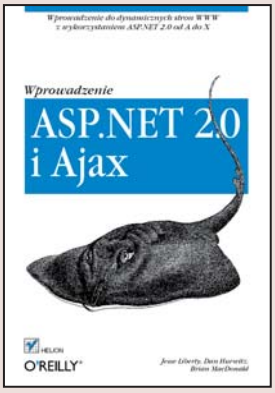
ASP.NET 2.0 i Ajax. Wprowadzenie



Autor: Jesse Liberty, Dan Hurwitz, Brian MacDonald
Tłumaczenie: Daniel Kaczmarek
ISBN: 978-83-246-1510-0
Tytuł oryginału: Learning ASP.NET 2.0 with AJAX: A Practical Hands-on Guide
Format: B5, stron: 488

- Jak stworzyć stronę WWW w ASP.NET 2.0?
- Jak skorzystać z dobrodziejstw technologii AJAX?
- Jak zapewnić bezpieczeństwo, poprawność i trwałość danych w ASP.NET 2.0?

ASP.NET jest technologią stworzoną przez firmę Microsoft, pozwalającą na tworzenie dynamicznych stron WWW. Dostarcza ona wielu kontrolek, które wspomagają programistę w codziennych działaniach i pozwalają na znaczące przyspieszenie prac nad projektowaną witryną. Wykorzystanie technologii AJAX ułatwia tworzenie stron dostarczających niezapomnianych wrażeń ich użytkownikom. ASP.NET wraz z technologią AJAX w połączeniu z narzędziami programistycznymi firmy Microsoft stanowią wiodące rozwiązanie na rynku aplikacji internetowych, a wsparcie potentata z Redmond gwarantuje stały rozwój tej platformy.

„ASP.NET 2.0 i Ajax. Wprowadzenie" odkrywa tajniki wykorzystania technologii AJAX oraz ASP.NET. Dzięki tej książce dowiesz się, jak tworzyć interaktywne i wydajne aplikacje WWW z wykorzystaniem języka VisualBasic 2005. Jednocześnie nauczysz się zapewniać bezpieczeństwo w tworzonych aplikacjach i korzystać z poszczególnych kontrolek. Autorzy książki pokażą Ci metody radzenia sobie z błędami, usterkami i wyjątkami występującymi w trakcie tworzenia i używania strony. Na koniec każdego rozdziału Jesse, Dan i Brian sprawdzą Twoją wiedzę w krótkim quizie – ale nie martw się, na końcu książki znajdziesz odpowiedzi na poszczególne pytania.

- Podstawy tworzenia stron WWW
- Założenia i wykorzystanie technologii AJAX
- Sposób zapisywania i odczytywania danych oraz kontrolki wspierające ten proces
- Zapewnianie poprawności danych
- Definiowanie i wykorzystanie arkuszy stylów CSS
- Wykorzystanie stron wzorcowych
- Obsługa i zasady nawigacji na stronie
- Zapewnienie bezpieczeństwa witrynie
- Wykrywanie i rozwiązywanie problemów w kodzie
- Obsługa wyjątków

Wejdź w świat dynamicznych stron WWW – wykorzystaj najnowsze technologie ASP.NET!

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

Spis treści

Ściąga treści

Ściąga z VB

Ściąga z SQL

ROZDZIAŁ 3.

Sprytne witryny WWW z technologią Ajax

Technologia Ajax zrewolucjonizowała ASP.NET, a od momentu jej udostępnienia praktycznie wszystkie aplikacje ASP.NET zawierają w sobie kontrolki Ajax. Dzięki technologii Ajax aplikacje ASP.NET, które wcześniej w 99 procentach przypadków zawierały jedynie kod wykonywany na serwerze, mogły wreszcie wykonywać większość przetwarzania po stronie klienta, którym zazwyczaj jest przeglądarka użytkownika. W ten sposób znacząco zwiększyła się zarówno rzeczywista, jak i odczuwana przez użytkownika wydajność aplikacji ASP.NET.

Aby zademonstrować dynamizm i elastyczność technologii Ajax, od nowa utworzymy formularz zamówień przedstawiony w rozdziale 2., wykorzystując w nim kontrolki Ajax. Witrynę rozszerzymy o wyświetlanie w polach, które służą do wprowadzania danych, tak zwanych znaków wodnych. Znak wodny to fragment tekstu, który jest wyświetlany w polu tekstowym, lecz znika w momencie, gdy użytkownik rozpocznie wpisywanie danych. Jest to zatem elegancki znak zachęty dla użytkownika. Utworzymy również panel pojawiający się w odpowiednich okolicznościach, dzięki któremu będzie można ukrywać kontrolki do czasu, aż użytkownik rzeczywiście będzie ich potrzebował. Dodane zostanie również rozwijane pole tekstowe, którego zadaniem będzie wyświetlanie informacji o produkcie w sposób pozwalający zaoszczędzić miejsce na stronie.

Krótka wizyta po stronie klienta

Aplikacje internetowe działające po stronie klienta mają niewątpliwie szereg zalet, lecz posiadają także jedną niezaprzeczalną wadę — za każdym razem, gdy trzeba wykonać jakiś kod (albo odczytać dowolne dane), trzeba ponieść koszt „podróży w dwie strony" z przeglądarki do serwera i z powrotem, a sama strona musi zostać wyświetlona całkowicie od nowa. Takie odwołania do serwera i z powrotem mogą zająć stosunkowo dużo czasu (mimo że prędkość przesyłu danych w internecie stale się zwiększa), zaś konieczność narysowania strony na nowo powoduje, że na chwilę znika ona z oczu użytkownika.

Technologia Ajax (jej nazwę powinno się tak naprawdę pisać jako AJX, choć byłoby to znacznie trudniejsze do wymówienia) to akronim słów *Asynchronous JavaScript and XML*. Jest to technika łączenia ze sobą klasycznych (niektórzy powiedzieliby „starych") technologii dla sieci internet na nowe sposoby, dzięki którym możliwe jest zdecydowane zwiększenie wydajności aplikacji

internetowych. Aplikacje, w których używa się technologii Ajax, są obecnie bardzo popularne, ponieważ pod względem wydajności biją one na głowę aplikacje działające wyłącznie na serwerze.

Ajax nie istnieje

Tak naprawdę nie istnieje coś takiego jak Ajax. Nie jest to produkt ani standard, trudno to nawet nazwać technologią. Ajax to jedynie pojęcie określające zbiór już istniejących technologii, dla których wymyślono nowe zastosowania, osiągając w ten sposób znacznie ciekawsze rezultaty.

Nazwa Ajax jako skrót sformułowania „Asynchronous JavaScript and XML" została użyta po raz pierwszy przez Jamesa Garretta w lutym 2005 roku. Garrett wymyślił to pojęcie pod prysznicem (jeśli kogoś to interesuje), rozmyślając w tym samym czasie o potrzebie ukucia skrótowej nazwy dla zestawu technologii proponowanych klientowi (którego, według gorących zapewnień Jamesa, wtedy pod prysznicem nie było).

Z drugiej strony, pojęcie to zostało użyte po raz pierwszy około 3000 lat temu przez Homera, który pisał o wojowniku imieniem Ajaks Wielki (a także o Ajaksie Małym) w „Iliadzie" (księga siódma, wiersze 181 – 312). Ajaks Wielki był najwyższym i najsilniejszym wojownikiem Achajów, a pod względem umiejętności i męstwa przewyższał go jedynie Achilles. Nie jest wcale pewne, czy historia technologii Ajax również będzie opiewana za 3000 lat (ani nawet za 3000 dni); trudno jednak oprzeć się wrażeniu, że *istnieje* silna analogia między wojną trojańską a wojnami środowisk graficznych dla systemów Linux, ale to już historia na inną książkę.

Według Garretta „Ajax […] to tak naprawdę zbiór technologii, z których każda posiada własne mechanizmy, a w połączeniu z innymi pozwala osiągać nowe, niespotykane dotąd efekty". Na Ajax składają się:

- *Standardowe środki prezentacji* oparte na technologiach XHTML i CSS, w których mechanizmy dynamicznej interakcji i wyświetlania zawartości korzystają z modelu DOM (ang. *Document Object Model*). Dzięki temu Ajax może bezpośrednio manipulować dowolnym elementem strony za pośrednictwem języka JavaScript.
- Środki wymiany i manipulowania danymi przy użyciu technologii XML i XSLT, czyli otwartych i niezależnych od platformy sposobów przetwarzania danych, dzięki którym Ajax może działać na dowolnej platformie, wykorzystując technologie powszechnie uznawane za standardowe.
- Asynchroniczne operacje wywoływania i odczytywania danych przy użyciu obiektu `XMLHttp` `↪Request`, który służy do wywoływania jednostkowych danych zawierających tylko część strony. Takie rozwiązanie charakteryzuje się dwiema bardzo ważnymi zaletami: znacznie zmniejsza się ilość danych, jakie trzeba przesyłać za pośrednictwem sieci, a przeglądarka nadal może kontynuować przetwarzanie innych fragmentów strony, czekając jednocześnie na odpowiedź z serwera.
- Silny nacisk na przetwarzanie po stronie klienta, którego celem jest wyeliminowanie jak największej liczby odwołań do serwera i z powrotem i zwiększenie dzięki temu wydajności aplikacji.
- Język JavaScript, który spaja ze sobą wszystkie pozostałe technologie. Ajax korzysta ze standardowego języka skryptowego, którego obsługa jest zaimplementowana w praktycznie wszystkich przeglądarkach internetowych.

Decydenci z firmy Microsoft szybko zdali sobie sprawę, że Ajax jest technologią, której nie można zignorować. Jednocześnie już wcześniej dostali nauczkę, że otwarte standardy muszą nadal pozostać otwarte, dlatego zdecydowali się *jeszcze bardziej* ulepszyć tę technologię, jednocześnie pozostawiając ją jako otwartą.

Cel ten został osiągnięty dzięki połączeniu możliwości, szybkości i elastyczności technologii Ajax z prostotą operacji „przeciągnij i upuść" powszechnie stosowanych w ASP.NET. Microsoft opracował bibliotekę kontrolek Ajax, z których korzysta się równie prosto jak z kontrolek serwerowych ASP.NET, używanych chyba już od czasów średniowiecza. Jednak co jeszcze ważniejsze, projektanci z firmy Microsoft opracowali także stosunkowo prosty sposób tworzenia własnych kontrolek Ajax, które również można przeciągać i upuszczać.

Wszystko to oznacza, że pracę z kontrolkami Ajax udostępnianymi przez Microsoft można zacząć od razu, bez konieczności uczenia się języków JavaScript ani DHTML. Okazuje się więc, że właściwie nie ma powodu, aby wzbraniać się przed używaniem technologii Ajax we wszystkich tworzonych aplikacjach ASP.NET.

Czytelnicy, którzy *uwielbiają* język JavaScript i *chcą* tworzyć własne kontrolki Ajax, nie mają powodów do obaw — jest to nadal możliwe. Podobnie jak w przypadku kontrolek opracowanych samodzielnie, funkcje kontrolek Ajax można rozszerzać, a nawet implementować ich zupełnie nowe zastosowania.

Obecnie można więc zjeść ciastko i mieć ciastko. Nadal można tworzyć aplikacje ASP.NET w tym samym środowisku IDE, co dotychczas, a jednocześnie dodawać do nich skrypty wykonywane po stronie klienta, realizujące asynchroniczne wywołania zwrotne (a zwłaszcza odczytujące dane w sposób asynchroniczny!). Ponadto można do tego celu wykorzystywać bibliotekę gruntownie przetestowanych, gotowych do użycia kontrolek zawierających pełny wymagany kod JavaScript.

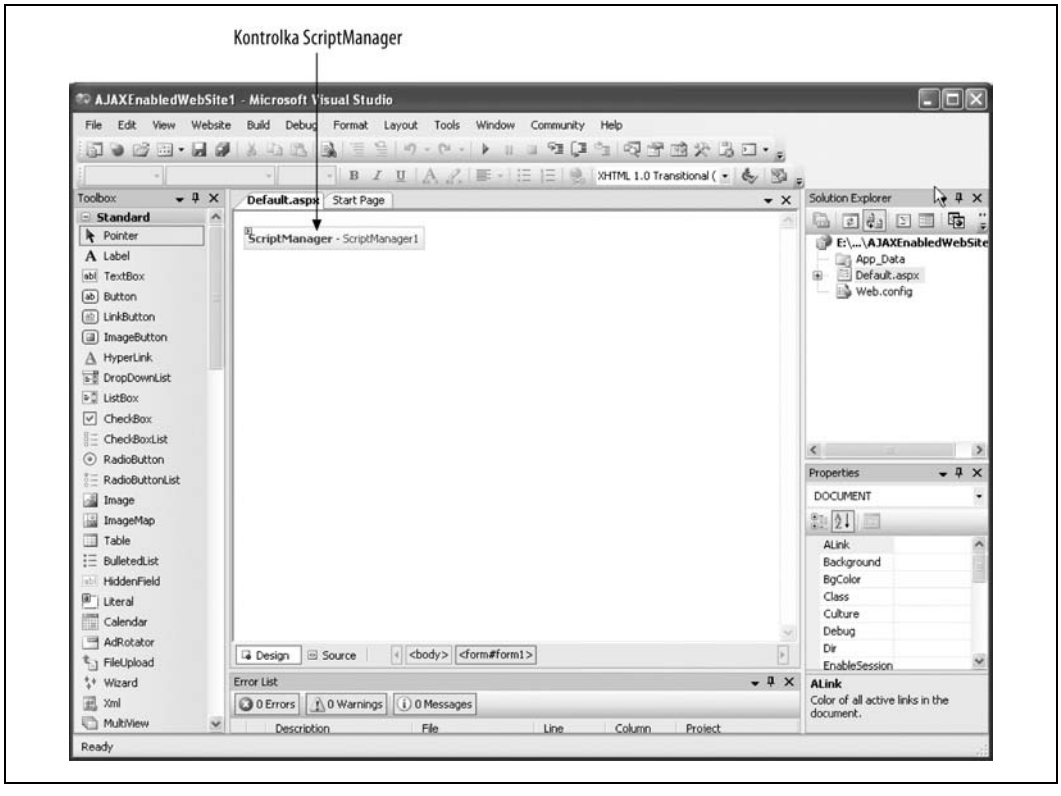
Jednak najważniejszą informacją jest to, że uaktualnienia asynchroniczne polepszają zarówno wydajność aplikacji, jak i sposób, w jaki tę wydajność odczuwa użytkownik. Efekt taki osiągany jest dzięki temu, że strona nie jest przesyłana z powrotem do serwera, lecz dane są odczytywane niezależnie od procesu odtwarzania wyglądu strony. Dzięki temu strona nie migoce, a operacja odczytu danych jest realizowana znacznie szybciej.

Kontrolka ScriptManager

W firmie Microsoft szybko zdano sobie sprawę, że integrowanie standardowych kontrolek ASP.NET i stron z kontrolkami Ajax (zawierających kody JavaScript i DHTML) byłoby zadaniem skomplikowanym, nużącym i wykonywanym wielokrotnie. Utworzono więc kontrolkę `ScriptManager`, dzięki czemu programiści mają dostęp do w pełni przetestowanego, niezawodnego narzędzia, którego zadaniem jest zarządzanie wykonaniem „brudnej" roboty. Dodanie kontrolki `ScriptManager` do strony (środowisko IDE dodaje ją automatycznie w momencie utworzenia projektu wykorzystującego technologię Ajax) rozwiązuje problem integracji kontrolek ASP.NET z kontrolkami Ajax, a obecność kontrolki na stronie, która z niej nie korzysta, nie stanowi tak naprawdę żadnego dodatkowego obciążenia. Poniżej przedstawiono deklarację, która musi pojawić się na każdej stronie i którą automatycznie umieszcza środowisko IDE:

```
<asp:ScriptManager ID="ScriptManager1" runat="server" />
```

Kontrolka `ScriptManager` jest widoczna w widoku projektowania *Design*, co widać na rysunku 3.1, natomiast po uruchomieniu witryny nie będzie ona widoczna dla użytkownika.



Rysunek 3.1. Kontrolka ScriptManager jest widoczna na stronie w widoku Design, natomiast w przeglądarce będzie ona niewidoczna dla użytkownika

Implementacja mechanizmu częściowych uaktualnień strony przy użyciu technologii ASP.NET i Ajax przebiega nadspodziewanie łatwo. Wystarczy jedynie pozostawić domyślną wartość `True` właściwości `EnablePartialRendering` kontrolki `ScriptManager`.

Po wykonaniu jakże skomplikowanego zadania polegającego na tym, by *nie* zmieniać wartości właściwości `EnablePartialRendering`, na stronę można przeciągnąć jedną lub więcej kontrolek `UpdatePanel`. Każdy panel `UpdatePanel` jest uaktualniany oddzielnie i w sposób asynchroniczny, bez wpływu na inne panele ani na inne kontrolki znajdujące się na stronie.

I to wszystko. Niemal bez wysiłku osiągnęliśmy natychmiastowy wzrost wydajności naszej aplikacji.

Aby przekonać się, jak imponujący efekt można w ten sposób osiągnąć, zmodyfikujemy projekt *AdventureWorks* z poprzedniego rozdziału — wstawimy do niego kontrolki `UpdatePanel`, aby zwiększyć wydajność działania aplikacji. Jak pamiętamy, w przykładzie tym (widocznym na rysunku 2.9) utworzono parę przycisków opcji (które wtedy nie zostały jednak udostępnione), których zadaniem było sterowanie widocznością kontrolki `Panel` służącej do gromadzenia informacji osobistych podawanych przez użytkownika. Teraz udostępnimy właśnie ten mechanizm.

Zazwyczaj kliknięcie przycisku opcji powoduje wykonanie wywołania zwrotnego do serwera, a to z kolei doprowadziłoby do chwilowego zniknięcia strony i jej ponownego wyświetlenia. Jednak dzięki kontrolce Ajax `UpdatePanel` uaktualnienie strony zostanie wykonane w sposób asynchroniczny, a więc strona będzie cały czas widoczna dla użytkownika.

Najpierw należy utworzyć kopię Formularza Zamówień Firmy AdventureWorks. Niech nowy projekt nosi nazwę *AdventureWorksRevisited*. Należy go od razu uruchomić, aby przekonać się, że działa poprawnie.

Sposób kopiowania witryn internetowych opisano w dodatku A.

Następnie trzeba usunąć wszystkie kontrolki znajdujące się pod panelem `pnlPersonalInfo` (wszystkie elementy znajdujące się pod kontrolką `Panel` z wyjątkiem znacznika zamykającego formularz).

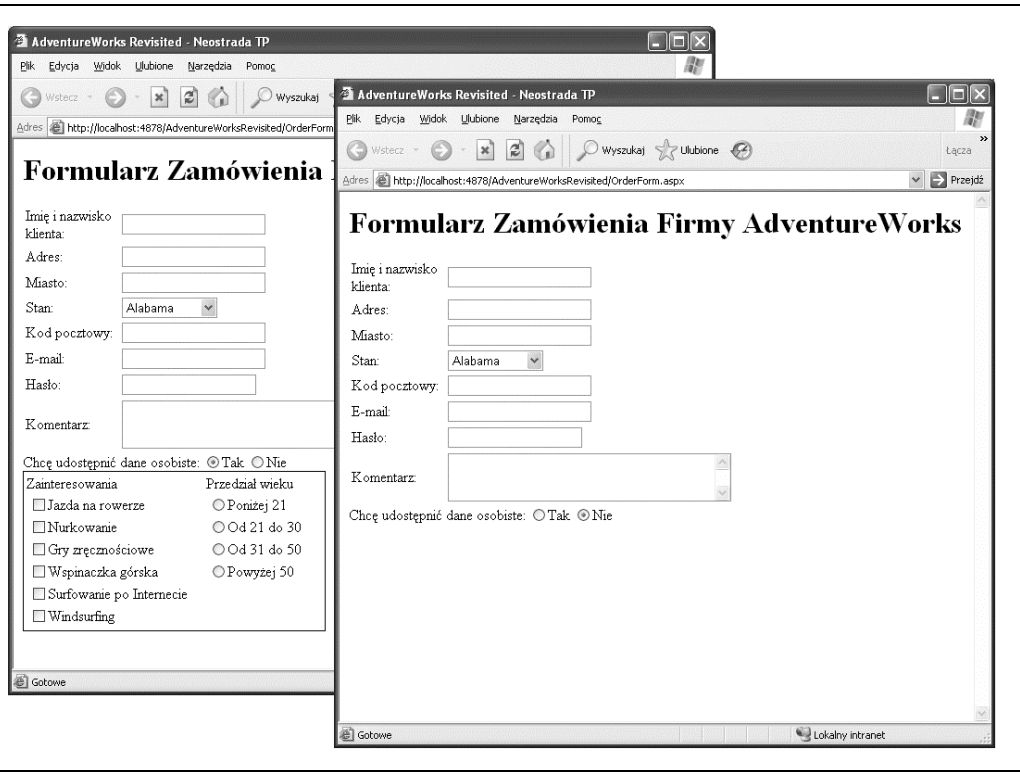
Konieczne będzie również usunięcie z pliku z kodem źródłowym niepotrzebnych już procedur obsługi zdarzenia. (Bez trudu można zidentyfikować te procedury, które nie są już potrzebne — wystarczy uruchomić witrynę i sprawdzić zgłoszone błędy kompilacji). Jak się okazuje, niepotrzebne są wszystkie procedury obsługi zdarzeń zaimplementowane w poprzednim przykładzie.

Teraz na stronie trzeba umieścić kontrolkę Ajax `UpdatePanel`, która otoczy przyciski opcji. Aplikację w finalnej postaci przedstawiono na rysunku 3.2 — widać na nim stronę z panelem — zarówno widocznym, jak i ukrytym.

Niech przyciski opcji *Tak* i *Nie* zaczną w końcu wykonywać jakieś działania. Każdą z kontrolek `RadioButton` należy zaznaczyć i sprawdzić zawartość okna właściwości *Properties*. Najpierw trzeba się upewnić, że właściwość `Text` każdej kontrolki ma właściwą wartość: `Tak` oraz `Nie`. Następnie właściwości `AutoPostBack` obydwu kontrolek należy przypisać wartość `True`. Jak wspomniano już w rozdziale 2., gdy `AutoPostBack` będzie mieć wartość `True`, wówczas natychmiast po jego kliknięciu przycisk opcji zainicjuje wywołanie zwrotne do serwera, co z kolei spowoduje wykonanie zaimplementowanych czynności.

Jak pokazano już w rozdziale 2., czynności przeznaczone do wykonania definiuje się w ramach procedur obsługi zdarzeń. W widoku projektowania *Design* należy utworzyć procedurę obsługi zdarzenia dla przycisku opcji *Tak* — w tym celu trzeba kliknąć myszą przycisk opcji `rbYes`. W oknie właściwości *Properties* należy kliknąć przycisk z błyskawicą, aby przełączyć się z widoku właściwości do widoku zdarzeń, w którym jednym z dostępnych zdarzeń będzie `CheckedChanged`.

W polu po prawej stronie nazwy zdarzenia należy wpisać tekst `YesNoEventHandler`, następnie nacisnąć klawisz tabulacji (albo *Enter*), aby przejść do pliku z kodem źródłowym, w którym kursor zostanie od razu umieszczony w szablonie nowej procedury obsługi zdarzenia i będzie można przystąpić do implementacji własnego kodu źródłowego. Jednak zanim wpiszemy odpowiednie polecenia, należy powrócić do widoku *Design* i kliknąć przycisk opcji `rbNo`. Po raz wtóry trzeba kliknąć myszą pole po prawej stronie zdarzenia `CheckedChanged`. Tym razem w polu tym pojawi się przycisk strzałki skierowanej w dół. Jego kliknięcie umożliwi wybranie z listy już istniejącej procedury obsługi zdarzenia, co widać na rysunku 3.3.



Rysunek 3.2. Witryna WWW AdventureWorkRevisited z ukrytą i widoczną kontrolką Panel. Wkrótce będzie się można przekonać, że wersja witryny korzystająca z technologii Ajax działa znacznie efektowniej niż wersja opracowana w rozdziale 2

Rysunek 3.3. Kliknięcie strzałki skierowanej w dół znajdującej się obok zdarzenia CheckedChanged kontrolki umożliwi podłączenie kontrolki do procedury obsługi zdarzenia, która została utworzona już wcześniej

Na liście rozwijanej należy wskazać procedurę `YesNoEventHandler`. Ponownie IDE przeniesie użytkownika bezpośrednio do kodu źródłowego procedury obsługi zdarzenia znajdującego się w pliku z kodem źródłowym. Warto zwrócić uwagę, że tym razem instrukcja `Handles` wskazuje, iż procedura obsługuje zdarzenie `CheckedChanged` już dla dwóch przycisków opcji. W procedurze należy dopisać kod źródłowy wyróżniony na poniższym listingu pogrubioną czcionką:

```
Protected Sub YesNoEventHandler(ByVal sender As Object, _
   ByVal e As System.EventArgs) _
   Handles rbNo.CheckedChanged, rbYes.CheckedChanged
       pnlPersonalInfo.Visible = rbYes.Checked
End Sub
```

Program można już uruchomić i kliknąć najpierw przycisk opcji *Tak*, a następnie przycisk *Nie*. Po kliknięciu *Tak* panel powinien zostać wyświetlony na stronie, natomiast zaznaczenie opcji *Nie* powinno doprowadzić do zniknięcia panelu. Jak pokazano w poprzednim rozdziale, atrybut `Visible` wskazuje, czy panel ma być wyświetlany, a w powyższym kodzie źródłowym wartość właściwości `Visible` powiązano z wartością kontrolki `rbYes`.

Jednak po kliknięciu dowolnego przycisku opcji da się zapewne zauważyć krótkie mignięcie strony albo jej poruszenie w momencie, gdy cała strona zostanie odświeżona. Czasami odświeżenie strony może nawet nastąpić z dość znacznym opóźnieniem. Przyczyną jest fakt, że po każdym kliknięciu przycisku opcji cała strona zostaje odesłana do serwera w celu jej przetworzenia, na serwerze wykonywana jest procedura obsługi zdarzenia, po czym strona znowu w całości jest przesyłana z powrotem do przeglądarki.

Ajax rozwiązuje problem migotania w taki sposób, że odświeżane są jedynie wybrane części strony, i to w trybie asynchronicznym. Dzięki odświeżaniu segmentów strony „w tle", unika się konieczności wysyłania całej strony do serwera oraz ponownego jej generowania całkowicie od nowa.

W przykładowej aplikacji *AdventureWorks* — w takiej postaci, w jakiej znajduje się ona obecnie, wywołanie zwrotne powoduje również utratę aktualnej pozycji użytkownika na stronie. Gdy strona jest generowana od nowa, przeglądarka wyświetla ją od początkowego elementu, co może być bardzo irytujące.

Niedogodności tej można uniknąć przez przypisanie właściwości `MaintainScroll` ↪`PositionOnPostBack` dyrektywy `Page` wartości `True`. W tym celu w widoku *Source* należy otworzyć plik z kodem znaczników i zmienić definicję dyrektywy `Page` znajdującej się na początku strony:

```
<%@ Page Language="VB" AutoEventWireup="true"
   CodeFile="OrderForm.aspx.vb" Inherits="_Default"
   MaintainScrollPositionOnPostBack="true" %>
```

W widoku Design należy zaznaczyć kontrolkę `ScriptManager` i sprawdzić zawartość okna *Properties*. Jeśli okno właściwości nie jest widoczne, należy kliknąć kontrolkę prawym przyciskiem myszy i wybrać polecenie *Properties*. W oknie właściwości trzeba się upewnić, że właściwość `EnablePartialRendering` ma wartość `True`.

Naszym celem jest doprowadzenie do tego, by w momencie, gdy użytkownik kliknie jeden z przycisków opcji, uaktualniany był wyłącznie sam panel `pnlPersonalInfo`. Aby to osiągnąć, trzeba umieścić `Panel` i przyciski opcji wewnątrz kontrolki `UpdatePanel`, dostępnej w sekcji *Ajax Extensions* okna narzędziowego *Toolbox*.

W oknie projektowania *Design* należy otworzyć zakładkę *Ajax Extensions* okna narzędziowego *Toolbox* i przeciągnąć z niej na formularz kontrolkę `UpdatePanel` (tę samą operację można oczywiście wykonać w widoku *Source*). Następnie trzeba zaznaczyć tekst pytania o udostępnienie danych osobistych, przyciski opcji i panel `pnlPersonalInfo`, po czym wszystkie te elementy przeciągnąć do kontrolki `UpdatePanel`. I to właściwie już wszystko.

Wykonanie tych samych czynności w widoku *Source* przebiega bardzo podobnie: najpierw z okna *Toolbox* trzeba przeciągnąć do okna kontrolkę `UpdatePanel`, a następnie przenieść odpowiednie znaczniki do jej wnętrza.

Program należy ponownie uruchomić i kliknąć najpierw jeden, a potem drugi przycisk opcji — różnica będzie widoczna gołym okiem. Tym razem strona nie powinna już migotać, ponieważ odświeżany jest wyłącznie panel. Możemy więc już z dumą powiedzieć: „Nieźle!”.

Rozszerzone kontrolki dostępne w Control Toolkit

Pakiet narzędziowy Ajax Control Toolkit udostępnia szereg dodatkowych kontrolek wykorzystujących technologię Ajax, za pomocą których można rozszerzyć możliwości tworzonej aplikacji internetowej. W tabeli 3.1 opisano wybrane, najbardziej użyteczne kontrolki dostępne w pakiecie Ajax Control Toolkit.

Kontrolka TextBoxWatermarkExtender

Wiele kontrolek dostępnych w pakiecie narzędziowym ma w nazwie słowo „extender” — oznacza to, że nie działają one samodzielnie, ale rozszerzają działanie jednej ze standardowych kontrolek. Na przykład kontrolka `TextBoxWatermarkExtender` współpracuje z kontrolką `TextBox` i dodaje do niej efekt znaku wodnego. Kontrolka rozszerzająca posiada właściwości, za pomocą których można wskazać modyfikowane przez nią pole tekstowe `TextBox`, zdefiniować tekst wyświetlany jako znak wodny, a także styl, jaki należy zastosować względem znaku wodnego. Na rysunkach 3.4 i 3.5 zaprezentowano działanie znaków wodnych.

Rysunek 3.4. Wygląd kontrolki ze znakiem wodnym, zanim użytkownik zacznie wpisywać jakiekolwiek dane. Znak wodny przypomina użytkownikowi, co należy wpisać w polu tekstowym, a także upewnia go, że w danym momencie pole tekstowe nie zawiera żadnej wartości

Rysunek 3.5. Gdy użytkownik zaczyna wpisywać dane w polu tekstowym, znak wodny znika i wobec danych stosowany jest styl wyraźnie odróżniający je od znaku wodnego

Tabela 3.1. Wybrane kontrolki z pakietu narzędziowego Ajax Control Toolkit

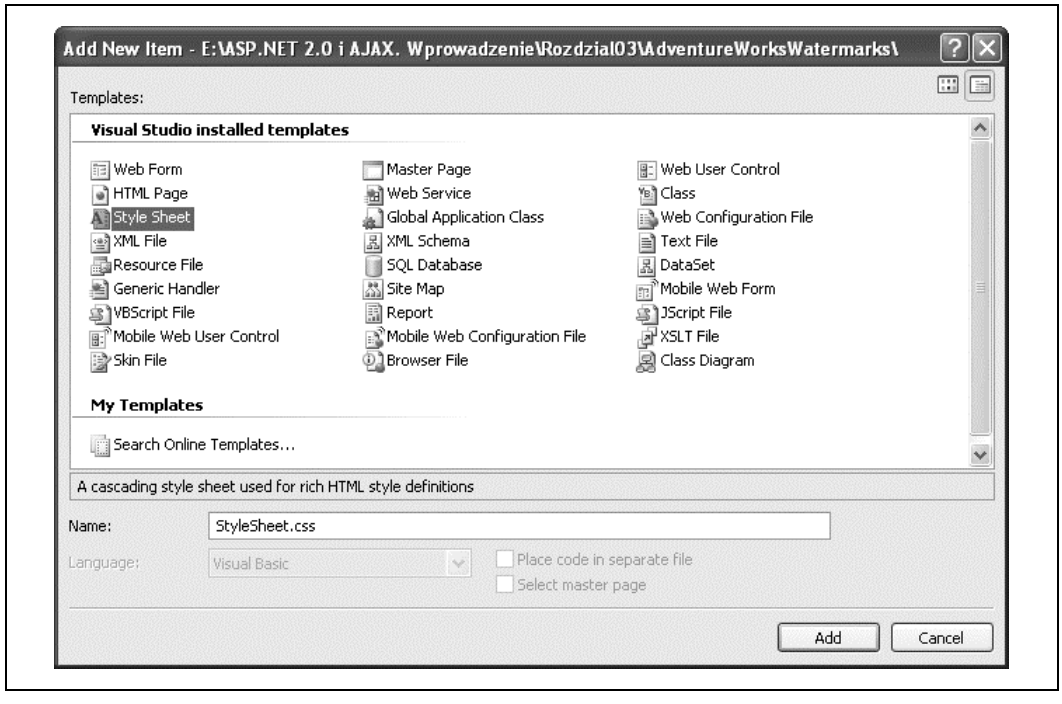
Kontrolka	Opis
`Accordion`	Kontrolka udostępnia okna zadaniowe, z których w danym momencie widoczne jest tylko jedno.
`AlwaysVisibleControlExtender`	Kontrolka pozostaje widoczna również w trakcie przewijania strony przez użytkownika.
`AnimationExtender`	Kontrolka wspiera animowanie panelu lub innej kontrolki znajdującej się na stronie.
`CascadingDropDown`	Opcja wybrana przez użytkownika w jednej kontrolce rozwijanej determinuje zakres wartości dostępnych w drugiej kontrolce.
`CollapsiblePanelExtender`	Umożliwia zwijanie i rozwijanie kontrolki `Panel`.
`ConfirmButtonExtender`	Gdy użytkownik kliknie przycisk, wyświetlane jest okno dialogowe z prośbą o potwierdzenie dokonanego wyboru.
`DragPanelExtender`	Pozwala użytkownikowi na przeciąganie panelu w dowolne miejsce strony.
`FilteredTextBoxExtender`	Zapewnia, że w polu tekstowym zostanie wpisany jedynie „prawidłowy" tekst.
`HoverMenuExtender`	Wyświetla menu, gdy nad kontrolką umieszczony zostanie kursor myszy.
`MutuallyExclusiveCheckBoxExtender`	Pozwala na wybranie jednego lub żadnego spośród pól wyboru. Kontrolka działa podobnie do przycisków opcji, z tą różnicą że użytkownik może również odznaczyć wszystkie dostępne opcje.
`NoBot`	Kontrolka, której zadaniem jest zablokowanie spamu i działalności robotów w witrynie internetowej.
`NumericUpDownExtender`	Dodaje do kontrolki `TextBox` możliwość przewijania wartości w górę i w dół. Przewijane mogą być wartości liczbowe albo wartości wskazane na liście.
`PagingBulletedListExtender`	Rozszerza działanie kontrolki `BulletedList` o możliwość stronicowania i sortowania po stronie klienta.
`PasswordStrength`	Pomaga użytkownikowi zdefiniować hasło trudne do złamania.
`Rating`	Kontrolka umożliwiająca dokonanie oceny przez wskazanie liczby gwiazdek z uwzględnieniem jej maksymalnego pułapu (kontrolki można użyć na przykład do zdefiniowania 4-gwiazdkowego, 5-gwiazdkowego albo 10-gwiazdkowego systemu oceny restauracji albo filmów wyświetlanych w kinach).
`ReorderList`	Pozwala użytkownikowi na zmianę kolejności elementów listy przez przeciągnięcie ich myszą na nowe pozycje.
`TextBoxWaterMarkExtender`	Wyświetla w polu tekstowym tekst pomocy i ukrywa go, gdy użytkownik rozpocznie wpisywanie danych w tym polu.
`UpdatePanelAnimationExtender`	Zapewnia animowanie panelu: przenoszenie, zmianę rozmiaru i znikanie.
`ValidatorCalloutExtender`	Rozszerza działanie kontrolek weryfikacji poprawności ASP.NET o możliwość wyświetlania ostrzeżenia z obrazkiem w przypadku, gdy wartość w polu okaże się nieprawidłowa.

Aby przekonać się, jak działa znak wodny, należy skopiować poprzedni przykład *AdventureWorksRevisited* do nowej witryny o nazwie *AdventureWorksWatermarks*.

Zanim przystąpimy do wprowadzania zmian na stronie, trzeba najpierw utworzyć arkusz stylów, w którym zdefiniowane będą style tekstu ze znakiem wodnym i bez niego.

Style i arkusze stylów zostaną szczegółowo opisane w rozdziale 6., dlatego w tym punkcie zaprezentujemy jedynie najważniejsze informacje na ich temat.

Aby utworzyć arkusz stylów, należy wybrać polecenie *Website/Add New Item…*. W oknie dialogowym *Add New Item* trzeba wskazać opcję *Style Sheet*, zaakceptować domyślną nazwę arkusza stylów *StyleSheet.css* i kliknąć przycisk *Add*, jak na rysunku 3.6.



Rysunek 3.6. Aby do witryny WWW dodać arkusz stylów, należy skorzystać z okna dialogowego Add New Item

W odpowiedzi w edytorze otwarty zostanie arkusz stylów z pustym elementem `body`. Do arkusza należy dopisać kod wyróżniony na listingu 3.1.

Listing 3.1. Zawartość arkusza stylów StyleSheet.css dla znaków wodnych

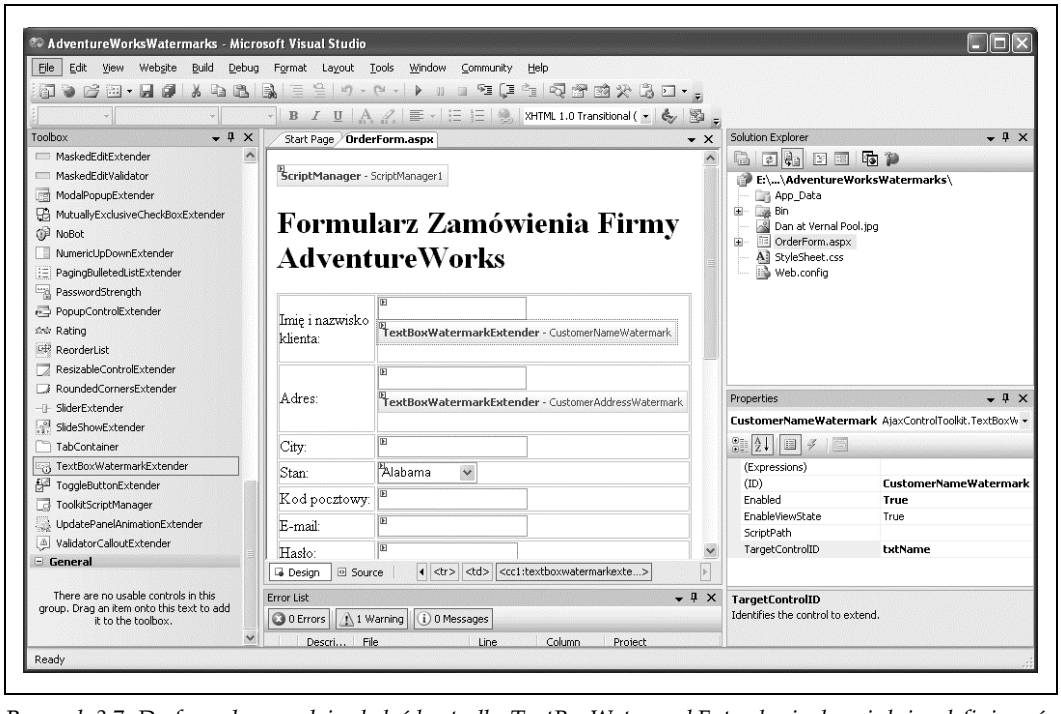
```
body {
}
.watermarked {
   padding:2px 0 0 2px;
   border:1px solid #BEBEBE;
   background-color:#F0F8FF;
   color:gray;
   font-family:Verdana;
   font-weight:normal;
}
.unwatermarked {
   height:18px;
   width:148px;
   font-weight:bold;
}
```

Przedstawiony kod dodaje dwie klasy stylów: `watermarked` i `unwatermarked`, które w niniejszym przykładzie będą stosowane dla tekstów.

W oknie *Design* trzeba otworzyć plik *OrderForm.aspx*, zaznaczyć pole tekstowe `txtName` służące do wpisywania imienia i nazwiska klienta i w oknie właściwości *Properties* przypisać właściwości `CssClass` wartość `unwatermarked` (wielkości liter w nazwach klas stylów są uwzględniane).

Tak samo należy postąpić względem pola tekstowego `txtAddress`. W ten sposób wskazana zostanie klasa stylu, który będzie stosowany względem tekstu wpisywanego przez użytkownika w polu tekstowym, co widać na rysunku 3.5.

W oknie narzędziowym *Toolbox* należy rozwinąć sekcję *Ajax Controls Toolkit* i przeciągnąć znajdującą się w nim kontrolkę `TextBoxWatermarkExtender` do tej samej komórki tabeli, w której znajduje się pole tekstowe `txtName`. Właściwości `ID` nowej kontrolki należy przypisać wartość `CustomerNameWatermark`, zaś jako `TargetControlID` wpisać `txtName`. Właściwość `ID` spełnia taką samą rolę, co `ID` wszystkich innych kontrolek, natomiast `TargetControlID` wskazuje kontrolkę, w której ma zostać zastosowany efekt znaku wodnego — w naszym przypadku jest to kontrolka pola tekstowego `TextBox` o nazwie `txtName`. Ekran powinien się prezentować podobnie jak na rysunku 3.7 (nietrudno zauważyć, że w oknie *Properties* właściwości posortowano alfabetycznie).



Rysunek 3.7. Do formularza należy dodać kontrolkę TextBoxWatermarkExtender i odpowiednio zdefiniować jej właściwości ID i TargetControlID

Konieczne jest jeszcze odpowiednie zdefiniowanie dwóch dodatkowych właściwości kontrolki: `WatermarkCssClass` oraz `WatermarkText`. Niestety, właściwości te nie są dostępne w oknie właściwości *Properties*, dlatego trzeba przełączyć się do widoku *Source* i wpisać je ręcznie. Jednak przedtem do komórki tabeli, w której znajduje się pole tekstowe `txtAddress`, należy przeciągnąć kolejną kontrolkę `TextBoxWatermarkExtender` i jako jej identyfikator `ID` wpisać `CustomerAddressWatermark`, zaś właściwości `TargetControlID` przypisać wartość `txtAddress`.

Po umieszczeniu kontrolki w komórce tabeli należy wrócić do widoku *Source* i dodać atrybuty `WatermarkCssClass` oraz `WatermarkText`. Do obydwóch kontrolek pola tekstowego trzeba dopisać dwa wiersze kodu, aby osiągnąć efekt przedstawiony na listingu 3.2.

Listing 3.2. Znacznik TextBoxWatermarkExtender

```
<tr>
    <td style="width: 100px">
        Imię i nazwisko klienta:</td>
    <td style="width: 100px">
        <asp:TextBox ID="txtName" runat="server"
            CssClass="unwatermarked">
        </asp:TextBox>
        <cc1:TextBoxWatermarkExtender ID="CustomerNameWatermark" runat="server"
            TargetControlID="txtName"
            WatermarkCssClass="watermarked"
            WatermarkText="Twoje imię i nazwisko">
        </cc1:TextBoxWatermarkExtender>
    </td>
</tr>
<tr>
    <td style="width: 100px">
        Adres:</td>
    <td style="width: 100px">
        <asp:TextBox ID="txtAddress" runat="server"
            CssClass="unwatermarked"></asp:TextBox>
        <cc1:TextBoxWatermarkExtender ID="CustomerAddressWatermark" runat="server"
            TargetControlID="txtAddress"
            WatermarkCssClass="watermarked"
            WatermarkText="Adres domowy">
        </cc1:TextBoxWatermarkExtender>
    </td>
</tr>
```

Powyższy kod dodaje do kontrolki `TextBox` znak wodny, zanim użytkownik zacznie cokolwiek wpisywać. Właściwość `WatermarkText` wskazuje tekst, który ma być wyświetlany w polu `TextBox`, zaś `WatermarkCssClass` zawiera klasę stylu do zastosowania względem znaku wodnego. Klasę tę zdefiniowaliśmy wcześniej w arkuszu stylów. Efekt jest taki, że w polach tekstowych `TextBox` wyświetlane jest elegancko sformatowane przypomnienie dla użytkownika, które znika dopiero w momencie, gdy użytkownik zacznie wpisywać jakąś wartość z klawiatury. Przypomnienie zostało przedstawione na rysunku 3.4.

W ostatnim kroku w pliku znaczników, w znaczniku `<head>`, należy dopisać jeszcze jeden wiersz kodu HTML, aby arkusz stylów stał się widoczny dla strony. Bez tego wiersza na stronie nie można byłoby zastosować żadnej klasy stylów zdefiniowanych wcześniej:

```
<style type="text/css">@import url(StyleSheet.css);</style>
```

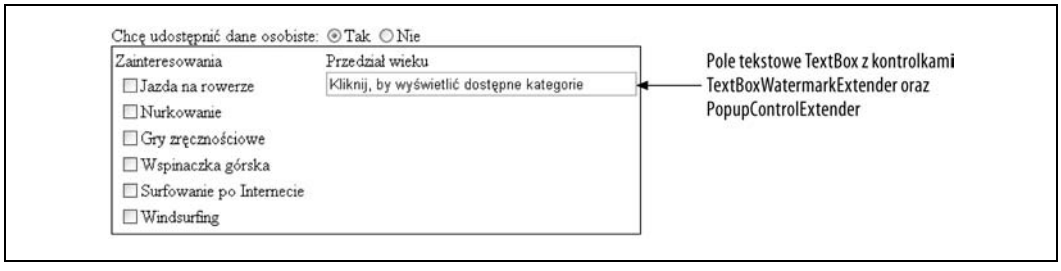
Teraz można już uruchomić aplikację. Pola tekstowe *Imię i nazwisko klienta* oraz *Adres* będą wyglądać podobnie jak na rysunku 3.4. Gdy w którymś z tych pól użytkownik zacznie wpisywać jakąś wartość, dotychczas znajdujący się w nich tekst zostanie usunięty, aby nie powodować ewentualnego zamieszania, co widać na rysunku 3.5.

Kontrolka PopupControlExtender

Oszczędne gospodarowanie miejscem dostępnym na stronie jest zawsze dobrą praktyką, dlatego kontrolka `PopupControlExtender` jawi się jako bardzo przydatne narzędzie do prezentowania jak największej ilości informacji na jak najmniejszej powierzchni. Kontrolkę `PopupControlExtender` dołącza się do innej, wybranej kontrolki. Gdy użytkownik kliknie taką kontrolkę, wówczas wyświetlane jest podręczne okno z dodatkową zawartością. Jeżeli w oknie podręcznym umieścimy kontrolkę `UpdatePanel`, będzie w nim można wyświetlać dane odczytywane z serwera w sposób asynchroniczny. Efekt będzie więc bardzo interesujący.

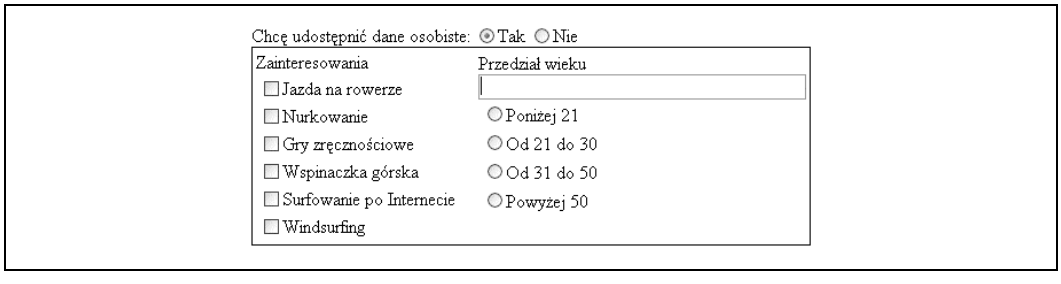
Aby zobaczyć, w jaki sposób działa kontrolka `PopupControlExtender`, zmodyfikujemy poprzedni przykład i zamiast listy przycisków opcji `RadioButtonList` służącej do wyboru kategorii wiekowej wstawimy zwykłe pole tekstowe `TextBox`. Następnie dodamy `PopupControlExtender` i połączymy ją z polem tekstowym. Kontrolka `PopupControlExtender` będzie używać panelu `UpdatePanel`, aby móc wyświetlać `RadioButtonList` w oknie podręcznym.

Na rysunku 3.8 widać pole tekstowe `TextBox`, które oczekuje na kliknięcie go przez użytkownika. Aby zachęcić użytkownika do kliknięcia pola tekstowego, wykorzystano dodatkowo kontrolkę `TextBoxWatermarkExtender`.



Rysunek 3.8. Pole tekstowe TextBox ma dodatkowo powiązane ze sobą kontrolki: PopupControlExtender oraz TextBoxWatermarkExtender, która zachęca użytkownika do kliknięcia pola myszą

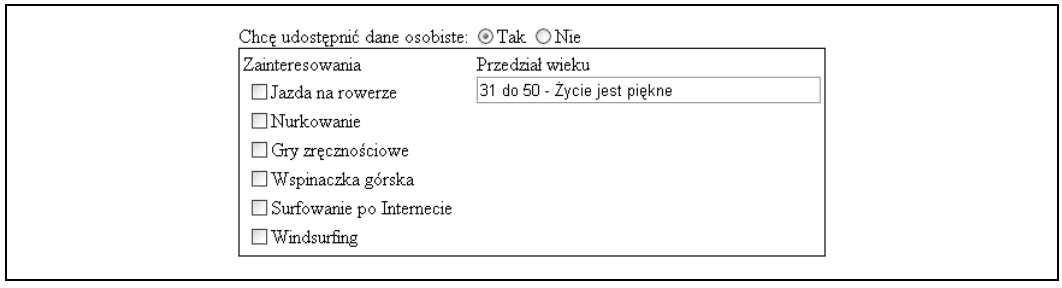
Rysunek 3.9 pokazuje, co dzieje się, gdy użytkownik kliknie myszą pole tekstowe `TextBox`. Znak wodny znika, a zamiast niego pojawia się okno podręczne zawierające listę przycisków opcji `RadioButtonList` z kategoriami wiekowymi do wyboru przez użytkownika. Ponieważ lista znajduje się wewnątrz panelu `UpdatePanel`, nie trzeba wykonywać wywołania zwrotnego do serwera (a więc unikamy także migotania ekranu). Wszystkie czynności są wykonywane po stronie klienta.



Rysunek 3.9. Gdy użytkownik kliknie myszą pole tekstowe TextBox z kontrolką PopupControlExtender, otwierane jest podręczne okno zawierające przyciski opcji

Gdy użytkownik zaznaczy konkretny przycisk opcji, wybrana wartość zostanie odesłana do serwera. Jednak również w tym przypadku użyty zostanie `UpdatePanel`, zatem cała reszta strony pozostanie bez zmian. Okno podręczne zostanie zamknięte, a w polu tekstowym `TextBox` wyświetlona zostanie wybrana wartość, jak na rysunku 3.10.

Aby zaimplementować rozwiązanie zaprezentowane powyżej, należy skopiować poprzednią przykładową aplikację *AdventureWorksWatermarks* do nowej witryny internetowej o nazwie *AdventureWorksPopupControl*. Najpierw przeciągniemy wszystkie wymagane kontrolki i uzupełnimy ich kod, a następnie opiszemy, w jaki sposób działa całe rozwiązanie.



Rysunek 3.10. Po wybraniu przez użytkownika konkretnej wartości UpdatePanel zostanie zamknięty, a sama wartość pojawi się w polu tekstowym TextBox

W poprzednim przykładzie znacznik komórki tabeli zawierającej tytuł *Przedział wieku* i przyciski opcji miał następującą postać:

```
<td style="width: 367px">
    Przedział wieku <br />
    <asp:RadioButtonList ID="rblAge" runat="server" AutoPostBack="True" Width="125px">
        <asp:ListItem>Poniżej 21</asp:ListItem>
        <asp:ListItem>Od 21 do 30</asp:ListItem>
        <asp:ListItem>Od 31 do 50</asp:ListItem>
        <asp:ListItem>Powyżej 50</asp:ListItem>
    </asp:RadioButtonList>
</td>
```

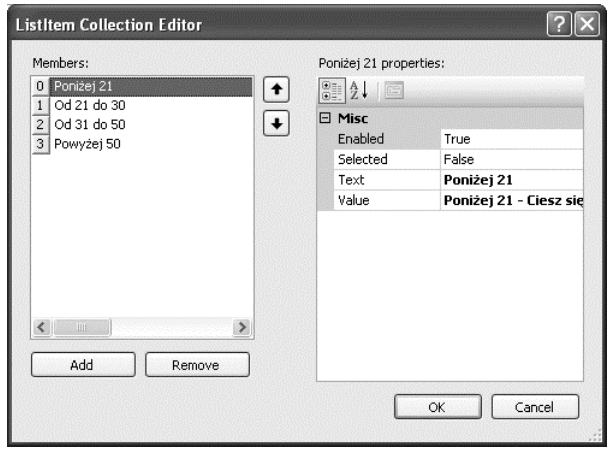
W widoku *Design* należy przeciągnąć standardową kontrolkę `TextBox` z okna narzędziowego *Toolbox* do komórki, w której aktualnie znajduje się grupa przycisków opcji `rblAge`. Jako `ID` nowego pola tekstowego należy wpisać `txtAgeCategory`, a jego właściwości `Width` przypisać wartość `280px`. Następnie z pakietu narzędziowego *Ajax Control Toolkit* okna *Toolbox* do tej samej komórki tabeli trzeba przeciągnąć kontrolki `TextBoxWatermarkExtender` oraz `PopupControlExtender`. Ponadto w komórce tej potrzebować będziemy standardowej kontrolki `Panel`, której identyfikatorem `ID` powinno być `pnlAgeCategories`.

Jako wartość właściwości `TargetControlID` kontrolki `TextBoxWatermarkExtender` należy wpisać `txtAgeCategory` (czyli `ID` nowego pola tekstowego `TextBox`), natomiast jako tekst znaku wodnego `WatermarkText` można wpisać *`Kliknij, by wyświetlić dostępne kategorie`* (tekst ten trzeba wpisać w widoku *Source*, ponieważ właściwość nie jest dostępna w widoku *Design*). Ponadto można wskazać klasę stylów `WatermarkCssClass`, jak uczyniono to w poprzednim punkcie, jednak nie będziemy się tym przejmować w tym przykładzie.

Pozostając nadal w widoku *Source* (niestety, wciąż nie wszystkie właściwości są dostępne w widoku projektowania *Design*), należy zdefiniować właściwości kontrolki `PopupControlExtender`. Właściwości `TargetControlID` należy przypisać identyfikator `ID` pola `TextBox` (`txtAgeCategory`). Dzięki temu okno podręczne zostanie wyświetlone, gdy użytkownik kliknie pole tekstowe `txtAgeCategory`. Jako `PopupControlID` należy wpisać `pnlAgeCategories` — jest to kontrolka, która zostanie wyświetlona w momencie kliknięcia pola tekstowego (samą kontrolkę `Panel` wypełnimy odpowiednimi elementami za chwilę). Na końcu jako `Position` kontrolki `PopupControlExtender` należy wskazać `Bottom`.

Teraz trzeba wypełnić kontrolkę `Panel`. W tym celu należy przełączyć się do widoku *Design*, a następnie z sekcji *Ajax Extensions* okna narzędziowego *Toolbox* do panelu `pnlAgeCategories` przeciągnąć kontrolkę `UpdatePanel`. Następnie już wcześniej istniejącą listę przycisków `RadioButtonList` o nazwie `rblAge` trzeba umieścić wewnątrz dopiero co przeciągniętego panelu `UpdatePanel`.

Moglibyśmy w tym momencie już zakończyć prace i wszystko działałoby zgodnie z oczekiwaniami. Warto jednak jeszcze jawnie zdefiniować wartości `Value` dla każdego elementu znajdującego się na liście przycisków opcji `RadioButtonList`. W tym celu trzeba kliknąć tag inteligentny kontrolki `RadioButtonList` i wybrać polecenie *Edit Items*, aby uruchomić edytor ListItem Collection Editor, widoczny na rysunku 3.11.



Rysunek 3.11. Po kliknięciu tagu inteligentnego kontrolki RadioButtonList zostanie uruchomiony edytor ListItem Collection Editor, służący do edycji elementów listy

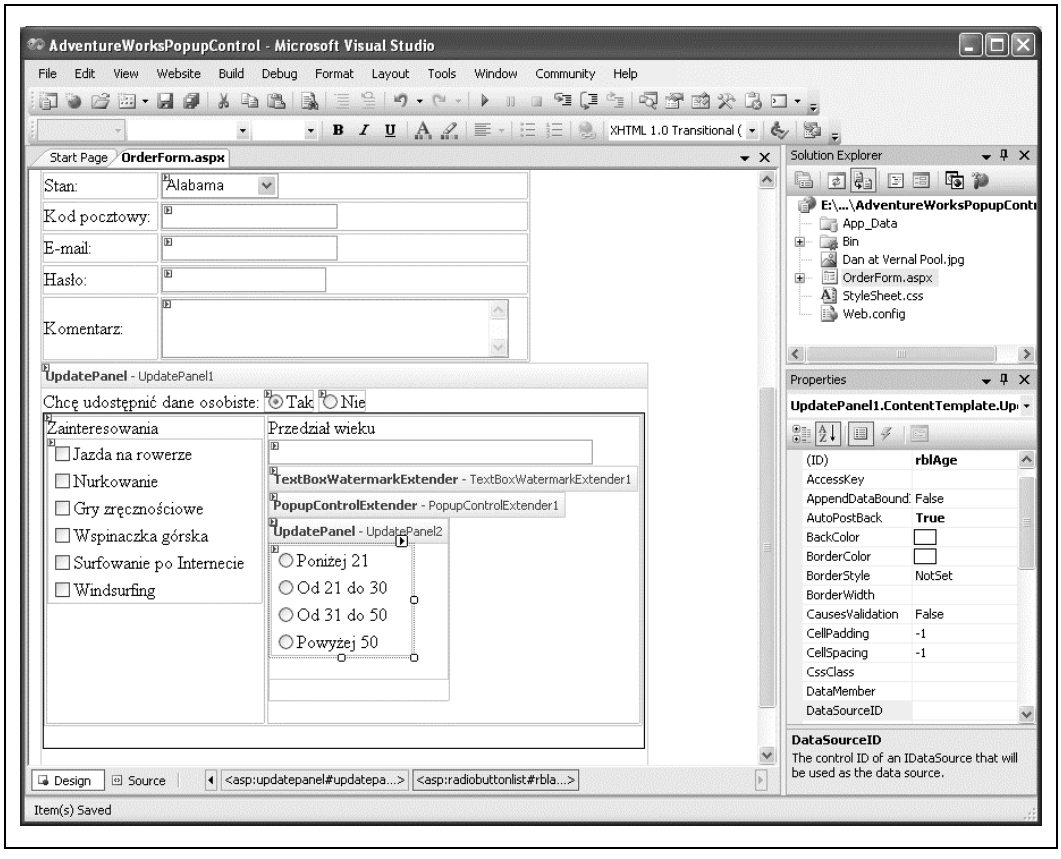
W oknie dialogowym *ListItem Collection Editor* należy kolejno wybierać wszystkie pozycje listy widoczne w oknie *Members* i zmieniać ich wartości `Value` zgodnie z poniższą tabelą:

Text	Value
Poniżej 21	Poniżej 21 — Ciesz się!
Od 21 do 30	21 do 30 — Życie na maksa
Od 31 do 50	31 do 50 — Życie jest piękne
Powyżej 50	Powyżej 50 — Złote lata

Gdy wszystkie wartości zostaną już zmodyfikowane, widok *Design* powinien się prezentować podobnie jak na rysunku 3.12.

Warto poświęcić dodatkową chwilę i wrócić do widoku *Source*, aby rzucić okiem na znaczniki wygenerowane przez środowisko IDE. Znaczniki te można oczywiście wpisać ręcznie w widoku *Source*, jednak wtedy trzeba by umieścić znacznik `<contentTemplate>` wewnątrz znacznika `<updatePanel>`. Natomiast IDE wykonuje tę operację automatycznie, gdy w widoku *Design* wykonana zostanie operacja przeciągania i upuszczania.

W ostatnim kroku trzeba zaimplementować procedurę obsługi zdarzenia dla kontrolki `Radio↪ButtonList` (o nazwie `rblAge`), która będzie obsługiwać zdarzenie polegające na zmianie wybranej opcji. Można tego dokonać w bardzo prosty sposób, podobny do stosowanego już wcześniej — wystarczy w widoku *Design* dwukrotnie kliknąć kontrolkę `rblAge`. Spowoduje to



Rysunek 3.12. Widok projektowania Design formularza zawierającego kontrolkę PopupControlExtender. Warto zwrócić uwagę na panel UpdatePanel i znajdującą się w nim listę przycisków opcji RadioButtonList

otwarcie pliku z kodem źródłowym, utworzenie szablonu procedury obsługi zdarzenia o nazwie `rblAge_SelectedIndexChanged` i umieszczenie we wnętrzu tej metody kursora, dzięki czemu od razu będzie można przystąpić do implementacji. Należy zatem wpisać następujący wiersz kodu:

```
PopupControlExtender1.Commit(rblAge.SelectedValue.ToString())
```

Wpisany kod będzie wykonywany za każdym razem, gdy użytkownik zmieni opcję wybraną na liście `RadioButtonList`. Wartość wybrana przez użytkownika zostanie przekształcona w ciąg znaków przez metodę `ToString()`, zaś wywołanie metody `Commit` wskaże kontrolce `PopupControlExtender`, że należy wymusić automatyczne odświeżenie strony.

`ToString()` jest specjalną metodą, przekształcającą wartości tekstowe w ciągi znaków, które można następnie wyświetlać jako tekst w etykietach albo w innej przeznaczonej do tego kontrolce.

Komórka tabeli zawierająca elementy listy *Przedział wieku* powinna prezentować się w widoku *Source* w sposób przedstawiony na listingu 3.3.

Listing 3.3. Kontrolka PopupControlExtender

```
<td style="width: 1O885px">
    Przedział wieku <br />
    <asp:TextBox ID="txtAgeCategory" runat="server" Width="28Opx"></asp:TextBox>

    <cc1:TextBoxWatermarkExtender ID="TextBoxWatermarkExtender1"
        runat="server"
        TargetControlID="txtAgeCategory"
        WatermarkText="Kliknij, by wyświetlić dostępne kategorie">
    </cc1:TextBoxWatermarkExtender>

    <cc1:PopupControlExtender ID="PopupControlExtender1"
        runat="server"
        TargetControlID="txtAgeCategory"
        PopupControlID="pnlAgeCategories"
        Position="Bottom">
    </cc1:PopupControlExtender>

    <asp:Panel ID="pnlAgeCategories" runat="server" Height="5Opx"
        Width="125px">
        <asp:UpdatePanel ID="UpdatePanel2" runat="server">
            <ContentTemplate>
                <asp:RadioButtonList ID="rblAge" runat="server"
                    AutoPostBack="True" Width="125px"
                    OnSelectedIndexChanged="rblAge_SelectedIndexChanged">
                    <asp:ListItem Value="Poniżej 21 - Ciesz się!">Poniżej
                       21</asp:ListItem>
                    <asp:ListItem Value="21 do 3O - Życie na maksa">Od 21 do
                       3O</asp:ListItem>
                    <asp:ListItem Value="31 do 5O - Życie jest piękne">Od 31 do
                       5O</asp:ListItem>
                    <asp:ListItem Value="Powyżej 5O - Złote lata">Powyżej
                       5O</asp:ListItem>
                </asp:RadioButtonList>
            </ContentTemplate>
        </asp:UpdatePanel>
    </asp:Panel>
     
</td>
```

Głowa do góry! Kod wygląda na skomplikowany, ale tak naprawdę nietrudno go zrozumieć.

Cały powyższy kod opisuje kontrolkę `TextBox` (o identyfikatorze `ID` `txtAgeCategory`), dwie kontrolki rozszerzające oraz kontrolkę `Panel`. Pierwszą kontrolką rozszerzającą jest `TextBoxWatermarkExtender`, drugą zaś kontrolka `PopupControlExtender` (fakt, że ze sobą sąsiadują, jest przypadkowy). Obydwie kontrolki rozszerzające mają zdefiniowaną właściwość `TargetControlID`, która zawiera identyfikator `ID` pola tekstowego `txtAgeCategory`, zresztą nietrudno zrozumieć dlaczego. Otóż zarówno `PopupControlExtender`, jak i `TextBoxWatermarkExtender` „rozszerzają" możliwości pola tekstowego `TextBox` o nazwie `txtAgeCategory`, dlatego celem obydwu kontrolek jest właśnie `txtAgeCategory`.

Definicja kontrolki `TextBoxWatermarkExtender` nie wymaga dodatkowych wyjaśnień, większe trudności może natomiast sprawić `PopupControlExtender`. Kontrolka ta nie tylko musi mieć wskazaną kontrolkę docelową (tzn. którą kontrolkę rozszerza), ale również znać `ID` własnej kontrolki `PopupControl` — czyli kontrolki, która ma zostać wyświetlona, gdy nadejdzie odpowiedni moment.

W naszym przykładzie wyświetlaną kontrolką jest `pnlAgeCategories`, czyli kontrolka ASP.NET `Panel`, która jednocześnie zawiera w sobie inne kontrolki. Pierwszą kontrolką znajdującą się

wewnątrz panelu jest `UpdatePanel` o nazwie `UpdatePanel2`. Jak wiadomo, wszystkie elementy znajdujące się wewnątrz panelu `UpdatePanel` są uaktualniane asynchronicznie, dlatego listę przycisków opcji `RadioButtonList` umieszczono właśnie wewnątrz `UpdatePanel`.

Sama kontrolka `RadioButtonList` zawiera serię elementów listy `ListItem`. Element listy `ListItem` może mieć dwie bardzo ważne właściwości (w naszym przypadku rzeczywiście je posiada). Właściwościami tymi są: tekst przeznaczony do wyświetlenia (znajdujący się między nawiasami otwierającym i zamykającym) oraz właściwość `Value`. W niektórych przypadkach właściwość `Value` okazuje się szczególnie użyteczna, ponieważ na jej podstawie programista może od razu odczytywać wartość, która jest powiązana z opcją wybraną przez użytkownika i nie musi w tym celu rozszyfrowywać wyświetlanego tekstu.

Dla kontrolki `RadioButtonList` zdefiniowano dodatkowo procedurę obsługi zdarzenia `SelectedIndexChanged`. Za każdym razem, gdy użytkownik zaznaczy któryś z przycisków opcji, nastąpi wywołanie zaimplementowanej metody (w naszym przykładzie noszącej nazwę `rblAge_SelectedIndexChanged`), co pozwoli kontrolce na odpowiednie zareagowanie na dokonaną zmianę. W przedstawionym przykładzie reakcja sprowadza się do odczytania wartości powiązanej z wybranym przyciskiem opcji i wyświetlenia go w polu tekstowym. W tym celu wywołuje się metodę `Commit` kontrolki `PopupControlExtender`.

Właściwość `Position` kontrolki `PopupControlExtender` ma wartość `Bottom`, dzięki czemu okno podręczne jest wyświetlane pod kontrolką docelową. Właściwość `Position` może przyjmować jedną z następujących dozwolonych wartości: `Bottom`, `Center`, `Left`, `Right` i `Top`.

Kontrolka CollapsiblePanelExtender

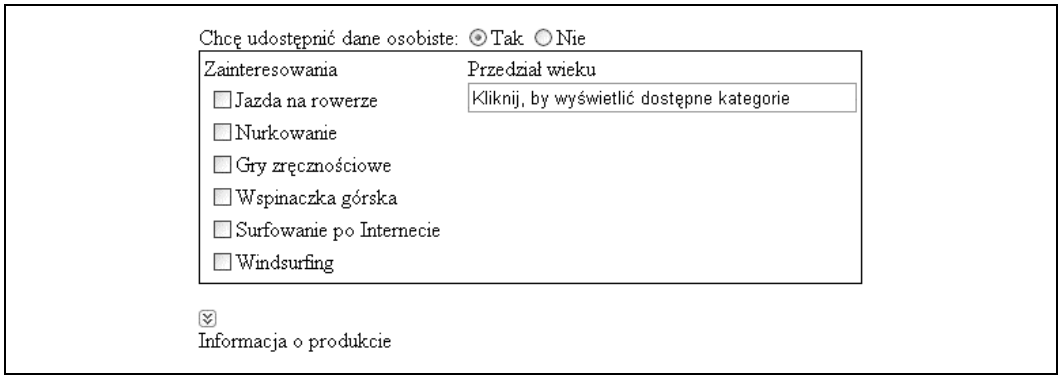
Kontrolka `CollapsiblePanelExtender` rozszerza działanie standardowej kontrolki `Panel` w ten sposób, że panel można zwijać i rozwijać. Dzięki temu na stronie można definiować regiony, które użytkownik może zwijać i rozwijać zgodnie z własną wolą. Panel rozwijany można zastosować na przykład do wyświetlania szczegółowych informacji na temat produktu, które mogą zainteresować klienta.

W naszym przykładzie w panelu zostanie umieszczony tekst oraz fotografia jednego z autorów tej książki, tak aby jego dzieciom wydawało się, że wniósł on niebagatelny wkład w jej powstanie — choć, prawdę mówiąc, wkład ten był mizerny.

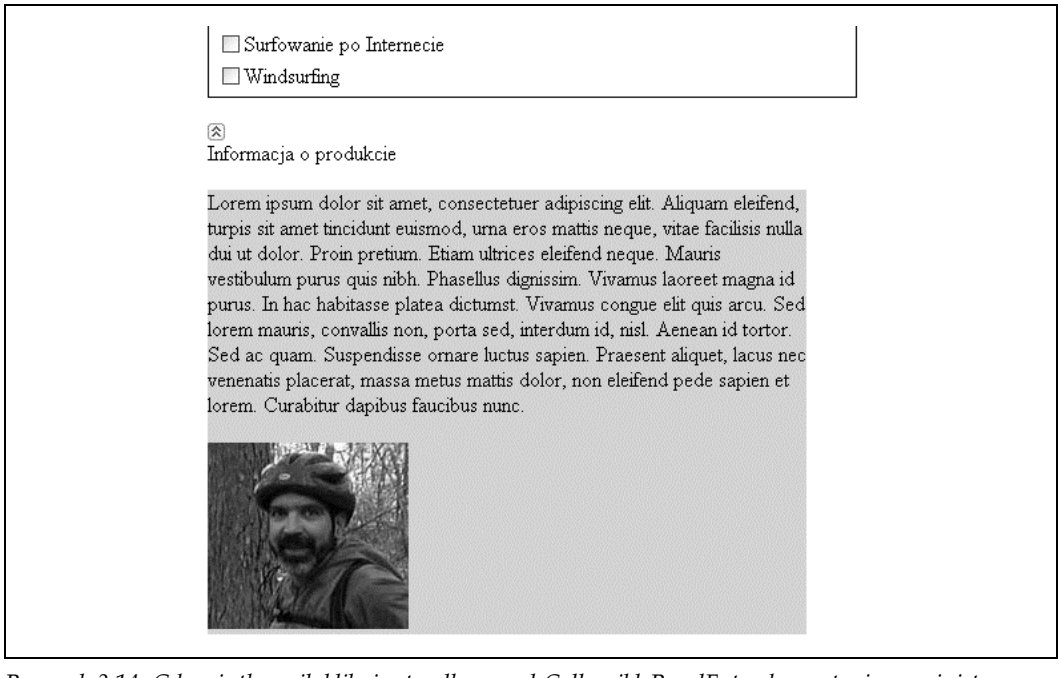
Tekst Lorem Ipsum to standardowy wypełniacz tekstowy stosowany już od bez mała 600 lat, który drukarze nazywają **greką**. Sam tekst ma umożliwić jego czytelnikowi skupienie się nie na konkretnych słowach, lecz na ich układzie. Fragment wywodzi się z dzieła Cycerona pod tytułem *De finibus bonorum et malorum* (*O granicy dobra i zła*).

Panel można samodzielnie zwijać i rozwijać w dowolnym momencie. Gdy `CollapsiblePanelExtender` będzie zwinięty, strona będzie wyglądać podobnie jak na rysunku 3.13. Natomiast po jego rozwinięciu strona będzie wyglądać jak na rysunku 3.14.

Aby sworzyć przykładową aplikację, należy skopiować wcześniej utworzoną witrynę *AdventureWorksPopupControl* do nowej witryny WWW o nazwie *AdventureWorksCollapsiblePanelExtender*. Więcej informacji na temat kopiowania witryn znajduje się w dodatku A.



Rysunek 3.13. Gdy panel CollapsiblePanelExtender jest zwinięty, nie jest on w ogóle widoczny na stronie, a o jego obecności świadczy jedynie strzałka



Rysunek 3.14. Gdy użytkownik kliknie strzałkę, panel CollapsiblePanelExtender zostanie rozwinięty i na stronie ukaże się tekst oraz obrazek

W widoku projektowania *Design* na stronę trzeba przeciągnąć standardową kontrolkę ASP.NET `Panel` (a nie kontrolkę Ajax `UpdatePanel`), dostępną w sekcji *Standard* okna narzędziowego *Toolbox*, i umieścić ją pod pozostałymi kontrolkami już obecnymi na stronie. Jako `ID` kontrolki `Panel` należy wpisać `pnlProductInfoHeader`. W panelu znajdować się będzie obrazek, za pomocą którego użytkownicy będą ten panel zwijać, oraz tekst informujący użytkownika o zawartości zwiniętego panelu. Aby móc prezentować obrazek, należy do panelu przeciągnąć kontrolkę `Image`, a następnie w oknie właściwości *Properties* jako `ID` obrazka wpisać `imgProductInfo_Toggle` ➥`Image`, zaś właściwości `ImagUrl` przypisać wartość *collapse.jpg*. Z kolei bezpośrednio w panelu należy wpisać słowa `Informacja o produkcie`.

Obrazki ze strzałkami, znajdujące się nad słowami „Informacja o produkcie" i widoczne na rysunkach 3.13 i 3.14, noszą nazwy, odpowiednio: *collapse.jpg* oraz *expand.jpg*. Obydwa obrazki zostały dołączone do kodu źródłowego przykładowej aplikacji. Aby wskazać je (albo dowolne inne obrazki) bezpośrednio w oknie właściwości *Properties*, trzeba najpierw dołączyć pliki obrazków do witryny WWW poleceniem *Website/Add Existing Item*….

Znacznik kontrolki `Panel`, który można odczytać po przełączeniu się do widoku *Source*, powinien mieć następującą postać:

```
<asp:Panel ID="pnlProductInfoHeader" runat="server" Height="50px" Width="144px">
    <asp:Image ID="imgProductInfo_ToggleImage" runat="server"
ImageUrl="~/collapse.jpg" /><br />
    Informacja o produkcie
</asp:Panel>
```

Tak naprawdę nie ma większego znaczenia, czy w definicji kontrolki `Image` jako `ImageUrl` wpisany zostanie plik *expand.jpg*, czy *collapse.jpg*, ponieważ i tak to kontrolka `CollapsiblePanelExtender` będzie sterować wyświetlaniem odpowiedniego obrazka.

Do obszaru projektowania należy przeciągnąć i umieścić pod panelem kolejną kontrolkę `Panel`. Ten panel z kolei będzie prezentować zawartość „rozwiniętej" kontrolki `Panel`. W oknie *Properties* jako `ID` nowego panelu należy wpisać `pnlProductInfo`, jako jego kolor tła `BackColor` wybrać `LightGray` (kolor wybiera się na zakładce *Web* palety kolorów), a jako `Width` wpisać `450`. W dalszej kolejności z okna narzędziowego *Toolbox* do panelu trzeba przeciągnąć standardową kontrolkę `Label` oraz również standardową kontrolkę `Image`. W oknie *Design* albo *Source* trzeba także zdefiniować właściwości `ID` i `Text` etykiety `Label` oraz właściwości `ID` i `ImageUrl` obrazka `Image`, zgodnie z przedstawionym poniżej fragmentem kodu.

```
<asp:Panel ID="pnlProductInfo" runat="server" BackColor="LightGray" Width="450px">
    <asp:Label ID="myLabel" runat="server" Text="Lorem ipsum dolor sit amet,
consectetuer adipiscing elit. Aliquam eleifend, turpis sit amet tincidunt euismod,
urna eros mattis neque, vitae facilisis nulla dui ut dolor. Proin pretium. Etiam
ultrices eleifend neque. Mauris vestibulum purus quis nibh. Phasellus dignissim.
Vivamus laoreet magna id purus. In hac habitasse platea dictumst. Vivamus congue elit
quis arcu. Sed lorem mauris, convallis non, porta sed, interdum id, nisl. Aenean id
tortor. Sed ac quam. Suspendisse ornare luctus sapien. Praesent aliquet, lacus nec
venenatis placerat, massa metus mattis dolor, non eleifend pede sapien et lorem.
Curabitur dapibus faucibus nunc.">
    </asp:Label>
    <br />
    <br />
    <asp:Image ID="Image1" runat="server" ImageUrl="Dan at Vernal Pool.jpg" />
</asp:Panel>
```

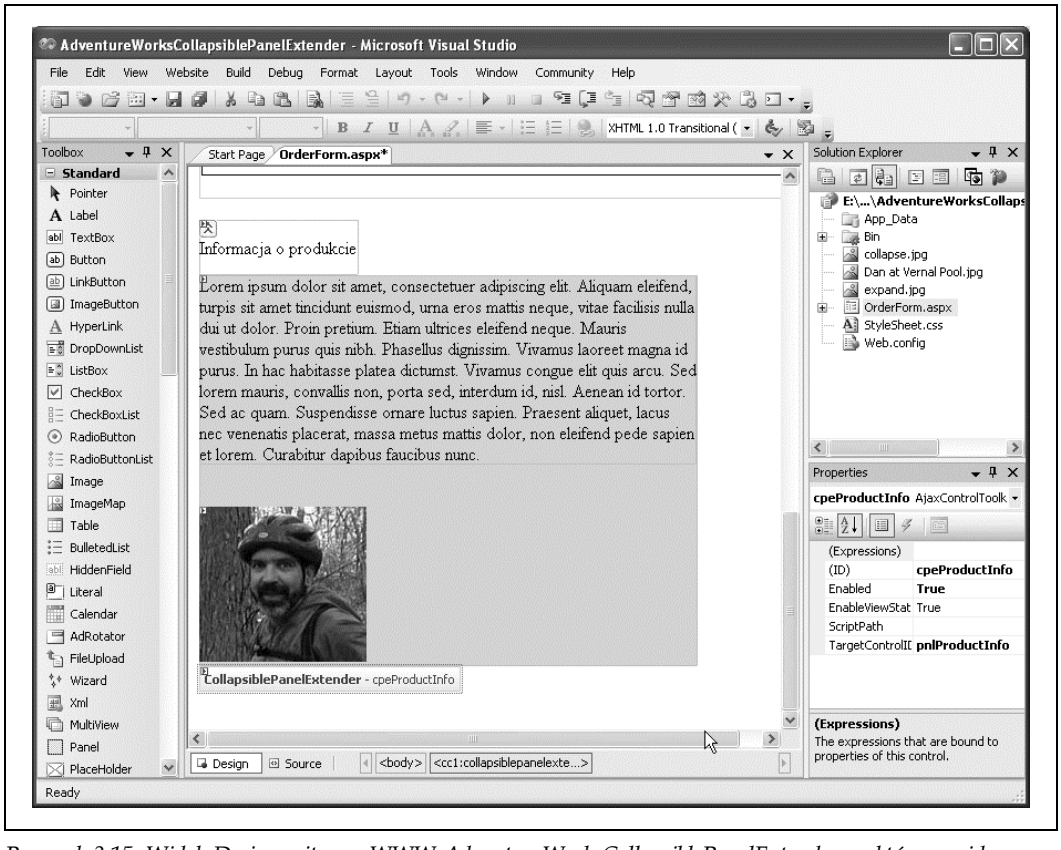
Panele zwinięty i rozwinięty znajdują się już na swoich miejscach, można zatem dodać kontrolkę Ajax `CollapsiblePanelExtender` i odpowiednio zdefiniować jej atrybuty. Kontrolkę `CollapsiblePanelExtender` należy przeciągnąć na stronę z sekcji *Ajax Control Toolkit* okna narzędziowego *Toolbox* i umieścić na samym dole. Jako identyfikator `ID` kontrolki trzeba wpisać `cpeProductInfo`, a pozostałe jej właściwości należy zdefiniować zgodnie z poniższym kodem deklaracji kontrolki:

```
<cc1:CollapsiblePanelExtender ID="cpeProductInfo" runat="server"
    CollapseControlID="pnlProductInfoHeader"
    Collapsed="true"
    CollapsedImage="expand.jpg"
```

```
    CollapsedText="Informacja o produkcie (pokaż szczegóły...)"
    ExpandControlID="pnlProductInfoHeader"
    ExpandedImage="collapse.jpg"
    ExpandedText="Informacja o produkcie (ukryj szczegóły...)"
    ImageControlID="imgProductInfo_ToggleImage"
    SuppressPostBack="true"
    TargetControlID="pnlProductInfo">
</cc1:CollapsiblePanelExtender>
```

Również w tym przypadku właściwości trzeba zdefiniować w widoku *Source*, ponieważ nie są one dostępne w widoku projektowania *Design*.

Na tym etapie widok *Design* powinien się prezentować podobnie jak na rysunku 3.15.



Rysunek 3.15. Widok Design witryny WWW AdventureWorksCollapsiblePanelExtender, w którym widoczna jest kontrolka Panel rozwijana przez użytkownika oraz kontrolka CollapsiblePanelExtender odpowiedzialna za samą operację rozwijania

Poniżej przedstawiono znaczenie poszczególnych właściwości kontrolki `CollapsiblePanelExtender`.

- `CollapseControlID/ExpandControlID` — te właściwości wskazują kontrolki, które będą, odpowiednio: rozwijać lub zwijać panel w reakcji na kliknięcie myszą przez użytkownika. Jeżeli obydwie właściwości będą mieć tę samą wartość, jak w naszym przykładzie, wówczas z każdym kliknięciem myszy przełączany będzie status samego panelu. Obydwu właściwościom należy przypisać wartość `pnlProductInfoHeader`.

- `Collapsed` — wskazuje początkowy stan rozwijanego panelu. W naszym przykładzie właściwości przypisano wartość `true`, dzięki czemu na początku panel będzie zwinięty. Gdyby właściwości przypisać wartość `false`, panel byłby na początku rozwinięty. Zazwyczaj przyjmuje się, że na początku panel powinien być zwinięty.
- `ImageControlID` — identyfikator `ID` kontrolki `Image`, w której umieszczana będzie ikona sygnalizująca stan panelu (zwinięty lub rozwinięty). Stosownie do sytuacji, kontrolka rozszerzająca zastąpi zawartość kontrolki `Image` obrazkami wskazywanymi przez adresy URL zawarte w właściwościach `CollapsedImage` i `ExpandedImage`. Jeżeli ustawione będą wartości właściwości `ExpandedText` oraz `CollapsedText`, zostaną one użyte jako tekst obrazka wyświetlany w oknie porady. W przykładowej aplikacji właściwości `ImageControlID` należy przypisać wartość `imgProductInfo_ToggleImage`.
- `CollapsedImage` — ścieżka do obrazka wyświetlanego przez `ImageControlID`, gdy `Panel` jest zwinięty. Gdy panel jest zwinięty, klienci powinni ujrzeć ikonę wskazującą, że można go rozwinąć. Zatem właściwości należy przypisać nazwę obrazka *expand.jpg*.
- `CollapsedText` — tekst, który ma być wyświetlany przez kontrolkę `CollapseControlID`, gdy panel jest zwinięty. Jest to zamienny tekst dla obrazka, gdy `ImageControlID` ma określoną wartość; jest on także wyświetlany jako porada dla użytkownika. Właściwości należy przypisać wartość `Informacja o produkcie (pokaż szczegóły...)`.
- `ExpandDirection` — tej właściwości można przypisać wartość `Vertical` lub `Horizontal`, aby wskazać, czy panel ma się rozwijać, odpowiednio: z góry na dół czy od strony lewej do prawej. W przykładowej aplikacji właściwość ma wartość `Vertical`.
- `ExpandedImage` — ścieżka do obrazka wyświetlanego przez `ImageControlID`, gdy `Panel` jest rozwinięty. Gdy panel jest rozwinięty, klienci powinni ujrzeć ikonę wskazującą, że można go zwinąć. Zatem właściwości należy przypisać nazwę obrazka *collapse.jpg*.
- `ExpandedText` — tekst, który ma być wyświetlany przez kontrolkę `CollapseControlID`, gdy panel jest rozwinięty. Jest to zamienny tekst dla obrazka, gdy `ImageControlID` ma określoną wartość; jest on także wyświetlany jako porada dla użytkownika. Właściwości należy przypisać wartość `Informacja o produkcie (ukryj szczegóły...)`.
- `SuppressPostBack` — właściwość `true` sprawi, że w momencie zwinięcia lub rozwinięcia panelu kontrolka nie wykona wywołania zwrotnego. Jest to zgodne z naszymi oczekiwaniami, zatem właściwości należy przypisać wartość `true`.
- `TargetControlID` — kontrolka podlegająca zwijaniu i rozwijaniu — w naszym przykładzie jest to `pnlProductInfo`. Kluczową rzeczą, o której należy pamiętać, jest fakt, że to nie `CollapsiblePanelExtender` jest kontrolką zwijaną albo rozwijaną, lecz odpowiada ona za zwijanie i rozwijanie *innej* kontrolki. Atrybut `TargetControlID` wskazuje panel, który będzie rozwijany przez `CollapsiblePanelExtender`.

Po uruchomieniu witryny panel będzie zwinięty, tak jak na rysunku 3.13. Kliknięcie myszą ikony znajdującej się nad tekstem *Informacja o produkcie* spowoduje rozwinięcie panelu i wyświetlenie zawartej w nim informacji, co przedstawiono na rysunku 3.14.

Kod źródłowy

Pełny kod źródłowy ostatniej przykładowej aplikacji prezentowanej w tym rozdziale przedstawiono na listingu 3.4.

Listing 3.4. OrderForm.aspx

```
<%@ Page Language="VB" AutoEventWireup="true" CodeFile="OrderForm.aspx.vb"
   Inherits="_Default" %>

<%@ Register Assembly="AjaxControlToolkit" Namespace="AjaxControlToolkit"
   TagPrefix="cc1" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <style type="text/css">@import url(StyleSheet.css);</style>
    <title>AdventureWorks - CollapsiblePanelExtender</title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:ScriptManager ID="ScriptManager1" runat="server" />
        <div>
                <h1>Formularz Zamówienia Firmy AdventureWorks</h1>
            <table>
                <tr>
                    <td style="width: 100px">
                        Imię i nazwisko klienta:</td>
                    <td style="width: 100px">
                        <asp:TextBox ID="txtName" runat="server"
                            CssClass="unwatermarked">
                        </asp:TextBox>
                        <cc1:TextBoxWatermarkExtender ID="CustomerNameWatermark"
                           runat="server"
                            TargetControlID="txtName"
                            WatermarkCssClass="watermarked"
                            WatermarkText="Twoje imię i nazwisko">
                        </cc1:TextBoxWatermarkExtender>
                    </td>
                </tr>
                <tr>
                    <td style="width: 100px">
                        Adres:</td>
                    <td style="width: 100px">
                        <asp:TextBox ID="txtAddress" runat="server"
                            CssClass="unwatermarked"></asp:TextBox>
                        <cc1:TextBoxWatermarkExtender ID="CustomerAddressWatermark"
                           runat="server"
                            TargetControlID="txtAddress"
                            WatermarkCssClass="watermarked"
                            WatermarkText="Adres domowy">
                        </cc1:TextBoxWatermarkExtender>
                    </td>
                </tr>
                <tr>
                    <td style="width: 100px">
                        Miasto:</td>
                    <td style="width: 100px">
                        <asp:TextBox ID="txtCity" runat="server"></asp:TextBox></td>
                </tr>
                <tr>
                    <td style="width: 100px">
                        Stan:</td>
                    <td style="width: 100px">
                        <asp:DropDownList ID="ddlState" runat="server">
                            <asp:ListItem Value="AL">Alabama</asp:ListItem>
                            <asp:ListItem Value="AK">Alaska</asp:ListItem>
                            <asp:ListItem Value="CA">California</asp:ListItem>
```

```
                    <asp:ListItem Value="CT">Connecticut</asp:ListItem>
                    <asp:ListItem Value="FL">Florida</asp:ListItem>
                    <asp:ListItem Value="PA">Pennsylvania</asp:ListItem>
                    <asp:ListItem Value="TX">Texas</asp:ListItem>
                    <asp:ListItem></asp:ListItem>
                </asp:DropDownList></td>
        </tr>
        <tr>
            <td style="width: 100px">
                Kod pocztowy:</td>
            <td style="width: 100px">
                <asp:TextBox ID="txtZip" runat="server"></asp:TextBox></td>
        </tr>
        <tr>
            <td style="width: 100px">
                E-mail:</td>
            <td style="width: 100px">
                <asp:TextBox ID="txtEmail" runat="server"></asp:TextBox></td>
        </tr>
        <tr>
            <td style="width: 100px">
                Hasło:</td>
            <td style="width: 100px">
                <asp:TextBox ID="txtPassword" runat="server"
                   TextMode="Password"></asp:TextBox></td>
        </tr>
        <tr>
            <td style="width: 100px">
                Komentarz:</td>
            <td style="width: 100px">
                <asp:TextBox ID="txtComment" runat="server" Rows="3"
                   TextMode="MultiLine" Width="300px"></asp:TextBox></td>
        </tr>
    </table>
</div>
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
Chcę udostępnić dane osobiste:
<asp:RadioButton ID="rbYes" runat="server" AutoPostBack="True" Checked="True"
    GroupName="grpPersonalInfo"
    Text="Tak" ToolTip="Podaj dane osobiste" />
<asp:RadioButton ID="rbNo" runat="server" AutoPostBack="True"
    GroupName="grpPersonalInfo"
    Text="Nie" ToolTip="Nie podawaj osobistych danych" /><br />
<asp:Panel ID="pnlPersonalInfo" runat="server" Height="50px" Width="125px"
   BorderWidth="1px">
    <table>
        <tr valign="top">
            <td style="width: 367px" >
                Zainteresowania<br />
                <asp:CheckBoxList ID="cblAreas" runat="server"
                   AutoPostBack="True" Width="190px">
                    <asp:ListItem>Jazda na rowerze</asp:ListItem>
                    <asp:ListItem>Nurkowanie</asp:ListItem>
                    <asp:ListItem>Gry zręcznościowe</asp:ListItem>
                    <asp:ListItem>Wspinaczka g&#243;rska</asp:ListItem>
                    <asp:ListItem>Surfowanie po Internecie</asp:ListItem>
                    <asp:ListItem>Windsurfing</asp:ListItem>
                </asp:CheckBoxList></td>
            <td style="width: 10885px">
                Przedział wieku <br />
                <asp:TextBox ID="txtAgeCategory" runat="server"
                Width="280px"></asp:TextBox>
```

```
                    <cc1:TextBoxWatermarkExtender ID="TextBoxWatermarkExtender1"
                        runat="server"
                        TargetControlID="txtAgeCategory"
                        WatermarkText="Kliknij, by wyświetlić dostępne kategorie">
                    </cc1:TextBoxWatermarkExtender>

                    <cc1:PopupControlExtender ID="PopupControlExtender1"
                        runat="server"
                        TargetControlID="txtAgeCategory"
                        PopupControlID="pnlAgeCategories"
                        Position="Bottom">
                    </cc1:PopupControlExtender>
                    <asp:Panel ID="pnlAgeCategories" runat="server" Height="50px"
                        Width="125px">
                        <asp:UpdatePanel ID="UpdatePanel2" runat="server">
                            <ContentTemplate>
                                <asp:RadioButtonList ID="rblAge" runat="server"
                                        AutoPostBack="True" Width="125px"
                                        OnSelectedIndexChanged="rblAge_
                                           SelectedIndexChanged">
                                    <asp:ListItem Value="Poniżej 21 - Ciesz
                                       się!">Poniżej 21</asp:ListItem>
                                    <asp:ListItem Value="21 do 30 - Życie na
                                       maksa">Od 21 do 30
                                            </asp:ListItem>
                                    <asp:ListItem Value="31 do 50 - Życie jest
                                       piękne">Od 31 do 50
                                            </asp:ListItem>
                                    <asp:ListItem Value="Powyżej 50 - Złote
                                       lata">Powyżej 50
                                            </asp:ListItem>
                                </asp:RadioButtonList>
                            </ContentTemplate>
                        </asp:UpdatePanel>
                    </asp:Panel>
                     
                </td>
            </tr>
        </table>
    </asp:Panel>
        </ContentTemplate>
    </asp:UpdatePanel>
    <br />
    <asp:Panel ID="pnlProductInfoHeader" runat="server" Height="50px"
       Width="144px">
        <asp:Image ID="imgProductInfo_ToggleImage" runat="server"
            ImageUrl="~/collapse.jpg" /><br />
        Informacja o produkcie</asp:Panel>
    <asp:Panel ID="pnlProductInfo" runat="server" BackColor="LightGray"
        Width="450px">
        <asp:Label ID="myLabel" runat="server" Text=
        "Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
Aliquam eleifend, turpis sit amet tincidunt euismod, urna eros mattis
neque, vitae facilisis nulla dui ut dolor. Proin pretium. Etiam ultrices
eleifend neque. Mauris vestibulum purus quis nibh. Phasellus dignissim.
Vivamus laoreet magna id purus. In hac habitasse platea dictumst. Vivamus
congue elit quis arcu. Sed lorem mauris, convallis non, porta sed, interdum
id, nisl. Aenean id tortor. Sed ac quam. Suspendisse ornare luctus sapien.
Praesent aliquet, lacus nec venenatis placerat, massa metus mattis dolor,
non eleifend pede sapien et lorem. Curabitur dapibus faucibus nunc.">
        </asp:Label>
        <br />
        <br />
        <asp:Image ID="Image1" runat="server" ImageUrl="Dan at Vernal Pool.jpg" />
```

```
        </asp:Panel>
        <cc1:CollapsiblePanelExtender ID="cpeProductInfo" runat="server"
            CollapseControlID="pnlProductInfoHeader"
            Collapsed="true"
            CollapsedImage="expand.jpg"
            CollapsedText="Informacja o produkcie (pokaż szczegóły...)"
            ExpandControlID="pnlProductInfoHeader"
            ExpandedImage="collapse.jpg"
            ExpandedText="Informacja o produkcie (ukryj szczegóły...)"
            ImageControlID="imgProductInfo_ToggleImage"
            SuppressPostBack="true"
            TargetControlID="pnlProductInfo">
        </cc1:CollapsiblePanelExtender>

    </form>
</body>
</html>
```

Na listingu 3.5 przedstawiono zawartość pliku z kodem źródłowym dla strony *OrderForm.aspx*.

Listing 3.5. OrderForm.aspx.vb

```
Partial Class _Default
    Inherits System.Web.UI.Page

    Protected Sub YesNoEventHandler(ByVal sender As Object, _
     ByVal e As System.EventArgs) _
     Handles rbNo.CheckedChanged, rbYes.CheckedChanged

        pnlPersonalInfo.Visible = rbYes.Checked
    End Sub


    Protected Sub rblAge_SelectedIndexChanged(ByVal sender As Object, ByVal e As
System.EventArgs)
        PopupControlExtender1.Commit(rblAge.SelectedValue.ToString())
    End Sub
End Class
```

Podsumowanie

- Ajax to technika przetwarzania kodu na serwerze oraz w przeglądarce użytkownika, która zdecydowanie zwiększa wydajność aplikacji — zarówno rzeczywistą, jak i odczuwaną przez użytkownika.
- Biblioteka kontrolek Ajax dostępna w ASP.NET zawiera szereg kontrolek, których można używać równie łatwo jak w przypadku standardowych kontrolek ASP.NET. Oznacza to, że nie trzeba znać kodu JavaScript definiującego sposób działania kontrolek.
- Kontrolka `ScriptManager` to kluczowa kontrolka, która zarządza kodem JavaScript i dzięki temu umożliwia integrację technologii ASP.NET i Ajax. Kontrolka jest domyślnie umieszczana na każdej stronie korzystającej z technologii Ajax, a jej właściwość `EnablePartialRendering` domyślnie ma wartość `True`, a więc programista nie musi jej dodatkowo parametryzować.
- Dzięki umieszczeniu kontrolek wewnątrz kontrolki `UpdatePanel` nie trzeba w celu ich uaktualnienia wykonywać wywołania zwrotnego do serwera.

- Pakiet narzędziowy Ajax Control Toolkit, dostępny jako oddzielny pakiet do pobrania z internetu, udostępnia znaczną liczbę tak zwanych kontrolek rozszerzających, które rozszerzają funkcje kontrolek istniejących, a nie są kontrolkami działającymi samodzielnie. Kontrolki rozszerzające mają właściwość `TargetControlID` służącą do wskazania kontrolki standardowej, której funkcje mają zostać rozszerzone.
- Kontrolka `TextBoxWaterMarkExtender` dodaje do istniejącego pola tekstowego efekt znaku wodnego, który stanowi dla użytkownika zachętę do wpisania określonych danych.
- Kontrolka `TextBoxWaterMarkExtender` potrafi zastosować w polu tekstowym odrębny styl, jeśli tylko w projekcie zdefiniowano arkusz stylów. Kontrolka może także dodać tekst wskazany przez programistę.
- Kontrolka `PopupControlExtender` pomaga zaoszczędzić miejsce na stronie WWW, ponieważ umożliwia ukrywanie określonej zawartości i jej prezentowanie dopiero po kliknięciu kontrolki myszą przez użytkownika.
- Kontrolkę `CollapsiblePanelExtender` można zastosować jako dodatek do zwykłej kontrolki `Panel`, aby ukrywać większość zawartości strony i wyświetlać ją dopiero po kliknięciu danego elementu przez użytkownika. W odpowiedzi panel zostanie rozwinięty i jego zawartość stanie się widoczna do czasu, gdy użytkownik go zwinie.

Znamy już dość dobrze podstawowe kontrolki oraz wiemy, w jaki sposób można używać kontrolek rozszerzających Ajax, aby osiągnąć dzięki nim bardzo interesujące efekty. Teraz *Formularz Zamówień Firmy AdventureWorks*, który tworzyliśmy etapami, wygląda już całkiem efektownie. Jednak jak już wcześniej wspomnieliśmy, formularz nie łączy się z żadnym źródłem danych, dlatego użytkownicy nie mogą przeglądać produktów firmy AdventureWorks ani składać zamówień na nie. Aby im to umożliwić, musimy najpierw poznać sposoby interakcji z bazami danych. ASP.NET udostępnia liczne kontrolki służące do odczytywania danych z bazy i wyświetlania ich na różne sposoby. A dzięki technologii Ajax kontrolki te można jeszcze rozbudowywać. Wszystkie niezbędne informacje na ten temat są przedstawione w następnym rozdziale.

Quiz

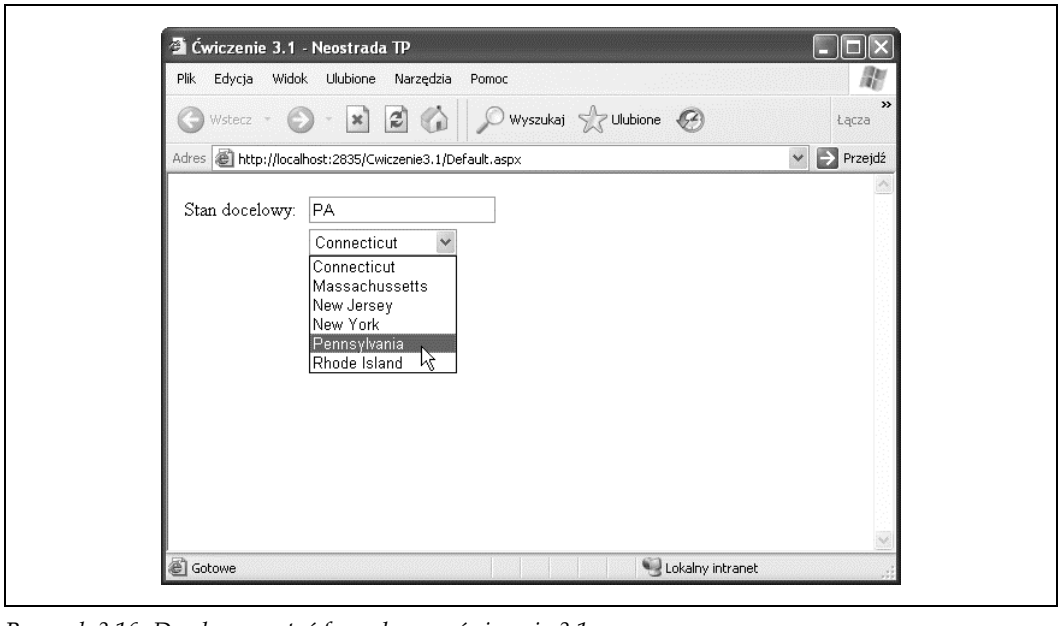
1. Czego potrzeba, aby można było skorzystać z kontrolki `ScriptManager`?
2. Która właściwość kontrolki `ScriptManager` umożliwia wykonywanie wywołań asynchronicznych?
3. Jaką kontrolkę należy umieścić na stronie, aby umożliwić asynchroniczne uaktualnianie jej zawartości?
4. Czy można utworzyć stronę, która będzie zawierała wyłącznie kontrolkę rozszerzającą?
5. Jaka właściwość jest wspólna dla wszystkich kontrolek rozszerzających Ajax?
6. W jakim widoku należy definiować właściwość `WatermarkText`?
7. Czy kontrolki `TextBoxWatermarkExtender` można używać bez zdefiniowanego arkusza stylów?
8. Jaka jest zaleta używania kontrolki `PopupExtenderControl`?

9. Jaką metodę kontrolki `PopupExtenderControl` trzeba wywołać, aby wyświetlić wyniki jej działania?
10. Z jakimi standardowymi kontrolkami współpracuje kontrolka `CollapsiblePanelExtender`?

Ćwiczenia

Ćwiczenie 3.1

Pierwsze ćwiczenie będzie proste. Załóżmy, że prowadzimy sklep, który wysyła zamówione towary wyłącznie do niektórych stanów w północno-wschodniej części Stanów Zjednoczonych. W formularzu zamówienia należy ograniczyć użytkownikom możliwość wyboru adresu docelowego wyłącznie do tych stanów, potrzebna zatem będzie lista rozwijana. Chcemy także zaoszczędzić miejsce na formularzu, a więc lista stanów powinna być ukrywana w kontrolce `Panel` z kontrolką `PopupControlExtender`. W ramach tego ćwiczenia trzeba jedynie utworzyć fragment formularza, w którym użytkownik będzie wskazywał docelowy stan. Gotowy formularz przedstawiono na rysunku 3.16.



Rysunek 3.16. Docelowa postać formularza z ćwiczenia 3.1

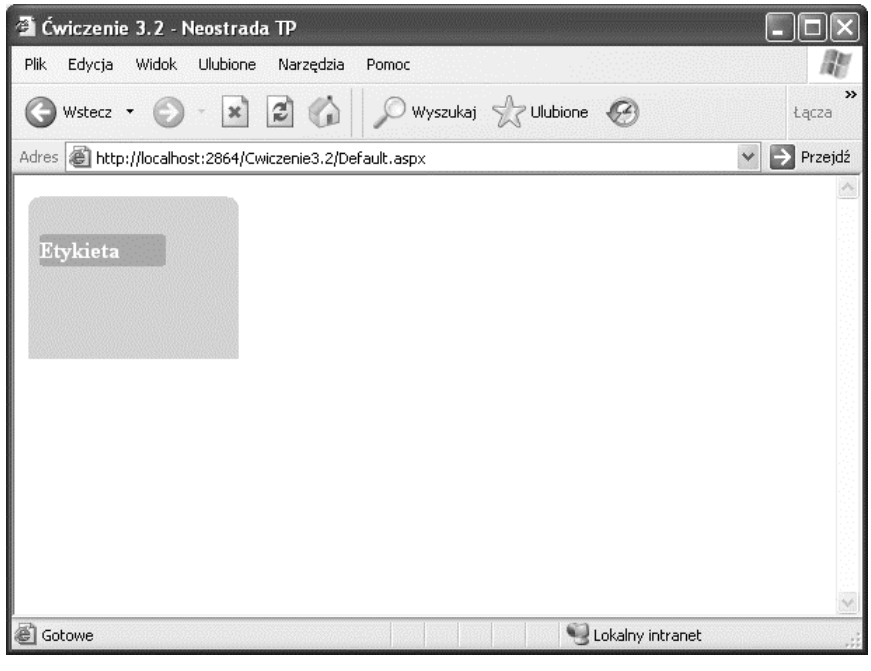
Lista rozwijana powinna zawierać tylko sześć stanów widocznych na rysunku 3.16. Gdy użytkownik wybierze jeden z nich, wówczas w polu tekstowym powinien pojawić się dwuliterowy skrót pocztowy tego stanu.

Ćwiczenie 3.2

Większość kontrolek rozszerzających Ajax, które zaprezentowano w tym rozdziale, realizuje tylko jedną funkcję, choć — trzeba przyznać — realizuje ją bardzo dobrze. Istnieje jednak wiele innych kontrolek rozszerzających, o których jeszcze nie mówiliśmy, a kolejne nowe kontrolki

są przez cały czas udostępniane. Każda kontrolka jest inna, a opisanie ich wszystkich wymagałoby znacznie więcej miejsca, niż mamy, albo ich opis w momencie zakończenia mógłby już być nieaktualny. Najlepszym źródłem informacji na temat kontrolek rozszerzających Ajax jest witryna ASP.NET Ajax Control Toolkit znajdująca się pod adresem *http://ajax.asp.net/ajaxtoolkit/*. W witrynie dostępne są najnowsze kontrolki rozszerzające oraz przykłady ich zastosowania. Większość kontrolek rozszerzających nie jest zbyt skomplikowana, a definiowanie ich właściwości jest łatwym zadaniem. W tym ćwiczeniu będzie trzeba skorzystać ze wspomnianej dokumentacji, aby poznać działanie nowej kontrolki rozszerzającej.

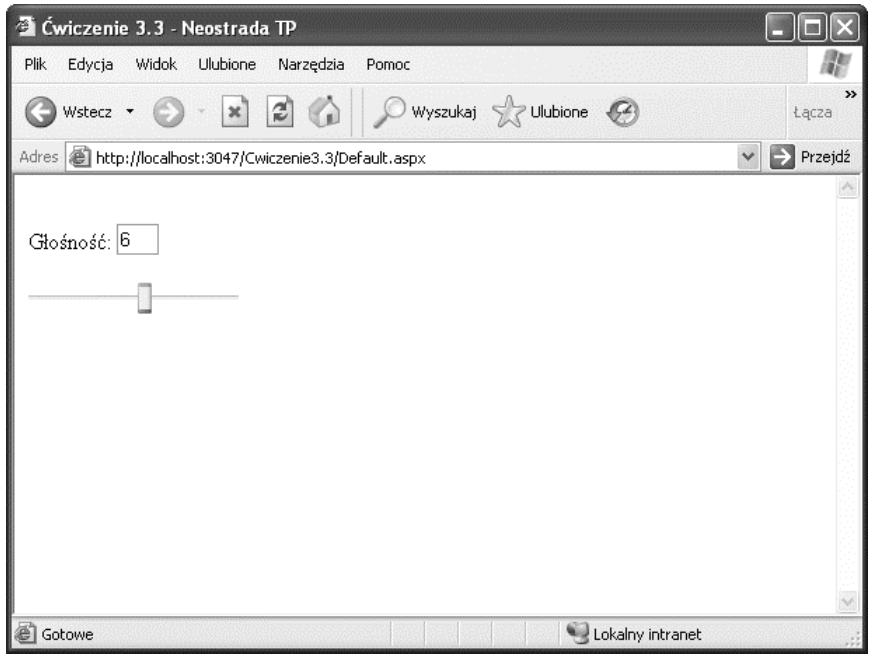
W ćwiczeniu wykorzystamy kontrolkę rozszerzającą `RoundedCorners`, pozwalającą na zaokrąglanie rogów. W celu jego wykonania należy utworzyć nową stronę, która będzie zawierać `Panel` o rozmiarach *150* pikseli na *100* pikseli, w kolorze jasnoszarym. W panelu powinna również znajdować się etykieta szeroka na *90* pikseli, z tłem w kolorze ciemnoszarym i tekstem wyświetlanym czcionką białą i pogrubioną (oczywiście można także zastosować inne kolory wybrane samodzielnie; kolory wybrane przez nas będą się po prostu dobrze prezentować w książce). W panelu zaokrąglone powinny być wyłącznie jego górne rogi, a zaokrąglenie powinno mieć promień równy *8*. Z kolei etykieta `Label` powinna mieć zaokrąglone wszystkie rogi, a promień zaokrąglenia powinien wynosić *2*. W celu wykonania tego ćwiczenia konieczne będzie odwołanie się do dokumentacji kontrolki. Gotowa strona została przedstawiona na rysunku 3.17.



Rysunek 3.17. Ukończona strona z ćwiczenia 3.2

Ćwiczenie 3.3

`SliderExtender` to kolejna interesująca kontrolka rozszerzająca, która jednak jest nieco bardziej skomplikowana, niż mogłoby się wydawać na pierwszy rzut oka. Aby zasymulować działanie suwaka głośności, należy utworzyć nową stronę, która będzie korzystać z kontrolki `SliderExtender`. Suwak powinien mieć orientację poziomą oraz zakres wartości od *0* do *10*. Strona docelowa powinna wyglądać jak na rysunku 3.18. (Wskazówka: Dokumentacja kontrolki może być niejasna. Konieczne będzie zastosowanie dwóch pól tekstowych — suwak blokuje wyświetlanie jednego z nich, dlatego potrzebne będzie drugie pole tekstowe, tak zwana *kontrolka wiążąca*, w której widoczna będzie aktualna wartość suwaka).



Rysunek 3.18. Ukończona strona z ćwiczenia 3.3