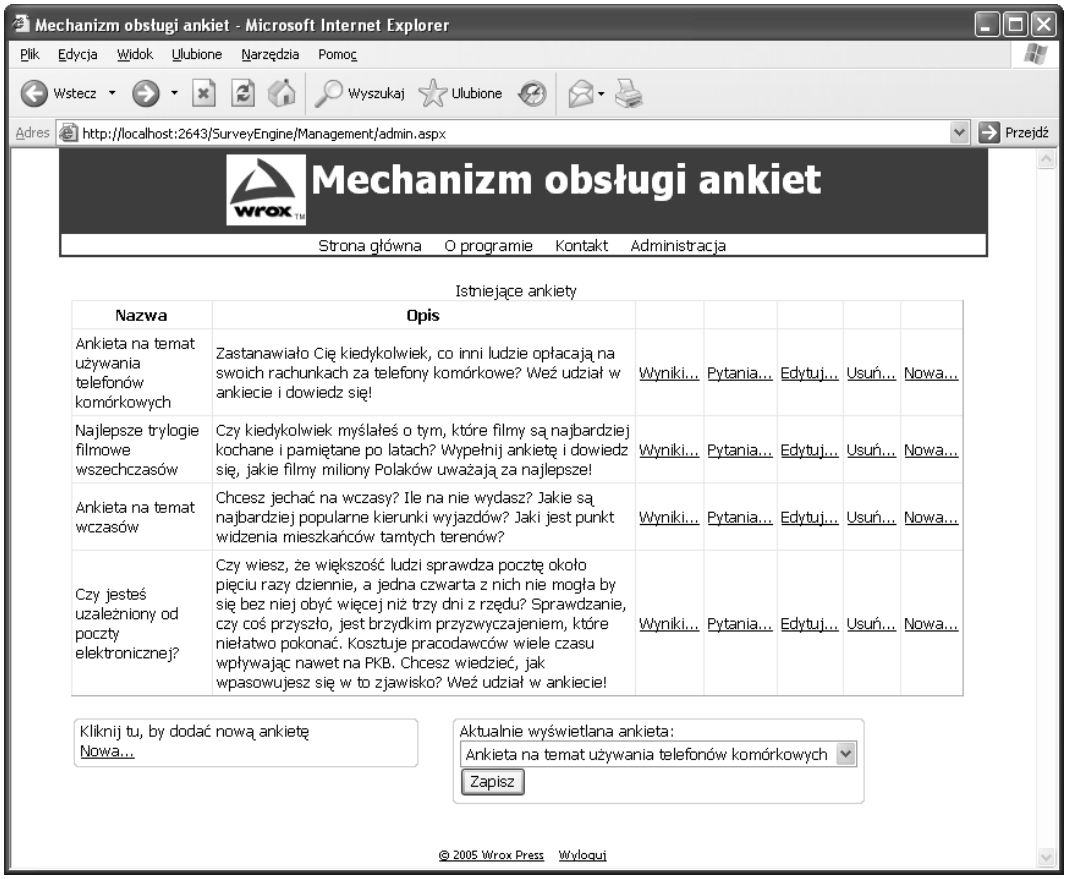
Klikając *Dalej*, możemy wprowadzić nazwę nowej ankiety (patrz rysunek 4.7).

Nazwę nowej ankiety można wprowadzić w polu tekstowym widocznym w górnej części ekranu. Nasza przykładowa ankieta będzie nosiła tytuł „Czy jesteś uzależniony od poczty elektronicznej?". Aby kontynuować po wprowadzeniu nazwy, należy kliknąć *Dalej*. Pojawi się wówczas następny krok kreatora, pokazany na rysunku 4.8.

Na rysunku 4.8 widać wielowierszowe pole tekstowe, wykorzystywane do przechowywania długich opisów ankiet, które użytkownicy będą mogli zobaczyć tuż pod tytułem (nazwą) ankiety. Po wprowadzeniu opisu kliknij *Dalej*, aby przejść na stronę *Dodaj pytania*, pokazaną na rysunku 4.9.

Możesz wprowadzać za każdym razem po jednym pytaniu (wraz z towarzyszącym mu zestawem odpowiedzi), klikając następnie przycisk *Zapisz pytanie* w celu zachowania go w bazie danych. Po wprowadzeniu wszystkich pytań kliknij *Dalej*. W tym momencie wszystkie pytania ankiety będą już zapisane w bazie danych. Ostatnia strona Kreatora dodawania ankiety została pokazana na rysunku 4.10.

Poza tworzeniem nowych ankiet masz też możliwość modyfikowania już istniejących.



Rysunek 4.5

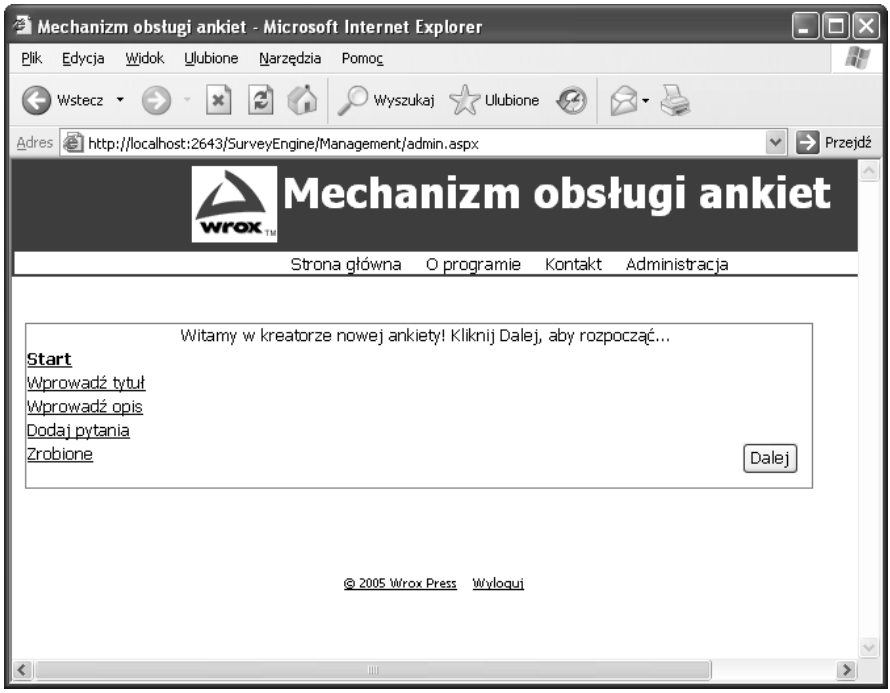
Edytowanie istniejącej ankiety

Jeżeli chcesz zmodyfikować istniejącą ankietę, możesz to zrobić, wracając na stronę administracyjną. W tym celu kliknij hiperłącze *Administracja* w głównym menu. Następnie kliknij jedno z hiperłączy *Pytania* wyświetlanych w wierszach tabeli ankiet. Spowoduje to wczytanie strony służącej do zarządzania istniejącymi ankietami, pokazanej na rysunku 4.11.

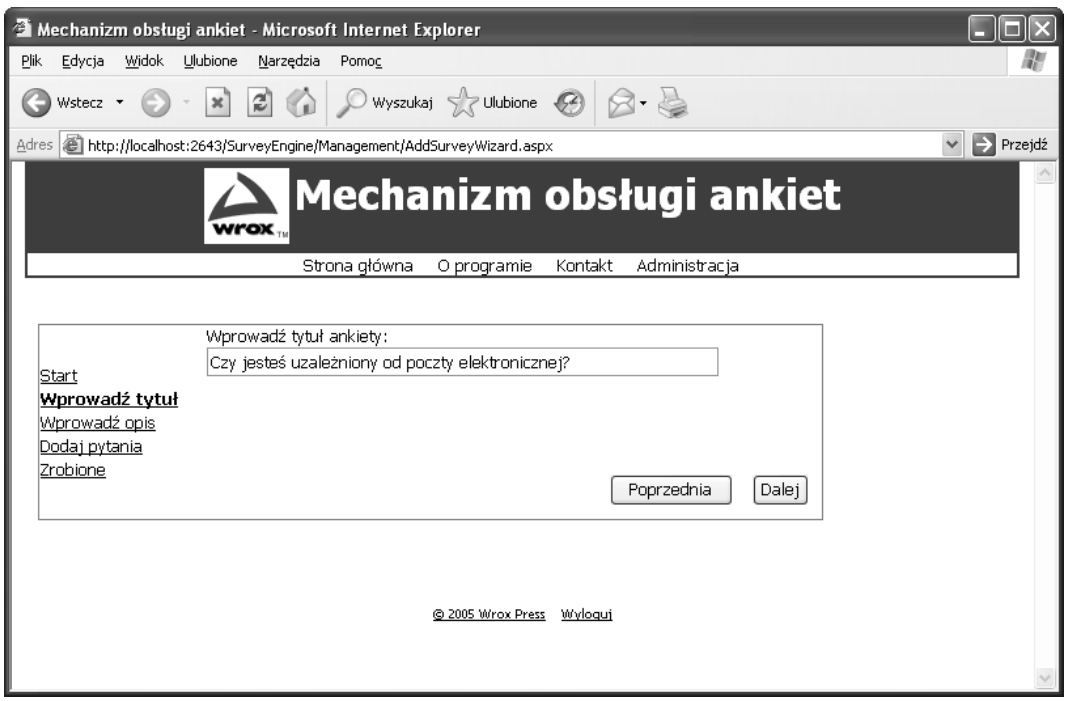
Na rysunku 4.11 widać wszystkie pytania danej ankiety, wraz z hiperłączami, które umożliwiają dodawanie, edycję i usuwanie pytań.

Możemy więc podsumować podstawowe zadania mechanizmu obsługi ankiet w następujący sposób: umożliwienie tworzenia i wyświetlania ankiet na stronach internetowych oraz udostępnianie narzędzi do monitorowania i kontrolowania ankiet w miarę napływania odpowiedzi.

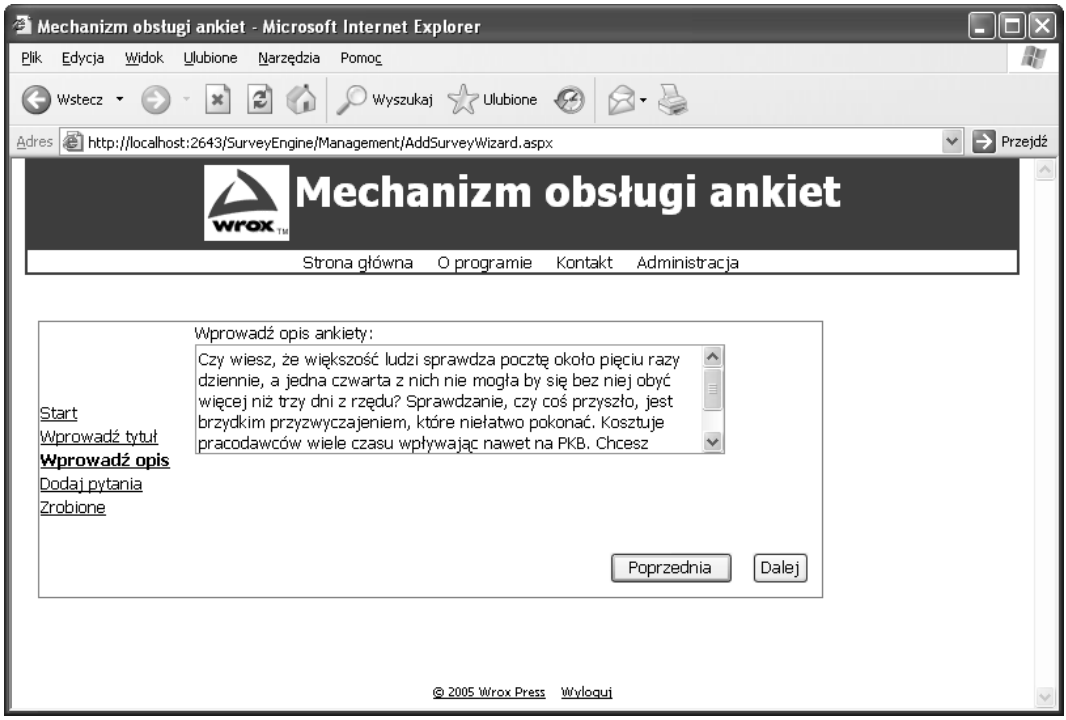
W następnym podrozdziale przyjrzymy się tym fragmentom aplikacji, które są szczególnie wymagające pod względem technicznym, i pokażemy, jak łączą się one w jedną całość. Zobaczysz, jak wygląda modelowanie klas i gdzie znajdują się ważne elementy konstrukcyjne projektu.



Rysunek 4.6



Rysunek 4.7



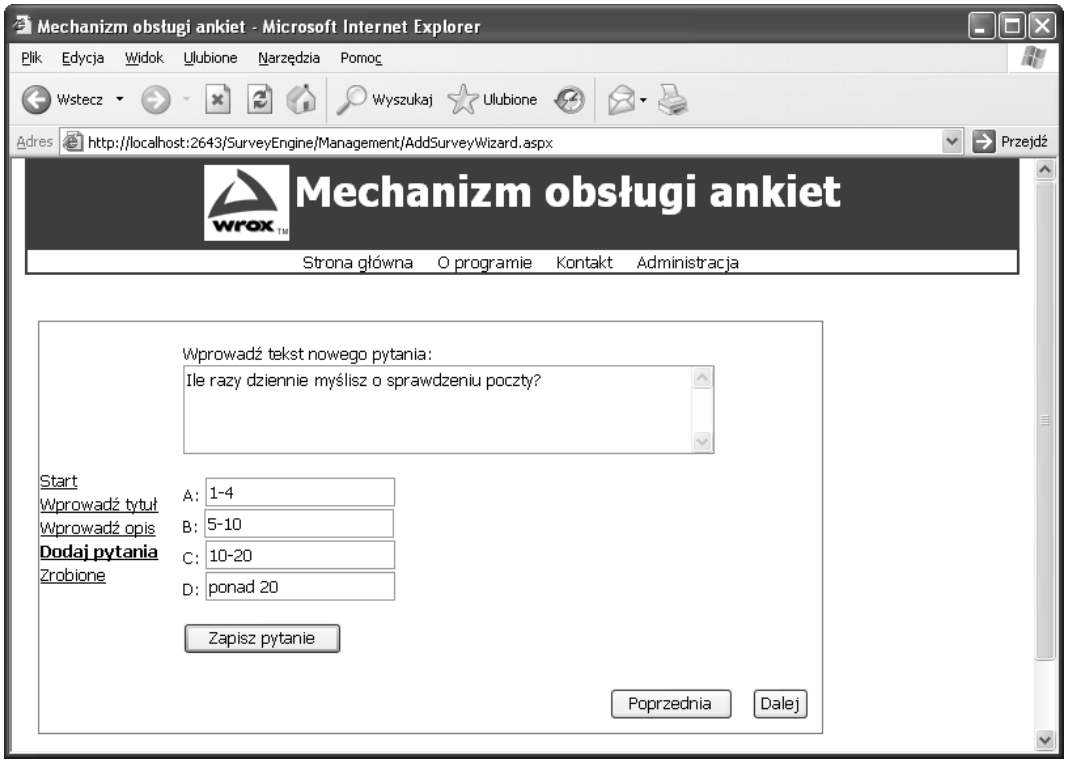
Rysunek 4.8

Projekt mechanizmu obsługi ankiet

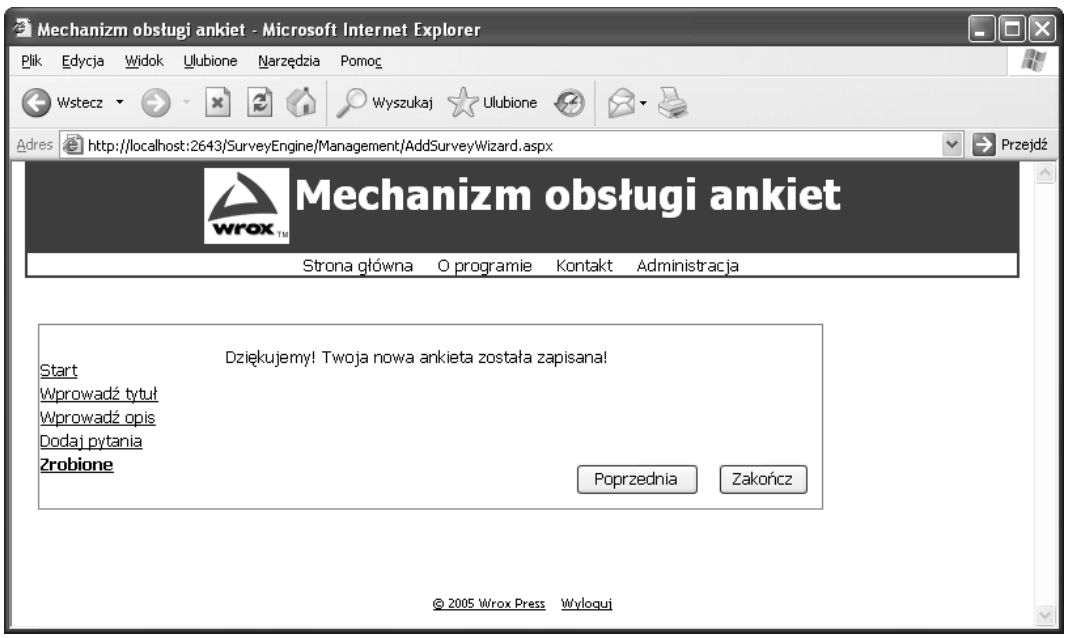
W tym podrozdziale przyjrzymy się dokładnie wewnętrznym mechanizmom naszej internetowej aplikacji, skupiając się na zastosowanych klasach i połączeniach między kontrolkami do obsługi danych a obiektami i danymi SQL Servera.

Projekt mechanizmu obsługi ankiet jest zorientowany obiektowo i podzielony na logiczne *warstwy*. Zastosowano tu do pewnego stopnia logiczną abstrakcję, wydzielając warstwę klienta, biznesową i danych. Dzięki temu programista może łatwiej wyobrazić sobie, jak mógłby przebiegać podział aplikacji na kilka projektów, serwerów i (lub) lokalizacji. W tym podrozdziale opiszemy też dokładnie dwa sposoby wiązania danych z elementami formularzy — kontrolki `DataSource` SQL Servera i `ObjectDataSource`.

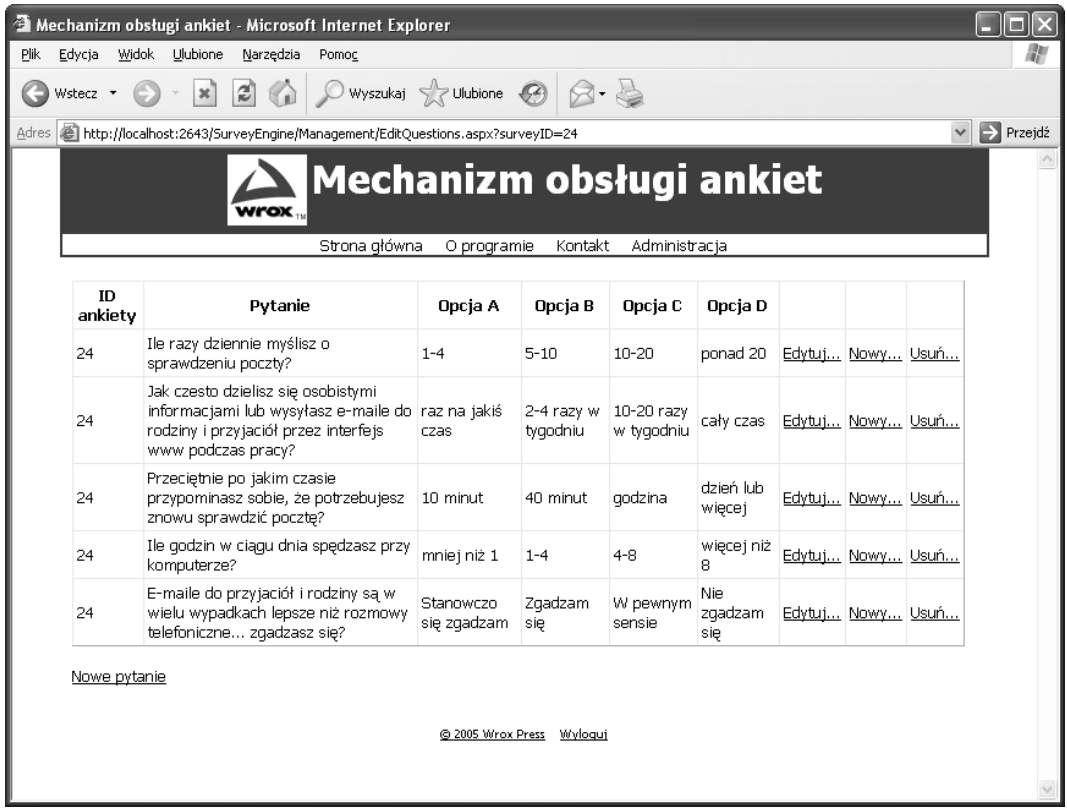
Zagadnieniem tym, a także pozostałymi aspektami architektury aplikacji, zajmiemy się w dalszej części tego podrozdziału.



Rysunek 4.9



Rysunek 4.10



ID ankiety	Pytanie	Opcja A	Opcja B	Opcja C	Opcja D			
24	Ile razy dziennie myślisz o sprawdzeniu poczty?	1-4	5-10	10-20	ponad 20	Edytuj...	Nowy...	Usuń...
24	Jak czesto dzielisz się osobistymi informacjami lub wysyłasz e-maile do rodziny i przyjaciół przez interfejs www podczas pracy?	raz na jakiś czas	2-4 razy w tygodniu	10-20 razy w tygodniu	cały czas	Edytuj...	Nowy...	Usuń...
24	Przeciętnie po jakim czasie przypominasz sobie, że potrzebujesz znowu sprawdzić pocztę?	10 minut	40 minut	godzina	dzień lub więcej	Edytuj...	Nowy...	Usuń...
24	Ile godzin w ciągu dnia spędzasz przy komputerze?	mniej niż 1	1-4	4-8	więcej niż 8	Edytuj...	Nowy...	Usuń...
24	E-maile do przyjaciół i rodziny są w wielu wypadkach lepsze niż rozmowy telefoniczne... zgadzasz się?	Stanowczo się zgadzam	Zgadzam się	W pewnym sensie	Nie zgadzam się	Edytuj...	Nowy...	Usuń...

Rysunek 4.11

Wiązanie obiektów i danych SQL Servera

Jeżeli aplikacja ma być pisana szybko i wydajnie, zawsze trzeba iść na jakiś kompromis. Na etapie planowania i projektowania projektu może pojawić się pokusa skorzystania z szybkich i „lekkich" kontrolek interfejsu użytkownika, które wykonują większość czynności za programistę i pozwalają zaoszczędzić mnóstwo godzin pracy. Chodzi na przykład o kontrolki `GridView` i `DataList`. Dzięki nim możemy po prostu przeciągnąć kontrolki na formularz *WebForm*, ustawić żądane właściwości w widoku *Design View* i uruchomić aplikację. Bez napisania choćby jednego wiersza kodu VB.NET i C# możemy skonfigurować kontrolkę źródła danych, tak aby w pełni wyświetlała się na stronie internetowej. Nowa kontrolka ASP.NET 2.0 sama tworzy za programistę obiekty ADO.NET `Connection`, `Command`, `DataSet` bądź `DataReader`, zapewniając wszystko, co potrzeba, aby połączyć się z danymi w czasie wykonywania programu. Potrafi ona też prawidłowo obsłużyć zdarzenia związane z korzystaniem z danych na stronie — coś, czego nie oferowało ASP.NET 1.1. Są to wspaniałe funkcje, z których warto korzystać. Jednak zdajemy sobie sprawę z tego, że wielu programistów czytających tę książkę będzie chciało poznać bardziej rozbudowane funkcje ASP.NET 2.0 i wkroczyć tym samym w nowy, ulepszony świat .NET. Z myślą o Czytelnikach zainteresowanych tworzeniem w ASP.NET 2.0 dużych projektów warto wspomnieć o zaletach i wadach, jakie niesie ze sobą stosowanie bardziej skalowalnych architektur, i o tym, co zyskujemy, a co tracimy, stosując

w ASP.NET 2.0 każdą z dostępnych metodologii. Skalowalność architektury ma silny związek z obciążeniem, jakie jest ona w stanie udźwignąć w sytuacji, gdy w grę wchodzą duże ilości danych lub gdy wymagana jest spora ilość przetwarzania. Skalowalność odnosi się też do stopnia kontroli, jaki mamy nad przebiegiem przetwarzania, kierowania ruchem i obsługą danych. ASP.NET 2.0 obsługuje wszystkie te elementy, oferując przy tym wiele korzyści, które czynią tę platformę idealnym rozwiązaniem do rozwijania aplikacji.

Prawdziwie wielowarstwowe architektury spotyka się w przemyśle informatycznym raczej w dużych aplikacjach korporacyjnych. Są one zazwyczaj bardziej ogólne i oparte na wzorcach. Warstwa użytkownika, biznesowa i danych stanowią wówczas niezależne sekcje aplikacji, mogące funkcjonować niezależnie od pozostałych. Taka jest natura programowania rozproszonego, które ostatnio ewoluowało w kierunku tak zwanych *aplikacji kompozytowych*. W projektowaniu rozproszonym logika dostępu do bazy danych byłaby umieszczona jedynie w tej warstwie kodu (w tych klasach), która ma jawne zadanie odwoływania się do danych. Tylko te konkretne klasy pozwalałyby wyciągać informacje z bazy danych, pełniąc rolę pośredników dla pozostałych modułów i klas. W związku z tym korzystanie z obiektów biznesowych (takich jak obiekty klasy `Survey`) umożliwiałoby przekazanie żądania danych do innych klas w warstwie danych (na przykład od klasy `SurveyDB`), a następnie zwrócenie danych z powrotem do klienta i związanie ich z siatką danych, polem listy itd. W przypadku większości aplikacji takie podejście jest akceptowalne.

Jeśli chodzi o wiązanie obiektów i danych, zauważysz, że w naszym mechanizmie obsługi ankiet wykorzystujemy wbudowane kontrolki ASP.NET o nazwie `ObjectDataSource` w celu związania danych z obiektów biznesowych z kontrolkami graficznego interfejsu użytkownika. W ten sposób naśladujemy wielowarstwowe, obiektowe podejście, które w ostatnich latach uważane jest za najlepszą praktykę programistyczną, choć nie zapewniamy wszystkich nietypowych i wydajnych mechanizmów, które mogą być wymagane w rozwiązaniach korporacyjnych. Musimy mieć możliwość serializacji obiektów niektórych klas i przekazywania ich za pośrednictwem połączenia internetowego między warstwami, dzięki czemu będziemy mogli stosować niezależne środowiska serwerowe. Obiekty innych klas i moduły będą zarządzane za pośrednictwem mechanizmu WMI lub monitorów wydajności ASP.NET (`PerfMon`). Te dodatkowe wymagania na poziomie obiektów można spełnić stosując kontrolki `ObjectDataSource`, ale w dalszym ciągu będzie to wymagało ręcznego programowania, o czym warto wspomnieć w tej książce.

Poza kontrolkami `ObjectDataSource` mechanizm obsługi ankiet wykorzystuje też kontrolki `DataSource` SQL Servera w celu wydobywania danych z plików bazy danych SQL Server Express 2005 i wiązania ich z kontrolkami interfejsu użytkownika. Wbrew tym założeniom, kontrolki `SqlDataSource` zazwyczaj są projektowane jako szybki i prosty sposób wybierania rekordów z bazy danych i wiązania ich z formularzem. W procesie tym stosuje się wyrażenia języka SQL, umieszczane wśród znaczników plików ASPX. Stoi to w jawnej sprzeczności z bezpieczniejszym i bardziej rozproszonym podejściem wykorzystującym luźno powiązane warstwy i stanowi zagrożenie dla aplikacji i łatwości zarządzania nią. W miarę jak z takich stron zaczyna korzystać coraz więcej użytkowników, tworzonych jest coraz więcej połączeń z bazą danych (w zależności od zastosowanych łańcuchów inicjujących połączenie). Może to ostatecznie doprowadzić do utraty skalowalności i szybkości i stanowić zagrożenie dla witryn o dużym natężeniu ruchu. Korzystanie z kontrolek `SqlDataSource` wymaga też uaktualniania plików źródłowych ASPX witryny za każdym razem, gdy w bazie danych wprowadzane są zmiany, które mają wpływ na zapytania i procedury przechowywane na serwerze. Wcale nie

oznacza to, że aplikacja musi być całkowicie przewidywalne i łatwa do zarządzania, ale — jak piszemy również w innych rozdziałach — tego rodzaju zagrożenia często nie są brane pod uwagę, a programiści skupiają się na stosowaniu kontrolek `ObjectDataSource` i `SqlDataSource`.

Struktura witryny

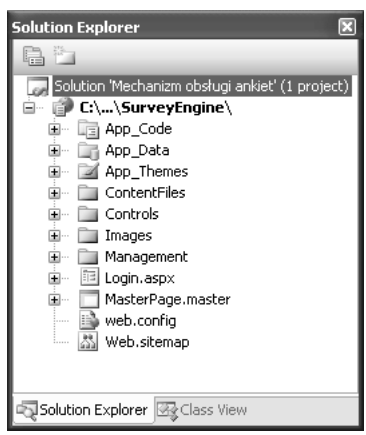
Struktura witryny jest bardzo poukładana. Wszystkie pliki zostały rozmieszczone w folderach tak, aby utrzymywanie kodu było jak najwydajniejsze. Folder *Controls* przechowuje wszystkie kontrolki użytkownika, a w folderze *ContentFiles* znajdują się główne pliki ASPX formularzy WebForm witryny.

W poniższej tabeli wymieniamy poszczególne foldery składające się na poszczególne sekcje aplikacji internetowej:

Folder	Opis
App_Code	Przechowuje klasę warstwy biznesowej (`SurveyDB.vb`).
App_Data	Standardowy folder .NET na pliki bazy danych.
App_Themes	Folder motywów, przechowujący dwa motywy, które można stosować w witrynie.
ContentFiles	Standardowe pliki ASPX formularzy WebForm służących do wyświetlania zawartości.
Controls	Przechowuje wszystkie kontrolki użytkownika.
Images	Przechowuje obrazki wykorzystywane w nagłówku lub na podstronach.
Management	Przechowuje zabezpieczone strony administracyjne WebForm.
Pliki różne	Do „plików różnych” zaliczamy stronę logowania, plik *Web.config*, plik z mapą witryny, a także plik z szablonem stron umieszczony w głównym katalogu witryny.

Na rysunku 4.12 pokazaliśmy, jak wyglądają foldery i pliki aplikacji z punktu widzenia programisty, oglądane w panelu *Solution Explorer*.

Rysunek 4.12



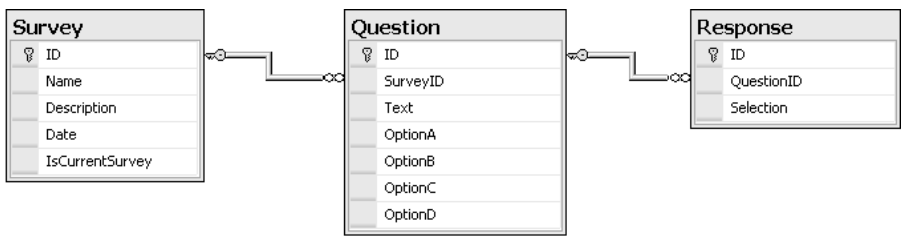
Następny podrozdział opisuje najważniejsze wykorzystywane elementy bazodanowe i sposób zaimplementowania poszczególnych cech ankiet w bazie danych.

Model danych i obiekty bazy danych

Model danych ma bardzo prostą naturę i składa się zasadniczo z trzech elementów:

- Ankiety
- Pytania
- Udzielone odpowiedzi

Każda ankieta zawiera pytania i odpowiedzi, które są zaznaczane przez osoby odwiedzające stronę. Po ich zaznaczeniu następuje wygenerowanie „udzielonych odpowiedzi". Rysunek 4.13 ukazuje diagram przedstawiający wykorzystywane tabele bazy danych.



Rysunek 4.13

Przejdziemy teraz do szczegółowego omówienia każdej z trzech tabel.

W poniższej tabeli omawiamy zawartość tabeli `Question`:

Nazwa pola	Typ danych	Opis
`ID`	`Int`	Unikatowy identyfikator rekordu.
`SurveyID`	`Int`	Ankieta, do której przynależy dane pytanie.
`Text`	`varchar(1000)`	Treść pytania.
`OptionA`	`varchar(1000)`	Pierwsza z czterech dostępnych odpowiedzi.
`OptionB`	`varchar(1000)`	Druga z czterech dostępnych odpowiedzi.
`OptionC`	`varchar(1000)`	Trzecia z czterech dostępnych odpowiedzi.
`OptionD`	`varchar(1000)`	Czwarta z czterech dostępnych odpowiedzi.

W kolejnej tabeli omawiamy tabelę `Survey`:

Nazwa pola	Typ danych	Opis
`ID`	`Int`	Unikatowy identyfikator rekordu.
`Name`	`varchar(200)`	Nazwa nadana ankiecie.

Nazwa pola	Typ danych	Opis
Description	varchar(1000)	Opis ankiety.
Date	Datetime	Znacznik daty i czasu z chwili utworzenia ankiety.
IsCurrentSurvey	Char(1)	Wartość 0 lub 1 określająca, czy ankieta jest aktualnie prezentowana w witrynie. 1 oznacza, że dany rekord definiuje bieżącą ankietę, a 0, że nie.

Poniższa tabela omawia zawartość tabeli Response:

Nazwa pola	Typ danych	Opis
ID	Int	Unikatowy identyfikator rekordu.
QuestionID	Int	Identyfikator pytania, do którego odnosi się ta odpowiedź.
Selection	Char(1)	Wartość A, B, C, lub D oznaczająca odpowiedź udzieloną przez użytkownika na dane pytanie.

Poza tymi trzema tabelami aplikacja wykorzystuje szereg procedur przechowywanych na serwerze. Ich nazwy powstały w oparciu o te same konwencje nazewnicze, które stosujemy w pozostałych rozdziałach:

- sproc*NazwaTabeli*SelectList
- sproc*NazwaTabeli*SelectSingleItem
- sproc*NazwaTabeli*InsertUpdateItem

Zgodnie z tymi założeniami procedurom przechowywanym na serwerze nadaliśmy następujące nazwy:

- sprocQuestionDeleteSingleItem
- sprocQuestionInsertUpdateItem
- sprocQuestionSelectList
- sprocResponseInsertItem
- sprocSurveyInsertUpdateItem
- sprocSurveySaveSingleItemAsCurrent
- sprocSurveySelectList
- sprocSurveySelectSingleItem
- sprocSurveySelectSingleItemWhereCurrent

Jak widać, przyjęta konwencja nazewnicza pozwala nam szybko i łatwo odnaleźć procedury przechowywane na serwerze, odnoszące się do konkretnej tabeli i ustalić, czy realizują one operację wybierania, wstawiania, uaktualniania, czy usuwania rekordów.

Warto przyjrzeć się wielu spośród tych procedur przechowywanych na serwerze. Pierwsza z nich pobiera z bazy danych pojedynczy rekord opisujący ankietę i wyświetla go na stronie głównej *Default.aspx* za pośrednictwem kontrolki CurrentSurvey:

```
ALTER PROCEDURE dbo.sprocSurveySelectSingleItemWhereCurrent
/*'=================================================================
 '  NAZWA:                 sprocSurveySelectSingleItemWhereCurrent
 '  DATA UTWORZENIA:   5 października 2005
 '  UTWORZONA PRZEZ:   Shawn Livermore (shawnlivermore.blogspot.com)
 '  UTWORZONA DLA:     ASP.NET 2.0 - Gotowe rozwiązania
 '  FUNKCJA:               Zwraca z bazy danych 'bieżącą' ankietę.
 '=================================================================
*/
as

select top 1 * from Survey where iscurrentsurvey = 1
```

Jak widać, w przypadku tego projektu stopień złożoności procedur przechowywanych na serwerze został ograniczony do minimum. Prosta struktura tabel aplikacji odpowiada częściowo za łatwość posługiwania się nią i niski poziom złożoności projektowej.

Następna procedura służy do pobrania wszystkich pytań przypisanych do ankiety o danym identyfikatorze:

```
ALTER PROCEDURE dbo.sprocQuestionSelectList
/*'=================================================================
 '  NAZWA:                 sprocQuestionSelectList
 '  DATA UTWORZENIA:   5 października 2005
 '  UTWORZONA PRZEZ:   Shawn Livermore (shawnlivermore.blogspot.com)
 '  UTWORZONA DLA:     ASP.NET 2.0 - Gotowe rozwiązania
 '  FUNKCJA:               Zwraca z bazy danych wszystkie pytania i
 '                             dostępne odpowiedzi dla danej ankiety
 '=================================================================
*/
(@id int)

as

SELECT Question.SurveyID, Question.Text, Question.OptionB, Question.OptionA,
Question.OptionD, Question.OptionC, Question.ID
FROM Survey INNER JOIN Question ON Survey.ID = Question.SurveyID
WHERE (Survey.ID = @id)
```

Są to proste, przykładowe procedury przechowywane na serwerze, ale są one reprezentatywne dla tego rodzaju aplikacji.

Poza tabelą i procedurami przechowywanymi na serwerze mechanizm obsługi ankiet wykorzystuje też widoki, uwidaczniające dość złożone zapytania wykorzystywane do ujmowania wyników ankiet w zestawienia procentowe. Widoki te to:

- `viewAnswerPercentByQuestion`
- `viewAnswerSumByQuestion`
- `viewNumberResponsesBySurvey`
- `viewQuestionCountBySurvey`
- `viewResponseCountBySurvey`

Są one wykorzystywane łącznie, gdyż zależą od siebie wzajemnie. Jeden widok wykorzystuje lub wskazuje pola innego widoku. Ostateczny wynik widoków prezentowany jest przez `viewAnswerPercentByQuestion`, który jest wykorzystywany przez kontrolkę `SurveyResults`.

Motywy i skórki

Projekt udostępnia prosty sposób przypisywania każdej stronie witryny wybranego motywu lub skórki, bez konieczności modyfikowania kodu HTML stron (dotyczy to nawet samego szablonu stron). Aby nadać całej witrynie wybrany motyw graficzny, wystarczy zmodyfikować plik *Web.config*, wskazując w nim nazwę żądanego motywu (przy założeniu, że istnieje on w ramach projektu i został umieszczony w folderze *App_Themes*). Motyw wczytywany jest na każdym formularzu ASP.NET za pośrednictwem poniższego kodu obsługi zdarzeń preinicjalizacyjnych:

```
Protected Sub Page_PreInit(ByVal sender As Object, ByVal e As System.EventArgs)
↪Handles Me.PreInit
        'procedura obsługi tego zdarzenia preinicjalizacyjnego jest wywoływana w celu
        'zainicjalizowania strony. Pozwala ona ustawić tytuł i motyw strony. Informacje
        'o nich pochodzą z ustawień w pliku web.config, które są wyciągane ze współ-
        'użytkowanych, publicznych właściwości klasy Config.
        Page.Theme = Config.CurrentTheme
        Page.Title = Config.PageTitle
End Sub
```

Kod ten uzyskuje dostęp do właściwości klasy `config` (wyciąganych z pliku *Web.config*), a następnie nadaje składowej strony definiującej bieżący motyw wartość odpowiadającą aktualnie wybranemu motywowi. Pozwala to zapewnić spójny wygląd całej witryny, a zmiany wyglądu i zachowania witryny można dokonać wprowadzając zaledwie jedną zmianę w pliku *Web.config*! Poniżej pokazujemy, w którym konkretnie miejscu w pliku *Web.config* należy wprowadzić zmianę. Chodzi o sekcję `appSettings`:

```
    <!--
    <add key="CurrentTheme" value="CleanBlue" />
    -->
    <add key="CurrentTheme" value="CleanRed" />
```

W pokazanym fragmencie kodu jeden wpis definiujący motyw jest oznaczony jako komentarz, a drugi nie. Wystarczy zamienić je miejscami, aby zmienić bieżący motyw.

Model bezpieczeństwa

Projekt wykorzystuje mechanizm Forms Authentication i obiekt typu SQL Server Security Provider. Po pierwszym zadeklarowaniu w narzędziu administracyjnym ASP.NET Security Administration tool, że ma być wykorzystywany właśnie ten obiekt, generowana jest nowa baza danych z informacjami systemu bezpieczeństwa. Zostaje ona włączona do projektu i trafiają do niej informacje o wszystkich kontach użytkowników i ustawieniach związanych z bezpieczeństwem. W takim modelu bezpieczeństwa mechanizm Forms Authentication nie jest wykorzystywany bezpośrednio, lecz za pośrednictwem wielu różnych, nowych kontrolek ASP.NET 2.0 zapewniających bezpieczeństwo. Mamy tu na myśli na przykład kontrolki

wykorzystywane do obsługi logowania użytkowników, wyświetlania informacji o stanie zalogowania, przypominania zapomnianych haseł, zmiany haseł czy też tworzenia nowych użytkowników.

Ten model bezpieczeństwa jest wykorzystywany w wielu obszarach aplikacji. Na przykład w referencjach do folderu *Management* witryny. Model bezpieczeństwa pozwala użytkownikowi zalogować się w witrynie i stać się uwierzytelnionym użytkownikiem. Formularz *Login.aspx* jest wczytywany automatycznie za każdym razem, gdy użytkownik próbuje uzyskać dostęp do któregokolwiek pliku ASPX z folderu *Management* bez uprzedniego uwierzytelnienia się. Tak w skrócie wyglądają możliwości oferowane przez nowy model bezpieczeństwa ASP.NET 2.0, zaimplementowany z wykorzystaniem obiektów Role Provider i Membership Provider. Zostało to skonfigurowane tak, aby jedynym sposobem zaimplementowania bezpieczeństwa było skorzystanie z obiektu kontrolki `Login` ASP.NET, tak jak w poniższym przykładzie:

```
    <asp:Login ID="Login1" runat="server" />
```

Stanowi to doskonały przykład zabezpieczenia folderu witryny i uzyskiwania dostępu do zawartych w nim stron w oparciu o role poszczególnych użytkowników, przypisane za pośrednictwem narzędzia ASP.NET 2.0 Configuration Tool. Narzędzie to służy do zarządzania uprawnieniami w systemie zabezpieczeń. Można się do niego dostać z poziomu Visual Studio, wybierając w menu polecenie *Website|ASP.Net Configuration*. Gdy program narzędziowy zostanie w pełni wczytany, na ekranie pojawi się zakładka Security. Kliknięcie jej pozwala modyfikować ustawienia wszystkich folderów wchodzących w skład witryny i zezwalać na dostęp lub ograniczać go użytkownikom, którzy pełnią określone role. Role te definiujemy samodzielnie i przypisujemy je poszczególnym użytkownikom. Na skutek tych działań generowany jest plik *Web.config*, który trafia do folderu, do którego chcemy ograniczyć dostęp. Poniżej pokazujemy przykładową zawartość pliku *Web.config*:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration xmlns="http://schemas.microsoft.com/.NetConfiguration/v2.0">
    <system.web>
        <authorization>
            <deny users="?" />
            <allow roles="Administrator" />
            <allow roles="SuperAdmin" />
        </authorization>
    </system.web>
</configuration>
```

Za ustawienia bezpieczeństwa odpowiadają w tym pliku konfiguracyjnym trzy główne wpisy. Stanowią one serię wyrażeń w formacie XML, definiujących uprawnienia do danego folderu w sposób hierarchiczny w ramach całej witryny. Przesłaniają one ustalenia poczynione w głównym pliku *Web.config* witryny, a także w pliku *machine.config* umieszczonym na serwerze. W pliku tym wyrażenie `<deny users="?" />` oznacza, że do folderu nie powinien zostać wpuszczony żaden nieuwierzytelniony użytkownik (symbolizowany przez znak zapytania). Wyrażenia `<allow roles= Admin" />` i `<allow roles="SuperAdmin" />` oznaczają, że wszyscy użytkownicy, którym przypisano rolę Admin lub Superadmin, będą mieli dostęp do folderu.

Do użytku w ramach mechanizmu obsługi ankiet tworzone są dwa konta i przypisywane są im dwie różne role:

Nazwa użytkownika	Hasło	Opis konta
Admin	password#	Temu użytkownikowi przypisywana jest rola o nazwie Administrator.
SuperAdmin	password#	Temu użytkownikowi przypisywana jest rola Superadministrator.

Wykorzystane tu dwie role są już zdefiniowane w bazie danych systemu bezpieczeństwa i odwołujemy się do nich w różnych obszarach aplikacji, gdy chcemy, aby pewne jej obszary były bardzo dobrze zabezpieczone:

Rola	Opis roli
Administrator	Ta rola pozwala dodawać, edytować i usuwać ankiety oraz ich pytania.
Superadministrator	Ta rola daje takie same przywileje co rola Administrator, a ponadto pozwala na usuwanie ankiet i (lub) ich poszczególnych pytań z systemu.

Tak więc możemy kontrolować zarówno dostęp do elementów formularzy, jak i do folderów, wykorzystując narzędzie ASP.NET Configuration Tool lub własne skrypty w języku VB.NET.

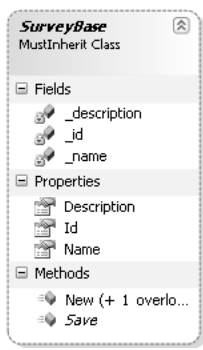
Wykorzystywane klasy

Mechanizm obsługi ankiet wykorzystuje tylko kilka podstawowych klas, przy czym są to klasy inteligentne. Zaprojektowane tak, aby pracowały w sposób „przyjazny obiektom". To znaczy w typowym środowisku obiektowym struktury tych klas funkcjonowałyby lepiej niż inne struktury obiektowe.

Klasa SurveyBase

Klasa `SurveyBase` (patrz rysunek 4.14) pełni rolę klasy bazowej, z której można dziedziczyć i do której odwołuje się każda ankieta. Pozwala ona obiektom klasy potomnej `Survey` udostępniać metody `Save` i `New`, które ułatwiają spójne i wygodne zarządzanie klasą.

Rysunek 4.14



W poniższej tabeli wymieniono metody dostępne w klasie `SurveyBase`:

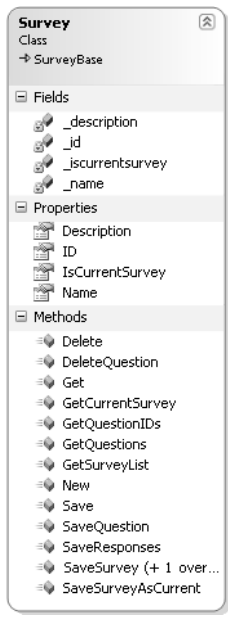
Metoda	Typ zwracanych danych	Opis
`New()`	nie dotyczy	Konstruktor klasy `SurveyBase`.
`Save()`	`Int`	Metoda `Save` wykorzystywana do zapisywania obiektów klasy pochodnej `Survey`.

Klasę `Survey` opisujemy po klasie `SurveyBase`, ponieważ jest to klasa, która dziedziczy z `SurveyBase`. Dzięki temu ma ona dostęp do współużytkowanych metod i funkcjonalności oferowanej przez klasę `SurveyBase`.

Klasa Survey

Klasa `Survey` (patrz rysunek 4.15) odpowiada za większą część operacji dostarczania obiektów do warstwy biznesowej aplikacji. Jej metody są dostępne jako publiczne i współużytkowane, co ułatwia ich stosowanie w różnorodnych formularzach i kontrolkach aplikacji. Oznacza to też, że aby wywoływać te metody, nie trzeba tworzyć obiektów klasy `Survey`. Zamiast tego do wywołania żądanej funkcji w dowolnym formularzu WebFrom zapisanym w języku VB.NET lub w dowolnej kontrolce wystarczy użyć składni `Survey.NazwaMetody()`.

Rysunek 4.15



W poniższej tabeli wymieniliśmy wszystkie dostępne składowe klasy `Survey`:

Metoda	Typ zwracanych danych	Opis
`Delete`	nie dotyczy	Usuwa ankietę z bazy danych, wywołując metodę `Delete()` klasy `SurveyDB`.
`DeleteQuestion`	nie dotyczy	Usuwa pytanie z bazy danych, wywołując metodę `DeleteQuestion()` klasy `SurveyDB`.

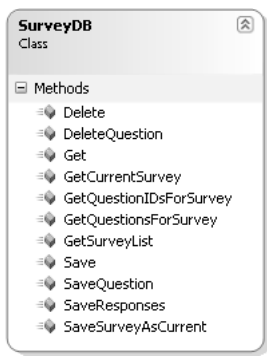
Metoda	Typ zwracanych danych	Opis
`DeleteQuestion`	nie dotyczy	Usuwa pytanie z bazy danych, wywołując metodę `DeleteQuestion()` klasy `SurveyDB`.
`Get`	Obiekt klasy `Survey`	Pobiera ankietę z bazy danych, wywołując metodę `Get()` klasy `SurveyDB`.
`GetCurrentSurvey`	`DataSet`	Zwraca z bazy danych bieżącą ankietę.
`GetQuestionIDs`	`Collection`	Pobiera zbiór identyfikatorów pytań z danej ankiety.
`GetQuestions`	`DataSet`	Pobiera zbiór pytań i możliwych odpowiedzi z danej ankiety.
`GetSurveyList`	`DataSet`	Zwraca z bazy danych listę ankiet z podanej kategorii.
`New`	nie dotyczy	Zapewnia potencjalną możliwość przetwarzania akcji i informacji w chwili tworzenia obiektów.
`Save`	`Integer`	Zapisuje ankietę w bazie danych, wywołując metodę `Save()` klasy `SurveyDB`. Ponieważ klasa ta dziedziczy z klasy `SurveyBase`, metodę `Save` można przeciążać i wykorzystywane jest słowo kluczowe `Me`.
`SaveQuestion`	`Boolean`	Zapisuje zbiór pytań ankiety.
`SaveResponses`	`Boolean`	Zapisuje zbiór odpowiedzi na pytania z danej ankiety.
`SaveSurvey`	nie dotyczy	Zapisuje ankietę w bazie danych.
`SaveSurveyAsCurrent`	nie dotyczy	Zapisuje ankietę jako bieżącą.

Następna klasa reprezentuje te metody aplikacji, które mają bezpośredni związek z danymi.

Klasa SurveyDB

Klasa `SurveyDB` (patrz rysunek 4.16) pełni rolę warstwy danych aplikacji. Zasadniczo jest to główny pośrednik dla wszystkich metod z warstwy biznesowej, które żądają dostępu do bazy danych. Poza klasą `SurveyDB` żadna inna klasa aplikacji ani żaden inny fragment kodu nie przeprowadza bezpośrednich operacji na bazie danych.

Rysunek 4.16



W poniższej tabeli wymieniamy wszystkie dostępne składowe klasy `SurveyDB`:

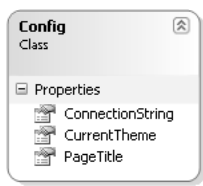
Metoda	Typ zwracanych danych	Opis
`Delete`	nie dotyczy	Usuwa ankietę z bazy danych.
`DeleteQuestion`	nie dotyczy	Usuwa pytanie z bazy danych.
`Get`	`Survey`	Zwraca obiekt klasy, przesyłając identyfikator ankiety.
`GetCurrentSurvey`	`DataSet`	Pobiera z bazy danych bieżącą ankietę.
`GetQuestionIDsForSurvey`	`Collection`	Pobiera z bazy danych kolekcję identyfikatorów pytań.
`GetQuestionsForSurvey`	`DataSet`	Pobiera z bazy danych zbiór (`DataSet`) pytań.
`GetSurveyList`	`DataSet`	Pobiera z bazy danych zbiór (`DataSet`) ankiet.
`Save`	`Integer`	Zapisuje ankietę w bazie danych.
`SaveQuestion`	`Boolean`	Zapisuje w bazie danych pytanie przypisane do ankiety.
`SaveResponses`	`Boolean`	Zapisuje odpowiedź na pytanie występujące w ankiecie.
`SaveSurveyAsCurrent`	nie dotyczy	Czyni wskazaną ankietę ankietą bieżącą.

Następna klasa to klasa konfiguracyjna, z której często korzystamy w tej książce.

Klasa Config

Klasa `Config`, pokazana na rysunku 4.17, pełni rolę menedżera konfiguracji aplikacji. Stanowi ona główny punkt wejścia dla wszystkich ustawień konfiguracyjnych, do których dostępu mogą wymagać poszczególne warstwy aplikacji. Poza klasą `Config` żadna inna klasa ani fragment kodu nie mogą odwoływać się bezpośrednio do danych konfiguracyjnych.

Rysunek 4.17



W poniższej tabeli wymieniliśmy dostępne składowe klasy `Config`:

Właściwość	Typ zwracanych danych	Opis
`ConnectionString`	`String`	Łańcuch połączenia, pochodzący z pliku *Web.config*.
`CurrentTheme`	`String`	Bieżący motyw witryny, zdefiniowany w pliku *Web.config*.

Właściwość	Typ zwracanych danych	Opis
PageTitle	String	Tytuł witryny, zdefiniowany w kodzie HTML i wyświetlany na każdej podstronie, a pochodzący z pliku *Web.config*.

Teraz powinieneś już mieć dobre rozeznanie w klasach, które zostały wykorzystane w aplikacji, i wiedzieć, jak się ich używa. W następnym podrozdziale omówimy szczegółowo logikę biznesową aplikacji i realizowany przez nią przepływ danych.

Kod i jego objaśnienie

Ten podrozdział objaśnia każdy z podstawowych plików z kodem źródłowym mechanizmu obsługi ankiet. Przyjrzymy się po kolei plikom z poszczególnych folderów i powiemy, jak współpracują one ze sobą w ramach całego projektu.

Pliki z folderu głównego

Główny folder aplikacji do obsługi ankiet zawiera wiele istotnych plików, w tym główne strony-powłoki ASPX, a także strony konfiguracyjne i formatujące.

Web.config

Plik *Web.config* przechowuje ważne wpisy konfiguracyjne wykorzystywane w aplikacji. Jeden z nich, o nazwie `SqlServerConnectionString`, kontroluje sposób łączenia się z bazą danych:

```
<connectionStrings>
add name="ConnectionString" connectionString="Data
➥Source=(local)\SqlExpress;AttachDbFilename=|DataDirectory|\SurveyDB.mdf;Integrated
➥Security=True;User Instance=True" providerName="System.Data.SqlClient"/>
</connectionStrings>
```

Plik *Web.config* zawiera też informacje zarządzające ustawieniami serwera SMTP, odpowiedzialnymi za wysyłanie poczty elektronicznej:

```
  <appSettings>
    <add key="EmailFrom" value="admin@mysurveyengine.com" />
    <add key="EmailTo" value="admin@mysurveyengine.Com" />
```

Plik *Web.config* jest wykorzystywany do łatwej zmiany motywów graficznych obowiązujących w całej witrynie. Więcej informacji na ten temat zamieściliśmy w podrozdziale „Motywy i skórki” we wcześniejszej części tego rozdziału.

Survey.vb

Klasa `Survey` stanowi jeden z najważniejszych obszarów naszej aplikacji. Zawiera ona metody i właściwości, które umożliwiają przechowywanie informacji związanych z ankietami, oraz logikę, które pozwala zaimplementować uaktualnianie tych informacji na poziomie warstwy dostępu do danych. Niektóre z metod zapewniają dostęp do ogólnych informacji na temat ankiet, inne zaś pozwalają uzyskać pełny zbiór danych na temat wszystkich ankiet. Ponadto metoda `GetQuestions` zwraca wszystkie pytania z danej ankiety.

Klasa `Survey.vb` może też zostać związana w ramach interfejsu użytkownika z kontrolką `ObjectDataSource`, tym samym tworząc warstwę biznesową aplikacji. Jej metody są zadeklarowane jako publiczne i współużytkowane, co umożliwia przyjęcie metodologii szybkiego tworzenia aplikacji i odwoływania się do składowych klasy `Survey` bez konieczności tworzenia obiektów tej klasy.

Dzięki stosowaniu w pliku klasy `Survey.vb` znaczników `#Region` środowisko programistyczne Visual Studio pozwala nam podzielić stronę na wiele wyodrębnionych sekcji. Do sekcji, które są często wykorzystywane właśnie do grupowania kodu, zaliczamy: `Variables` (zmienne), `Constructors` (konstruktory), `Methods` (metody) i `Properties` (właściwości). Nie ma to żadnego wpływu na kod wynikowy generowany przez środowisko .NET, ale ułatwia zarządzanie logiką aplikacji. Na rysunku 4.18 pokazaliśmy, w jaki sposób tak podzielony kod jest wyświetlany w środowisku Visual Studio.

Rysunek 4.18

```
Imports System.Data

''' <summary>
''' Klasa Survey zapewnia dostęp do informacji
''' zawartych w ankiecie, takich jak jej pytania i tytuł
''' </summary>
Public Class Survey
    Inherits SurveyBase

Zmienne prywatne

Konstruktor(y)

Metody publiczne

Właściwości publiczne

End Class
```

Jedną z ważniejszych metod wywoływanych w odniesieniu do ankiet jest metoda `SaveSurvey`, której kod przedstawia się następująco:

```
Public Shared Sub SaveSurvey(ByVal Name As String, ByVal Description As String, ByVal ID
As Integer)

        Dim mSurvey As New Survey
        mSurvey.ID = ID
        mSurvey.Name = Name
        mSurvey.Description = Description
        SurveyDB.Save(mSurvey)

End Sub
```

Metoda ta zapewnia środki niezbędne do przekazania obiektu klasy `Survey` do warstwy danych celem jego przetworzenia.

Config.vb

Klasa `Config` jest wykorzystywana w charakterze dostępnego obiektu o trzech składowych statycznych. Jej składowe są zadeklarowane jako właściwości w celu uabstrakcyjnienia lokalizacji, w których wartości te są przechowywane. Obecnie te trzy właściwości to: `ConnectionString`, `CurrentTheme` oraz `PageTitle`. Wartości tych właściwości są przechowywane w pliku *Web.config* i wczytywane w razie potrzeby za pośrednictwem klasy `Config`:

```
Imports Microsoft.VisualBasic
Public Class Config
    ''' <summary>
    ''' Łańcuch definiujący połączenie, pochodzący z pliku web.config
    ''' </summary>
    Public Shared ReadOnly Property ConnectionString() As String
        Get
            Return ConfigurationManager.ConnectionStrings("ConnectionString").
            ➥ConnectionString
        End Get
    End Property
    ''' <summary>
    ''' Bieżący motyw witryny, zdefiniowany w pliku web.config
    ''' </summary>
    Public Shared ReadOnly Property CurrentTheme() As String
        Get
            Return ConfigurationManager.AppSettings("CurrentTheme").ToString()
        End Get
    End Property
    ''' <summary>
    ''' Tytuł witryny zdefiniowany w kodzie HTML i wyświetlany na każdej podstronie, a pochodzący
    ''' z pliku web.config
    ''' </summary>
    Public Shared ReadOnly Property PageTitle() As String
        Get
            Return ConfigurationManager.AppSettings("PageTitle").ToString()
        End Get
    End Property
End Class
```

Jak widać na przykładzie klasy `Config`, właściwości `ConnectionString`, `CurrentTheme` i `PageTitle` są oznaczone jako `Public Shared Readonly`, co pozwala uzyskiwać do nich dostęp z dowolnego miejsca w projekcie, stosując notację config-kropka. Na przykład: `config.➥ConnectionString()`. To wywołanie zwróci łańcuch inicjujący połączenie z klasy `Config`, bez uprzedniego tworzenia obiektu tej klasy.

SurveyDB.vb

Ta klasa stanowi warstwę danych aplikacji. Udostępnia ona metody, które umożliwiają pobieranie informacji z bazy danych, a także wstawianie i uaktualnianie informacji w bazie. Jest to jedyna klasa, za pośrednictwem której można się odwoływać do plików bazy danych. Dzięki

temu izolujemy operacje przeprowadzane na danych i umieszczamy je poza warstwą biznesową. Chroni to programistów przed koniecznością powielania kodu realizującego dostęp do danych, pozwala lepiej go zorganizować i umieścić w jednym miejscu. Dzięki takiemu rozwiązaniu aplikacja może też być podzielona na warstwy, co ułatwia migrację bądź rozbudowę aplikacji na kilka serwerów.

Zgodnie z udokumentowanym wywołaniem funkcji z klasy `Survey` klasa `surveyDB` zawiera metodę `Save`, której kod pokazujemy poniżej:

```
Public Shared Function Save(ByVal mSurvey As Survey) As Integer

    Using mConnection As New SqlConnection(Config.ConnectionString)

        Dim mNewSurveyID As Integer
        Dim mCommand As SqlCommand =
        ↪New SqlCommand("sprocSurveyInsertUpdateItem", mConnection)
        mCommand.CommandType = CommandType.StoredProcedure
        If mSurvey.ID > 0 Then
            mCommand.Parameters.AddWithValue("@id", mSurvey.ID)
        Else
            mCommand.Parameters.AddWithValue("@id", DBNull.Value)
        End If
        mCommand.Parameters.AddWithValue("@name", mSurvey.Name)
        mCommand.Parameters.AddWithValue("@description", mSurvey.Description)
        If mSurvey.IsCurrentSurvey = False Then
            mCommand.Parameters.AddWithValue("@iscurrentsurvey", 0)
        Else
            mCommand.Parameters.AddWithValue("@iscurrentsurvey", 1)
        End If

        mConnection.Open()
        mNewSurveyID = mCommand.ExecuteScalar()
        mConnection.Close()

        Return mNewSurveyID

    End Using
End Function
```

Metoda ta przyjmuje parametr typu `Survey` i uzyskuje dostęp do składowych celem zapisania ich w bazie danych. Inną ciekawą metodą jest metoda `GetCurrentSurvey()`, zwracająca obiekt `DataSet` zawierający bieżącą ankietę. Poniżej pokazujemy fragment kodu tej metody:

```
''' <summary>
''' Pobiera z bazy danych 'bieżącą' ankietę
''' </summary>
Public Shared Function GetCurrentSurvey() As DataSet
    Dim dsSurveys As DataSet = New DataSet()
    Try
        Using mConnection As New SqlConnection(Config.ConnectionString)

            Dim mCommand As SqlCommand =
            ↪New SqlCommand("sprocSurveySelectSingleItemWhereCurrent",
            ↪mConnection)
            mCommand.CommandType = CommandType.StoredProcedure
            Dim myDataAdapter As SqlDataAdapter = New SqlDataAdapter()
            myDataAdapter.SelectCommand = mCommand
```

```
                myDataAdapter.Fill(dsSurveys)
                mConnection.Close()
                Return dsSurveys
            End Using
        Catch ex As Exception
            'Wywołując wyrażenie "Throw" przekazujemy błąd do pliku global.asax, który wykorzysta
            'domyślną stronę obsługi błędów do przetworzenia błędu i wyświetlenia użytkownikowi
             'zdefiniowanego przez nas komunikatu o błędzie.
            Throw
        End Try
    End Function
```

Pokazana tu logika aplikacji obejmuje następujące czynności:

1. Utworzenie nowego obiektu `SqlCommand` i przekazanie do niego nazwy procedury przechowywanej na serwerze i połączenia.
2. Określenie rodzaju polecenia jako procedury przechowywanej na serwerze.
3. Utworzenie nowego obiektu `DataAdapter`.
4. Przypisanie składowej `SelectCommand` obiektu `DataAdapter` do nowo utworzonego polecenia.
5. Wywołanie metody `Fill` obiektu `DataAdapter` i przekazanie obiektu `DataSet`, który ma zostać wypełniony danymi.
6. Zamknięcie połączenia.
7. Zwrócenie obiektu `DataSet` metodzie wywołującej.

Warto wspomnieć, że obecnie ankieta jest prezentowana użytkownikom za pośrednictwem internetowej kontrolki `CurrentSurvey`. Chcąc rozszerzyć aplikację, mógłbyś przedstawić listę ankiet, z której można by wybierać żądaną ankietę, lub zaimplementować dynamiczną witrynę, która automatycznie wybierałaby właściwą ankietę do wyświetlenia danemu użytkownikowi.

Formularze WebForm

Formularze WebForm to standardowe strony ASPX opisujące graficzny interfejs użytkownika aplikacji, prezentowany po stronie klienta. Kilka spośród formularzy WebForm występujących w projekcie ma szczególnie istotne znaczenie. Opisujemy je w kolejnych podrozdziałach.

Default.aspx

Plik *Default.aspx* to oczywiście pierwsza strona, która pojawia się po wejściu na witrynę. Umieściliśmy na niej kontrolkę `currentsurvey.ascx` zapewniającą widoczność tytułu i opisu ankiety, która w bazie danych jest oznaczona jako bieżąca. Dzięki temu osoby odwiedzające stronę będą mogły zobaczyć ankietę, wypełnić ją, klikając wybrane odpowiedzi i zobaczyć wyniki.

Login.aspx

Strona logowania zawiera kontrolki `Login` i `PasswordRecovery`. Jak już wspomnieliśmy w innych rozdziałach, są one nowością w środowisku .NET. Strona *Login.aspx* jest umieszczona w głównym folderze witryny i nie wymaga szablonu (ang. *master page*). Kontrolki `Login` zawierają znaczniki języka HTML (pokazane w poniższym kodzie), które definiują konkretne wartości dla strony docelowej i wartości tekstowe kontrolek.

```
    <fieldset style="height: 128px; width: 270px;">
    <asp:Login ID="Login1" runat="server"
    ↪DestinationPageUrl="~/Management/Admin.aspx">
    </asp:Login>
    </fieldset>

    <fieldset style="height: 118px; width: 270px;">
    <asp:PasswordRecovery ID="PasswordRecovery1" runat="server">
    </asp:PasswordRecovery>
    </fieldset>
```

Pokazany kod HTML zawiera definicje kontrolek `Login` i `PasswordRecovery` i ich właściwości.

TakeSurvey.aspx

Formularz *TakeSurvey.aspx* służy do wyświetlania ankiety z bazy danych i zapisywania odpowiedzi w tabeli odpowiedzi. Podstawowe kontrolki wykorzystane w formularzu to: `ObjectDataSource`, `SqlDataSource`, `DataList`, oraz zbiór pól w ramach kontrolki `DataList`, które są związane z właściwościami obiektu. Poniższy fragment kodu ukazuje zdefiniowane wartości kontrolki `ObjectDataSource`, która jest wykorzystywana do związania aplikacji z wartościami pochodzącymi z metody `SelectMethod` przypisanego do niej obiektu biznesowego klasy `Survey`. Metoda `GetQuestions` jest wykorzystywana do pobrania rekordów z tabeli `Survey` w postaci obiektu `DataSet`, z którym jest następnie związywana kontrolka `ObjectDataSource`:

```
<asp:ObjectDataSource ID="odsSurveyQuestions" runat="server"
↪SelectMethod="GetQuestions" TypeName="Survey">
     <SelectParameters>
         <asp:QueryStringParameter Name="id" QueryStringField="surveyID"
         ↪Type="Int32" />
     </SelectParameters>
</asp:ObjectDataSource>
```

Tuż pod tym fragmentem formularza znajduje się kontrolka `DataList`, z którą jest związywana kontrolka `ObjectDataSource`. Pola i ustawienia tej kontrolki istnieją wyłącznie w ramach znaczników HTML, co pokazano poniżej:

```
<asp:DataList ID="DataList1" runat="server" DataSourceID="odsSurveyQuestions">
  <ItemTemplate>
    <%=GetQuestionNum()%>. <%#Server.HtmlEncode(Eval("Text").ToString())%>
      <br />
       <input name="Q<%#Eval("ID")%>" type="radio" value="A">A.
            <%#Server.HtmlEncode(Eval("OptionA").ToString())%></option><br />
       <input name="Q<%#Eval("ID")%>" type="radio" value="B">B.
            <%#Server.HtmlEncode(Eval("OptionB").ToString())%></option><br />
```

```
      <input name="Q<%#Eval("ID")%>" type="radio" value="C">C.
           <%#Server.HtmlEncode(Eval("OptionC").ToString())%></option><br />
      <input name="Q<%#Eval("ID")%>" type="radio" value="D">D.
           <%#Server.HtmlEncode(Eval("OptionD").ToString())%></option><br />
     <br /><br />
  </ItemTemplate>
</asp:DataList>
```

Ten kod określa właściwości obiektu, z którymi związywana jest kontrolka `DataList`, poprzez znaczniki `<%#Eval("ID")%>`. Zapewnia to powtarzającym się wartościom danych w kontrolce `DataList` łączność i związanie z właściwościami obiektu.

Kontrolki użytkownika

Niektóre kontrolki użytkownika ułatwiają nawigację po witrynie i wyświetlanie treści na różnych podstronach. Ponieważ kontrolki interfejsu użytkownika stosowane w projektach internetowych promują tworzenie kodu nadającego się do wielokrotnego użytku, przygotowaliśmy je w taki sposób, aby można je było stosować na wielu stronach witryny.

header.ascx

Kontrolka `header` ma za zadanie wypełnić górną część każdej strony jakąś sensowną treścią. Jeżeli cokolwiek musi trafić na samą górę strony lub w jej pobliże, umieść to w kontrolce `header`, tak aby było to widoczne w całej witrynie.

Poniższy kod przedstawia cały kod źródłowy z pliku *header.ascx*:

```
<%@ Control Language="VB" AutoEventWireup="false" CodeFile="header.ascx.vb"
➥Inherits="Controls header" %>
<div style="text-align: center">
    <table><tr>
        <td><img src="../Images/headerlogo.gif" /></td>
        <td><h1><% Response.Write(Page.Title) %></h1>
        </td>
    </tr></table>
</div>
```

Zauważ, jak wykorzystujemy znaczniki `<%Response.Write(Page.Title)%>` do zapisania w strumieniu tytułu witryny, który ma figurować na każdej stronie, a który pochodzi z pliku *Web.config*.

footer.ascx

Kontrolka `footer` odpowiada za dolną część każdej podstrony opartej na szablonie. To znaczy odwołania do kontrolki `footer` oraz innych kontrolek znajdują się w szablonie. Dzięki temu jest ona obecna na każdej stronie.

Kod tej kontrolki prezentujemy poniżej:

```
<%@ Control Language="VB" AutoEventWireup="false" CodeFile="footer.ascx.vb"
➥Inherits="Controls_footer" %>
<a href="http://wrox.com" target="_blank">&copy; 2005 Wrox Press</a>   
<asp:LoginStatus ID="LoginStatus1" runat="server" LogoutAction="RedirectToLoginPage"
    LogoutPageUrl="~/Login.aspx" />
```

W tym fragmencie pojawiło się odwołanie do kontrolki `LoginStatus`, która jest nowością w ASP.NET 2.0. Wyświetla ona przycisk o zmiennym hiperłączu, który obsługuje logowanie i wylogowywanie użytkowników. Gdy użytkownik jest zalogowany w witrynie, przycisk ten pozwala mu się wylogować. Po kliknięciu przycisku użytkownik zostanie przeniesiony na stronę logowania, na której ta sama kontrolka pozwoli się zalogować.

navigation.ascx

Zadaniem kontrolki `navigation` jest wyświetlanie na każdej stronie tego samego menu. Sama kontrolka menu jest nowością w ASP.NET 2.0 i wiąże się z kontrolką `SiteMapDataSource`. Ta ostatnia również należy do nowych elementów, jakie pojawiły się w środowisku uruchomieniowym .NET 2.0. Służy ona do odwoływania się do plików XML, w których, w poszczególnych wpisach, umieszczone są pliki podstron.

Poniższy fragment kodu przedstawia znaczniki języka HTML obecne w kontrolce `navigation`:

```
<%@ Control Language="VB" AutoEventWireup="false" CodeFile="navigation.ascx.vb"
➥Inherits="Controls_Navigation" %>
<asp:Menu ID="Menu1" runat="server" DataSourceID="SiteMapDataSource1"
➥Orientation="Horizontal"
    StaticDisplayLevels="2"></asp:Menu>
<asp:SiteMapDataSource ID="SiteMapDataSource1" runat="server" />
```

Tu zaś pokazujemy plik XML kontrolki `SiteMapDataSource`:

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
    <siteMapNode url="ContentFiles/default.aspx" title="Strona główna"
    ➥description="">
      <siteMapNode url="ContentFiles/about.aspx" title="O aplikacji"
      ➥description="" />
      <siteMapNode url="ContentFiles/contact.aspx" title="Kontakt"
      ➥description="" />
      <siteMapNode url="Management/admin.aspx" title="Administracja"
      ➥description="" />
    </siteMapNode>
</siteMap>
```

Aby umieścić w menu witryny hiperłącze do kolejnej strony, wystarczy skopiować któryś z wpisów obecnych w pliku *Web.sitemap* i wkleić go do tego samego pliku, wprowadzając niezbędne modyfikacje. W ten oto sposób, stosując jedynie referencję do kontrolki `navigation`, możemy mieć takie samo menu na każdej podstronie.

surveyresults.ascx

Kontrolka `SurveyResults` wyświetla wyniki ankiety wskazywanej przez wartość `QueryString`. Referencja do niej pojawia się na stronie *SurveyResults.aspx* umieszczonej w folderze *ContentFiles*, a także na stronie *MgtSurveyResults.aspx* z folderu *Management*. Kontrolka *SurveyResults* zapewnia możliwość wyświetlenia procentowego ujęcia wyników danej ankiety w rozbiciu na poszczególne pytania.

Poniższy kod HTML pochodzi ze strony *SurveyResults.aspx*, wyświetlającej kontrolkę `SqlDataSource`. Zwróć uwagę na wykorzystanie identyfikatora `surveyID` obiektu `QueryString` w sekcji `<SelectParameters>` pliku:

```
    <asp:SqlDataSource ID="SqlSurveyResults" runat="server" ConnectionString="<%$
    ➥ConnectionStrings:ConnectionString %>" SelectCommand="SELECT * FROM
    ➥[viewAnswerPercentByQuestion] WHERE ([SurveyID] = @SurveyID)">
        <SelectParameters>
            <asp:QueryStringParameter Name="SurveyID" QueryStringField="surveyID"
            ➥Type="Int32" />
        </SelectParameters>
    </asp:SqlDataSource>
```

Następna sekcja pliku zawiera związaną z danymi kontrolkę `DataList` wraz ze wszystkimi potrzebnymi polami wymienionymi wewnątrz znaczników przetwarzanych po stronie serwera, tak jak w poniższym kodzie:

```
<asp:DataList ID="DataList1" runat="server" DataSourceID="SqlSurveyResults">
    <ItemTemplate>
        <%=GetQuestionNum()%>. <%#Server.HtmlEncode(Eval("Text").ToString())%>
       <table border="0" cellpadding="1" cellspacing="0">
        <tr><td>
           <span class="Pct"><%#Server.HtmlEncode(Eval("PctA").ToString())%>%</span>
        </td><td>
           <%#Server.HtmlEncode(Eval("OptionA").ToString())%>
       </td></tr><tr><td>
           <span class="Pct"><%#Server.HtmlEncode(Eval("PctB").ToString())%>%</span>
        </td><td>
                <%#Server.HtmlEncode(Eval("OptionB").ToString())%>
       </td></tr><tr><td>
           <span class="Pct"><%#Server.HtmlEncode(Eval("PctC").ToString())%>%</span>
        </td><td>
                <%#Server.HtmlEncode(Eval("OptionC").ToString())%>
       </td></tr><tr><td>
           <span class="Pct"><%#Server.HtmlEncode(Eval("PctD").ToString())%>%</span>
        </td><td>
                <%#Server.HtmlEncode(Eval("OptionD").ToString())%>
       </td></tr></table>
         <br />
    </ItemTemplate>
</asp:DataList>
```

Powyższy kod HTML zapewnia kontrolki związane z danymi, które pokazują, które pola są wyświetlane w przeglądarce w ramach tabel i pól zdefiniowanych w języku HTML.

currentsurvey.ascx

Kontrolka `CurrentSurvey` dostarcza nazwę i opis ankiety, która w polu `isCurrentSurvey` w tabeli `Survey` ma wartość 1. Wartość ta oznacza, że dany rekord opisuje ankietę, która ma zostać wyświetlona użytkownikowi w trakcie wykonywania aplikacji. Na typowej witrynie użytkownik będzie mógł brać w danej chwili udział tylko w jednym badaniu. Stąd też dajemy administratorowi możliwość wyboru, która ankieta z zestawu ma być ankietą bieżącą.

Górny fragment kodu kontrolki `CurrentSurvey` wygląd tak:

```
<%@ Control Language="VB" AutoEventWireup="false" CodeFile="currentsurvey.ascx.vb"
➥Inherits="Controls_currentsurvey" %>

<asp:ObjectDataSource ID="odsCurrentSurvey" runat="server"
    SelectMethod="GetCurrentSurvey" TypeName="Survey"></asp:ObjectDataSource>
```

Zwróć uwagę na sposób pobrania z bazy danych jednego rekordu z tabeli `Survey`, który w polu `IsCurrentSurvey` ma wartość 1. Wykorzystujemy do tego kontrolkę `ObjectDataSource` i metodę `SelectMethod` z `GetCurrentSurvey`. Kolejny fragment kodu kontrolki `CurrentSurvey`, pokazany poniżej, wyświetla kontrolkę `DataList` po nawiązaniu połączenia z właściwościami kontrolki `ObjectDataSource`. W ten sposób pozyskiwana jest nazwa i opis ankiety oznaczonej w bazie danych jako bieżąca:

```
<asp:DataList ID="DataList1" runat="server" DataSourceID="odsCurrentSurvey">
   <ItemTemplate>
         <fieldset class="CurrentSurveySection">
         <b><a href="TakeSurvey.aspx?surveyID=<%# Eval("ID") %>">
         <%#Server.HtmlEncode(Eval("Name").ToString())%></a></b><br />
         <%#Server.HtmlEncode(Eval("Description").ToString())%> <br />
         <a href="TakeSurvey.aspx?surveyID=<%# Eval("ID") %>">...wypełnij
         ➥ankietę!</a>
         </fieldset>
   </ItemTemplate>
</asp:DataList>
```

Jak już wcześniej wspominaliśmy, istnieją też inne sposoby wyświetlania ankiet, ale nasze podejście jest naprawdę proste i sprawdza się dobrze w sytuacji, gdy nie ma potrzeby prezentowania więcej niż jednej ankiety w tym samym czasie. Wykorzystujemy przy tym jedynie niewielki fragment cennej przestrzeni strony WWW. Kontrolkę tę powinno się dać zmieścić na każdej witrynie, co czyni ją dobrze przystosowaną do przewidzianych zastosowań.

W następnym podrozdziale opisujemy dokładnie proces instalacji i konfiguracji plików źródłowych naszej witryny.

Konfiguracja projektu

Nadszedł czas, aby powiedzieć, jak możesz zainstalować mechanizm obsługi ankiet, i przekonać się na własne oczy, jak szybko i łatwo można uruchomić taką aplikację. Witrynę możesz zainstalować jako kod źródłowy do edycji w Visual Studio 2005 lub VWD.

Instalacja w środowisku programistycznym

W tym podrozdziale zakładamy, że środowisko uruchomieniowe .NET Framework 2.0 zostało już zainstalowane, podobnie jak Visual Studio 2005 lub VWD. Jeżeli chcesz wczytać projekt do Visual Studio lub VWD, wykonaj następujące czynności:

1. Utwórz w środowisku Visual Web Developer lub Visual Studio 2005 nową witrynę.
2. Pobierz z serwera Wydawnictwa Helion archiwum *ftp://ftp.helion.pl/przyklady/*
   ➥*aspngr.zip*, otwórz w nim folder *Rozdzial 04 – Mechanizm obslugi ankiet*
   ➥*Zrodla* i wypakuj jego zawartość do wybranego folderu na twardym dysku.
3. Otwórz Eksploratora Windows i przejdź do folderu zawierającego wypakowane pliki. Następnie ułóż okna Visual Web Developera i Eksploratora Windows w taki sposób, aby były one jednocześnie widoczne na ekranie.
4. Będąc w Eksploratorze Windows, zaznacz wszystkie foldery i pliki składające się na kod źródłowy aplikacji i przeciągnij je z okna Eksploratora do okna *Solution Explorer* środowiska VWD. Jeżeli zostaniesz zapytany, czy chcesz zastąpić istniejące pliki, odpowiedz twierdząco. Po wykonaniu tej operacji w oknie *Solution Explorer* powinny się znaleźć wszystkie pliki niezbędne do uruchomienia projektu.
5. W pliku *Web.config*, w sekcji `appSettings`, zmodyfikuj wartości `EmailTo` i `EmailFrom` (patrz poniższy kod), tak aby definiowały one administracyjne konta e-mail wykorzystywane do wysyłania i odbierania poczty elektronicznej (gdybyś w przyszłości zechciał skorzystać z tej możliwości). W tym miejscu możesz też zmienić właściwość `PageTitle`, w której przechowywany jest napis, jaki ma być wyświetlany na pasku tytułu każdej strony:

```
  <appSettings>
    <add key="EmailFrom" value="admin@mysurveyengine.com" />
    <add key="EmailTo" value="admin@mysurveyengine.Com" />
    <add key="PageTitle" value="Mechanizm obsługi ankiet" />
```

6. Również w pliku *Web.config* możesz zmodyfikować wartość `smtp` w sekcji `mailSettings` (patrz poniższy kod), tak aby odpowiadała ona nazwie serwera poczty wychodzącej, za pośrednictwem którego będzie mogła być wysyłana poczta.

```
<system.net>
    <mailSettings>
      <smtp deliveryMethod="Network">
        <network host="smtp.NazwaTwojegoSerweraPoczty.com" port="25" />
      </smtp>
    </mailSettings>
</system.net>
<system.web>
```

7. Kliknij prawym przyciskiem myszy formularz *Default.aspx* w folderze *ContentFiles* i wybierz opcję *Set as Start Page*. Następnie naciśnij klawisz *F5*, aby uruchomić aplikację w środowisku programistycznym. Aby przetestować działanie sekcji administracyjnej, kliknij hiperłącze *Administracja* i zaloguj się, wprowadzając nazwę użytkownika `SuperAdmin` i hasło `password#`.

Podsumowanie

W tym rozdziale przyjrzeliśmy się paru interesującym kontrolkom, jakie pojawiły się w ASP.NET 2.0, takim jak `Login`, `LoginStatus`, `PasswordRetrieval`, `SiteMap`, `SqlDataSource`, `ObjectDataSource`, `SiteMapDataSource`, `GridView`, `DataList` i `Menu`. Poruszyliśmy też zagadnienie szablonów stron. W rozdziale zamieściliśmy dużo ilustracji i przykładów, aby ułatwić Czytelnikom śledzenie logiki aplikacji. Pokazaliśmy, jak nowe i ekscytujące elementy ASP.NET 2.0 umożliwiają błyskawiczne tworzenie aplikacji w środowisku Visual Studio, na co wszyscy czekali.

W rozdziale skupiliśmy się na tym, abyś zrozumiał, w jaki sposób ankiety są wypełniane i pobierane. Położyliśmy też nacisk na warstwowy charakter aplikacji. Dużo miejsca poświęciliśmy porównaniu kontrolki `ObjectDataSource` z kontrolką `SqlDataSource`. Wymieniliśmy zalety i wady obu tych rozwiązań i ostatecznie doszliśmy do wniosku, że preferujemy kontrolkę `ObjectDataSource`.