PROGRAMMER TO PROGRAMMER™

Doborowi autorzy – wyśmienita książka.
Wszystko, co powinieneś wiedzieć o ASP.NET!

- Jakie kontrolki serwerowe udostępnia ASP.NET 4?
- Jak monitorować stan aplikacji ASP.NET?
- Jak zapewnić najwyższą wydajność aplikacji?

Zaawansowane programowanie ASP.NET 4 z wykorzystaniem C# i VB

Bill Evjen, Scott Hanselman, Devin Rader

Helion

ASP.NET 4 z wykorzystaniem C# i VB. Zaawansowane programowanie

Autor: Bill Evjen, Scott Hanselman, Devin Rader
Tłumaczenie: Wojciech Moch, Tomasz Walczak
ISBN: 978-83-246-2846-9
Tytuł oryginału: Professional ASP.NET 4 in C# and VB
Format: 172×245, stron: 1792

Doborowi autorzy – wyśmienita książka. Wszystko, co powinieneś wiedzieć o ASP.NET!

- Jakie kontrolki serwerowe udostępnia ASP.NET 4?
- Jak monitorować stan aplikacji ASP.NET?
- Jak zapewnić najwyższą wydajność aplikacji?

Platforma ASP.NET to główny konkurent języka Java w zakresie tworzenia aplikacji internetowych oraz dynamicznych stron internetowych. Każda jej wersja dostarcza wiele interesujących ulepszeń, a wśród nich te najważniejsze – pozwalające na zdjęcie z programisty obowiązku pisania dużych ilości nudnego kodu, bez którego jeszcze niedawno aplikacja nie mogłaby istnieć.

Niniejsza książka została napisana przez grupę wyjątkowych autorów. Bill Evjen to najaktywniejszy promotor technologii .NET, Scott Hanselman to główny menedżer w jednym z działów firmy Microsoft, prowadzący szkolenia dotyczące ASP.NET na całym świecie, a Devin Rader to pracownik firmy Infragistics. Ta doborowa trójka stworzyła świetny podręcznik, w całości poświęconą ASP.NET. Znajdziesz w niej informacje na temat stosowania języków VisualBasic oraz C# do tworzenia dynamicznych stron. Dowiesz się, jak wykorzystać kontrolki serwerowe, budować aplikacje z wykorzystaniem wzorca MVC oraz tchnąć życie w strony za pomocą technologii AJAX. Te i wiele innych bezcennych informacji, porad i wskazówek odkryjesz dzięki tej wyjątkowej książce!

- Produktywność programistów
- Infrastruktura i tworzenie aplikacji ASP.NET
- Środowisko ADO.NET
- Kompilacja aplikacji ASP.NET
- Monitorowanie stanu aplikacji
- Kontrolki serwerowe
- Wykorzystanie stron wzorcowych
- Kompozycje i skórki
- Wykonywanie zapytań z wykorzystaniem LINQ
- Wykorzystanie formatu XML
- Bezpieczeństwo aplikacji ASP.NET
- Instrumentacja

Sprawdź, jaka moc drzemie w platformie ASP.NET!

Helion SA
ul. Kościuszki 1c
44-100 Gliwice
tel. 32 230 98 63
e-mail: helion@helion.pl

Spis treści

5

Praca ze stronami wzorcowymi

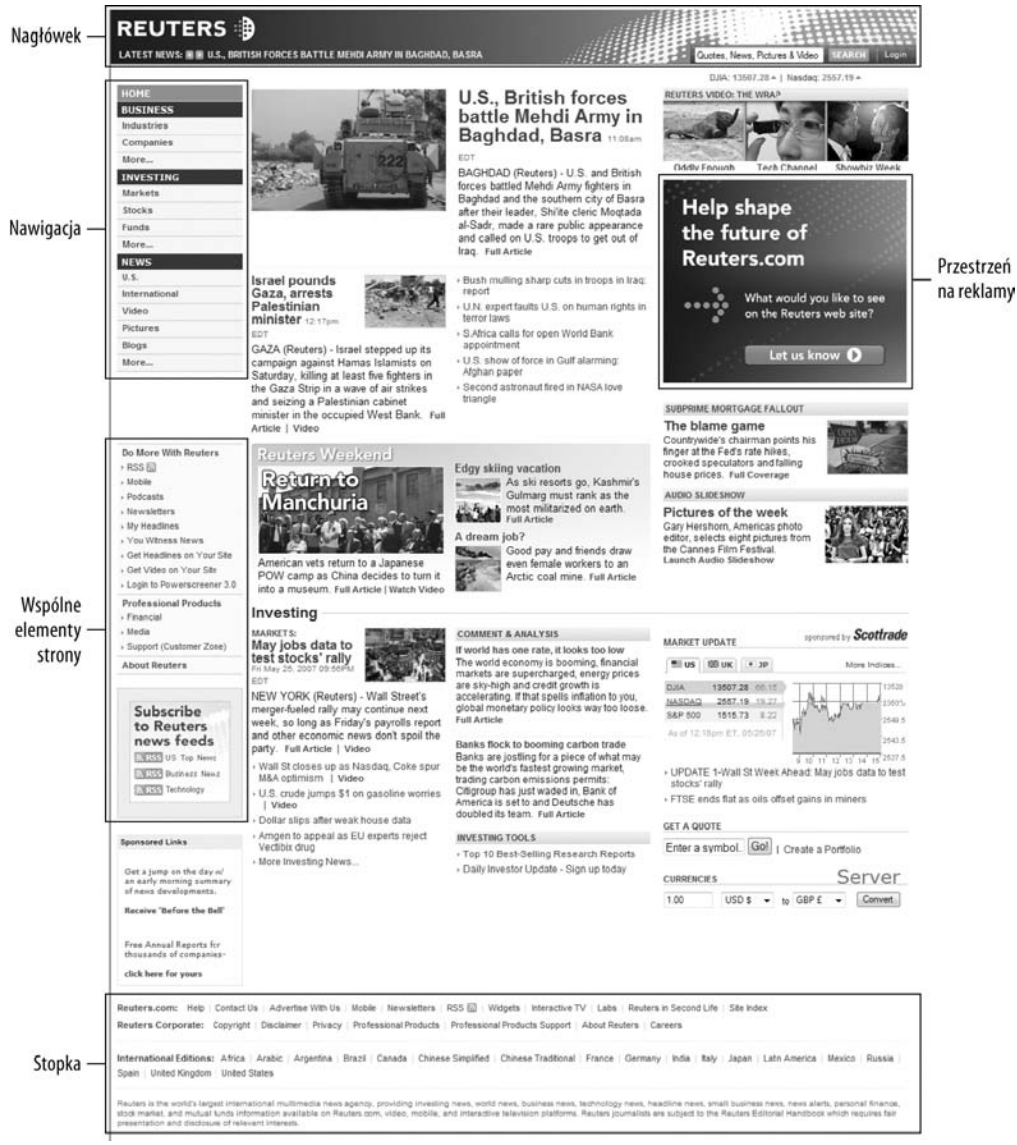
ZAWARTOŚĆ ROZDZIAŁU:

- Tworzenie stron wzorcowych i stron z zawartością
- Stosowanie stron wzorcowych do określania domyślnych elementów
- Programowe przypisywanie stron wzorcowych
- Zagnieżdżanie stron wzorcowych
- Strony wzorcowe dla różnych przeglądarek
- Używanie stron wzorcowych z technologią ASP.NET AJAX

Dziedziczenie wizualne jest wspaniałym usprawnieniem, z którego można korzystać w ASP.NET przy budowaniu stron WWW. Taka możliwość została wprowadzona do ASP.NET w wersji 2.0. W efekcie otrzymujemy szansę utworzenia pojedynczej strony wzorcowej, która może być potem użyta jako podstawa dla dowolnej ilości zwykłych stron z zawartością w aplikacjach ASP.NET. Takie wzorce, zwane **stronami wzorcowymi**, zwiększają produktywność, sprawiając, że aplikacje tworzy się łatwiej. Prostsze jest także późniejsze zarządzanie takimi aplikacjami. Visual Studio 2010 daje nam pełne wsparcie przy tworzeniu stron wzorcowych za pomocą projektanta formularzy. Możliwości projektowe są teraz większe niż kiedykolwiek wcześniej. W niniejszym rozdziale pokazane zostanie, jak wykorzystać strony wzorcowe w aplikacjach w najlepszy możliwy sposób. Zaczniemy jednak od omówienia zalet wykorzystania stron wzorcowych.

Do czego potrzebne są strony wzorcowe?

W dzisiejszych czasach większość portali internetowych posiada wspólne elementy, które wykorzystywane są przez większość stron aplikacji. Przyjrzyjmy się dla przykładu stronie głównej serwisu Reuters News (pod adresem *www.reuters.com*). Z łatwością można wyróżnić wspólne elementy używane na całym portalu. Oznaczono je na rysunku 5.1.



Rysunek 5.1

Na pokazanym zrzucie ekranu warto zwrócić uwagę między innymi na sekcję nagłówka, sekcję nawigacji i sekcję stopki. Większość stron aplikacji korzysta z tych samych elementów. Przed erą stron wzorcowych także istniały sposoby umieszczania tych samych elementów na wszystkich stronach. W większości przypadków przy stosowaniu różnych rozwiązań pojawiały się problemy.

Niektórzy programiści zwyczajnie kopiowali i wklejali kod wspólnych sekcji do każdej strony, która go potrzebowała. To działało, ale wymagało dużego nakładu pracy. Istnieje jednak problem poważniejszy. Gdy potrzebna była zmiana w jednej z tych sekcji, wtedy programista zmuszony był do przejrzenia wszystkich stron portalu i wprowadzenia do każdej ze stron podobnych zmian. To nie jest oczywiście przyjemne. Istnieją lepsze sposoby wykorzystania czasu.

W czasach klasycznego języka ASP (ang. *Active Server Pages*) popularnym rozwiązaniem było umieszczenie wspólnych sekcji w **pliku dołączanym**. Taki plik mógł być potem umieszczany na stronie w taki oto sposób:

```
<!-- #include virtual="/myIncludes/header.asp" -->
```

Problem podczas korzystania z plików dołączanych polegał na tym, że należało brać pod uwagę otwarte znaczniki języka HTML z nagłówkowego pliku dołączanego. Znaczniki musiały być zamknięte w głównym dokumencie lub w kolejnym pliku dołączanym (ze stopką). Bardzo trudno było w takich sytuacjach utrzymać porządek znaczników języka HTML, zwłaszcza wtedy, gdy nad jednym projektem pracowało kilka osób. Strony internetowe czasami były pokazywane w dziwny sposób. Gdzieś bowiem po drodze mógł znaleźć się niewłaściwy lub nieistniejący znacznik zamykający lub otwierający. Trudno było także pracować z plikami dołączanymi w projektancie formularzy. Użycie plików dołączanych powodowało, że projektant nie był w stanie wyświetlić strony tak, jak powinna być pokazywana w przeglądarce. Programista kończył pisanie strony w sekcjach i *miał nadzieję*, że wszystkie kawałki w jakiś sposób poskładają się zgodnie z planem. Wiele godzin było marnowane na „ściganie tabel" otwartych w pliku dołączanym i prawdopodobnie zamkniętych później!

Wraz z wprowadzeniem ASP.NET 1.0 w 2000 roku programiści zaczęli używać **kontrolek użytkownika** do ukrywania powtarzających się sekcji stron WWW. Można było na przykład stworzyć stronę, która zawierała nagłówek, panel nawigacji i stopkę, przeciągając i upuszczając te sekcje kodu na każdą stronę, która tych elementów potrzebowała.

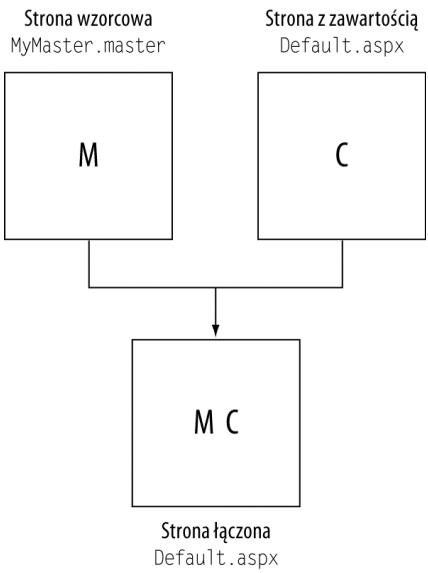
Ta technika działała, ale także powodowała pewne problemy. Przed pojawieniem się środowiska Visual Studio 2005 oraz technologii ASP.NET 2.0 kontrolki użytkownika prowadziły do problemu omówionego przy okazji plików dołączanych. Podczas pracy w widoku projektanta stron w środowisku Visual Studio .NET 2002 i 2003 wspólne obszary były wyświetlane w postaci szarych prostokątów. To utrudniało tworzenie strony. Nie dało się zobaczyć, jak tworzona właśnie strona wygląda, dopóki projekt nie został skompilowany i uruchomiony w przeglądarce. Kontrolki użytkownika nie były też wolne od drugiego problemu dotyczącego plików dołączanych — należało właściwie sparować otwierające i zamykające znaczniki języka HTML umieszczone w dwóch różnych plikach. Generalnie kontrolki użytkownika są lepsze niż pliki dołączane, ale to także nie jest doskonały sposób na rozwiązanie omawianych problemów w aplikacjach. Można zauważyć, że w środowisku Visual Studio poprawiono kilka problemów związanych z renderowaniem zawartości kontrolek użytkownika w oknie projektowym. Kontrolki użytkownika są dobre wtedy, gdy trzeba umieścić na stronach jakieś niewielkie sekcje. Problem w dalszym ciągu nie jest rozwiązany, gdy zachodzi potrzeba zastosowania ich jako większych wzorców.

W świetle problemów pojawiających się przy plikach dołączanych i kontrolkach użytkownika grupa odpowiedzialna za rozwój ASP.NET wpadła na pomysł **stron wzorcowych** — to nowy sposób stosowania wzorców w aplikacjach. *Zmieniono* dzięki temu sposób, w jaki programiści atakowali ten problem. Strony wzorcowe umieszczone są *poza* tworzonymi stronami. Jest to przeciwieństwo kontrolek użytkownika, które umieszczane były na stronie i były wielokrotnie powielane. Strony wzorcowe pozwalają oddzielić wspólne obszary, które umieszczane są na każdej stronie, i obszary, które są unikatowe dla każdej ze stron. Wkrótce pokażemy, że praca ze stronami wzorcowymi jest łatwa i przyjemna. W kolejnym podpunkcie omówione zostaną podstawowe elementy pracy ze stronami wzorcowymi ASP.NET.

Podstawy stron wzorcowych

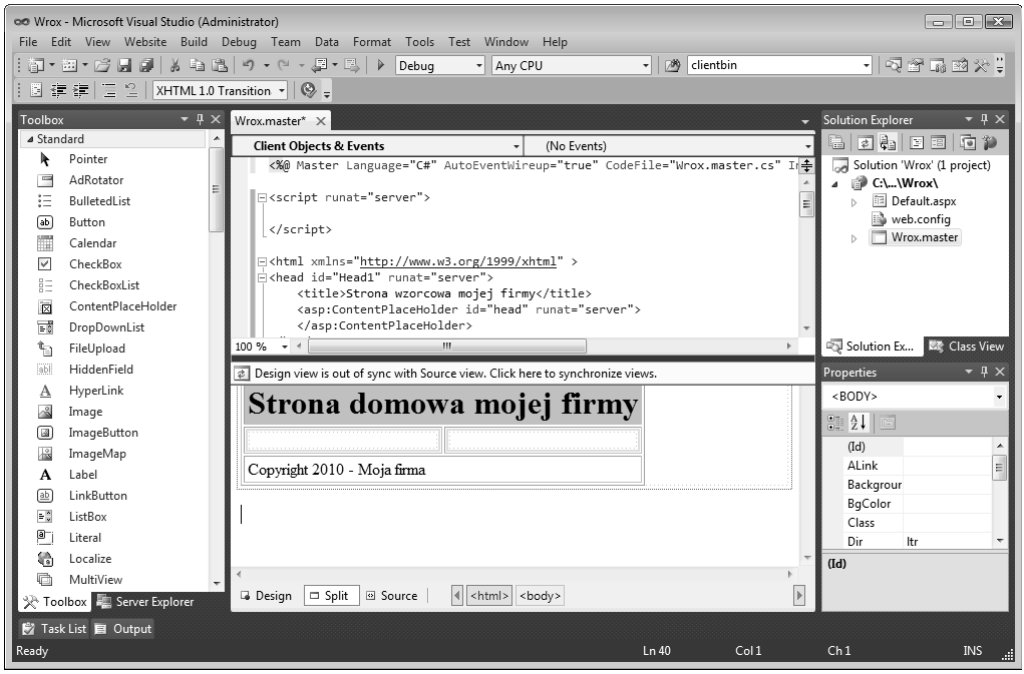
Strony wzorcowe w łatwy sposób udostępniają pewien wzorzec, który może być stosowany przez dowolną ilość stron ASP.NET w aplikacji. Pracę ze stronami wzorcowymi rozpoczyna się od utworzenia pliku wzorcowego, który będzie wskazywany przez **podstronę**, zwaną również **stroną z zawartością**. Strony wzorcowe mają rozszerzenie *.master*, podczas gdy strony z zawartością używają znanego już rozszerzenia *.aspx*. Strony z zawartością jako takie deklaruje się wewnątrz dyrektywy `Page`.

W pliku *.master*, który jest stroną wzorcową, można umieścić praktycznie wszystko. Może to być nagłówek, panel nawigacyjny lub stopka, które wykorzystywane są w całej aplikacji. Strona z zawartością obejmuje wtedy wszystkie elementy składające się na jej treść, ale bez elementów umieszczonych na stronie wzorcowej. W czasie wykonywania aplikacji silnik ASP.NET łączy te elementy dla użytkownika w jedną stronę. Na rysunku 5.2 pokazano diagram, który przedstawia zasadę działania tego mechanizmu.



Rysunek 5.2

Jedną z przyjemniejszych rzeczy w trakcie pracy ze stronami wzorcowymi jest to, że podczas tworzenia stron z zawartością w IDE można zobaczyć podgląd wzorca. W związku z tym, że podczas pracy widoczna jest cała strona, znacznie łatwiej można ją rozwijać. Podczas pracy ze stroną z zawartością wszystkie elementy pochodzące ze wzorca są wyszarzone i nie można ich modyfikować. Elementy, które można edytować, są wyróżnione. Te obszary robocze, zwane **obszarami zawartości**, są zdefiniowane w samej stronie wzorcowej. Na stronie wzorcowej określa się obszary strony, które mogą być używane przez strony z zawartością. Na stronie wzorcowej, jeżeli zachodzi taka potrzeba, może znaleźć się więcej takich obszarów zawartości. Na rysunku 5.3 pokazano stronę wzorcową z kilkoma obszarami zawartości.



Rysunek 5.3

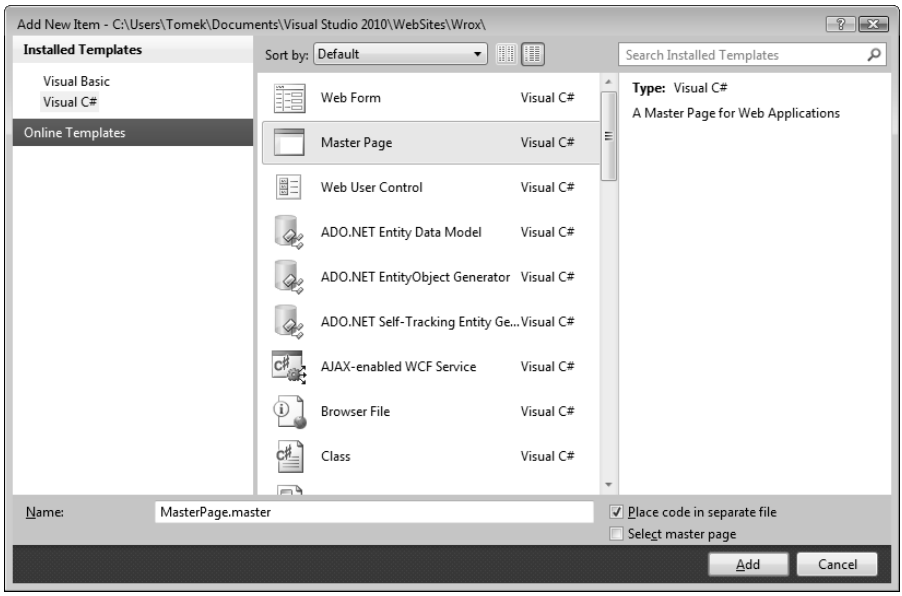
Patrząc na zrzut ekranu zaprezentowany na rysunku 5.3, można zauważyć dwa zdefiniowane obszary — obszary zawartości. Obszar zawartości wyróżniany jest w widoku *Design* za pomocą jasnego prostokąta z ramką w postaci punktów. W ten sposób reprezentowana jest kontrolka `ContentPlaceHolder`. Oprócz tego, jeżeli najedziemy kursorem myszy nad obszar zawartości, nad kontrolką pojawi się jej nazwa (półprzezroczysta). Moment najechania wskaźnikiem myszy nad kontrolkę jest także ujęty na rysunku 5.3.

Dla firm i instytucji wykorzystanie stron wzorcowych to idealne rozwiązanie. Technologia ta doskonale odpowiada typowym wymaganiom biznesowym. Wiele firm tworzy wszystkie swoje strony intranetowe tak, że ich wygląd i obsługa są podobne. Mogą one teraz udostępnić działom firmy stronę wzorcową *.master* do stworzenia strony wydziału w intranecie. Taki proces zdecydowanie ułatwia zachowanie jednolitego wyglądu i podobnej obsługi strony w całym intranecie.

Pisanie kodu stron wzorcowych

Przyjrzyjmy się teraz poszczególnym etapom budowania strony wzorcowej pokazanej wcześniej na rysunku 5.3. Stronę można stworzyć w dowolnym edytorze tekstowym, na przykład w Notatniku lub w narzędziu Visual Web Developer Express Edition, ale można także użyć nowego środowiska Visual Studio 2010. W niniejszym rozdziale pokażemy, jak to się robi właśnie w środowisku Visual Studio 2010.

Strony wzorcowe dodawane są do projektów w taki sam sposób jak zwykłe strony *.aspx* — wystarczy podczas dodawania pliku do aplikacji wybrać opcję *Master Page*. Pokazano to na rysunku 5.4.



Rysunek 5.4

Dodawanie stron wzorcowych podobne jest do tworzenia zwykłych stron *.aspx*. W oknie dialogowym *Add New Item* znajduje się element, który pozwala utworzyć strony wzorcowe. Można skorzystać z modelu code-inline lub umieścić kod dla strony w oddzielnym pliku. Jeżeli nie umieścimy kodu serwera w oddzielnym pliku, będzie to oznaczało, że podczas tworzenia strony wzorcowej wykorzystywany jest model code-inline. Pozwala to utworzyć pojedynczy plik *.master*. Wybranie opcji *Place code in separate file* oznacza, że podczas tworzenia strony wykorzystywany jest model code-behind. Zaznaczenie pola wyboru *Place code in separate file* pozwala utworzyć pojedynczą stronę *.master* oraz skojarzony z nią plik *.master.vb* lub *.master.cs*. Istnieje także możliwość osadzenia strony wzorcowej w innej stronie wzorcowej poprzez wybranie opcji *Select master page*. Pokazane jest to w dalszej części rozdziału.

Przykładowa strona wzorcowa, która wykorzystuje model code-inline, pokazana jest na listingu 5.1.

Listing 5.1. Przykładowa strona wzorcowa

```
<%@ Master Language="VB" %>

<script runat="server">

</script>

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Strona wzorcowa mojej firmy</title>
    <asp:ContentPlaceHolder id="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <form id="form1" runat="server">
        <table cellpadding="3" border="1">
            <tr style="background:silver">
```

```
                <td colspan="2">
                    <h1>Strona domowa mojej firmy</h1>
                </td>
            </tr>
            <tr>
                <td>
                    <asp:ContentPlaceHolder ID="ContentPlaceHolder1"
                     runat="server">
                    </asp:ContentPlaceHolder>
                </td>
                <td>
                    <asp:ContentPlaceHolder ID="ContentPlaceHolder2"
                     runat="server">
                    </asp:ContentPlaceHolder>
                </td>
            </tr>
            <tr>
                <td colspan="2">
                    Copyright 2010 - Moja firma
                </td>
            </tr>
        </table>
    </form>
</body>
</html>
```

Jest to prosta strona wzorcowa. Wspaniałe przy tworzeniu stron wzorcowych w Visual Studio 2010 jest to, że można pracować z nią w widoku kodu, ale można również przełączyć się do widoku projektanta i utworzyć tę stronę tak, jak tworzy się każdą inną stronę ASP.NET.

Rozpoczniemy od analizy kodu strony wzorcowej. Pierwszy wiersz to dyrektywa:

```
<%@ Master Language="VB" %>
```

Zamiast używać dyrektywy `Page` tak, jak w zwykłych stronach *.aspx*, dla stron wzorcowych używa się dyrektywy `Master`. Pokazana strona wzorcowa korzysta tylko z jednego atrybutu, `Language`. Wartością atrybutu `Language` jest w tym przypadku `VB`, ale oczywiście można użyć `C#`, jeżeli tworzy się strony wzorcowe z wykorzystaniem tego języka.

Pozostała część kodu strony wzorcowej wygląda tak samo jak każda inna strona *.aspx*. Można korzystać z kontrolek serwerowych, zwykłego kodu w HTML-u i tekstu, obrazków, zdarzeń oraz wszystkich innych składników wykorzystywanych na stronach *.aspx*. Oznacza to, że strona wzorcowa może obsługiwać zdarzenie `Page_Load` oraz inne potrzebne zdarzenia.

W kodzie pokazanym na listingu 5.1 warto zwrócić uwagę na wykorzystanie nowej kontrolki serwerowej — kontrolki `<asp:ContentPlaceHolder>`. Kontrolka definiuje obszary strony wzorcowej, w którym strona z zawartością może umieszczać swoje elementy:

```
<tr>
    <td>
        <asp:ContentPlaceHolder ID="ContentPlaceHolder1"
         runat="server">
        </asp:ContentPlaceHolder>
    </td>
    <td>
        <asp:ContentPlaceHolder ID="ContentPlaceHolder2"
```

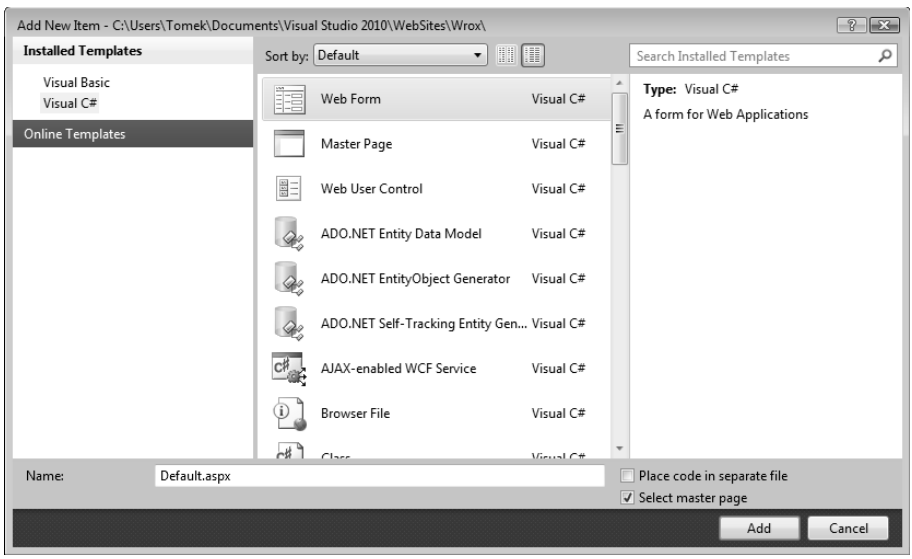
```
        runat="server">
      </asp:ContentPlaceHolder>
    </td>
</tr>
```

W pokazanej stronie wzorcowej istnieją dwa zdefiniowane obszary, gdzie można umieszczać zawartość. Strona zawiera oprócz tego obszar nagłówka i obszar stopki. Zdefiniowane są także dwa obszary, w których strony dziedziczące mogą wstawiać swoje własne elementy. Przyjrzyjmy się teraz, w jaki sposób strona z zawartością wykorzystuje tę stronę wzorcową.

Pisanie kodu strony z zawartością

Gdy w aplikacji umieszczono już stronę wzorcową, to można użyć jej w stronach z zawartością w danej aplikacji. Kliknijmy prawym przyciskiem myszy w oknie *Solution Explorer* i wybierzmy opcję *Add New Item* w celu utworzenia w aplikacji nowej strony z zawartością.

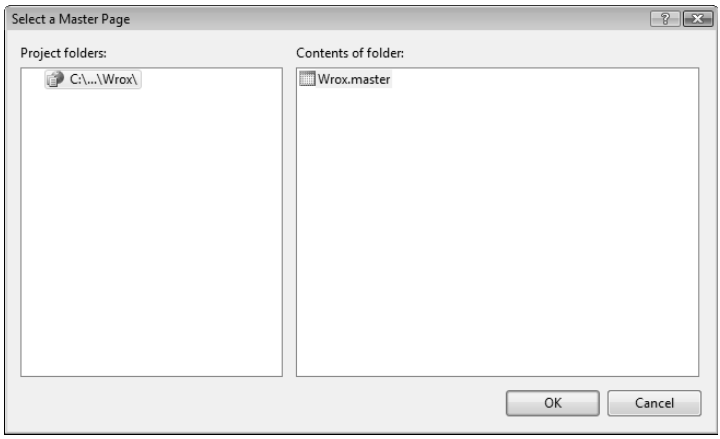
Aby utworzyć stronę z zawartością lub stronę, która wykorzystuje stronę wzorcową, w oknie dialogowym *Add New Item* spośród wielu opcji należy wybrać *Web Form* (zobacz rysunek 5.5). Zamiast jednak tworzyć typową stronę, trzeba zaznaczyć pole *Select master page*. Dzięki temu można połączyć tworzony formularz z wybraną stroną wzorcową.



Rysunek 5.5

Po nazwaniu strony z zawartością i kliknięciu przycisku *Add* w oknie dialogowym *Add New Item* pojawi się okno dialogowe *Select a Master Page* pokazane na rysunku 5.6.

Okno dialogowe pozwala wybrać stronę wzorcową, na podstawie której tworzona będzie strona z zawartością. Wyboru można dokonać spośród wszystkich stron wzorcowych dostępnych w aplikacji. W tym przypadku wybierzmy stronę wzorcową pokazaną na listingu 5.1 i kliknijmy *OK*. W ten sposób tworzona jest nowa strona z zawartością. Nowa strona jest zwykłą stroną *.aspx*. Zawiera tylko kilka wierszy kodu. Pokazano to na listingu 5.2.



Rysunek 5.6

Listing 5.2. Utworzona strona z zawartością

VB

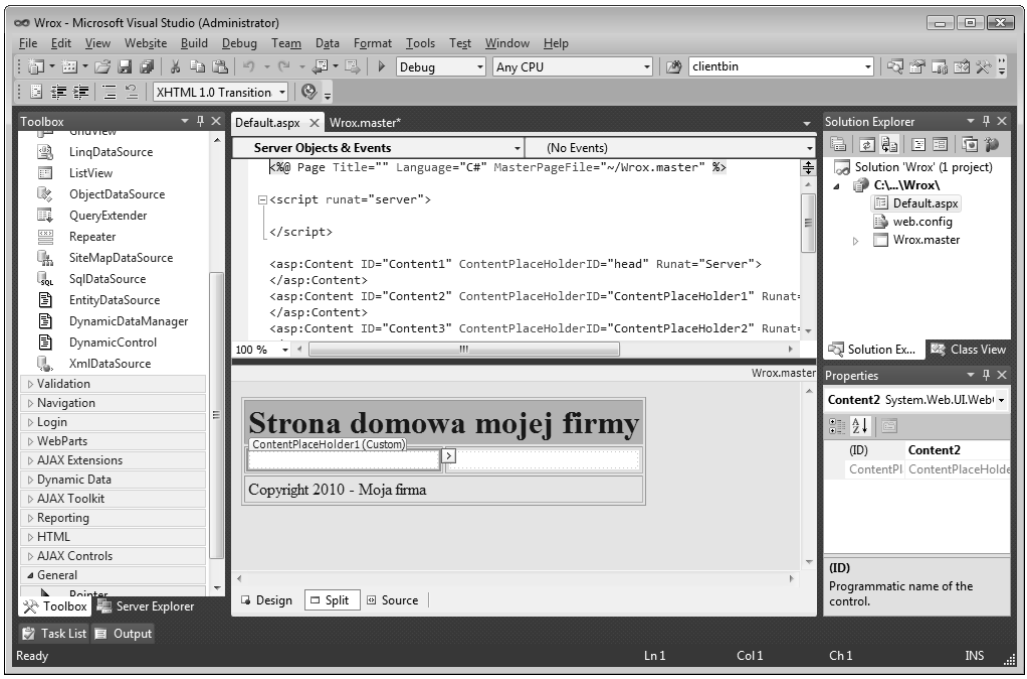
```
<%@ Page Language="VB" MasterPageFile="~/Wrox.master" Title="" %>

<script runat="server">

</script>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"
 Runat="Server">
</asp:Content>
```

Strona z zawartością nie różni się bardzo od typowej strony *.aspx*, którą mieliśmy okazję tworzyć wcześniej. Dużą różnicą jest obecność w dyrektywie `Page` atrybutu `MasterPageFile`. Dzięki wykorzystaniu tego atrybutu można wskazać, że ta konkretna strona *.aspx* tworzona jest na podstawie innej strony. Położenie strony wzorcowej w aplikacji określone jest przez wartość atrybutu `MasterPageFile`.

Kolejną dużą różnicą jest to, że strona nie zawiera żadnych znaczników `<form id="form1" runat="server">` ani żadnych otwierających i zamykających znaczników języka HTML, które w normalnej stronie *.aspx* powinny być obecne.

Strona z zawartością może się wydawać prosta, ale jeżeli przełączymy się na widok projektanta w Visual Studio 2010, wtedy będzie można dostrzec możliwości, jakie dają strony wzorcowe. Dzięki dziedziczeniu wizualnemu otrzymujemy rezultat pokazany na rysunku 5.7.

Na pokazanym zrzucie ekranu można zauważyć, że aby wykorzystać stronę wzorcową z pliku *Wrox.master* poprzez dziedziczenie wizualne, wystarczy dodać atrybut `MasterPageFile` do dyrektywy `Page`. Korzystając z widoku projektanta w środowisku Visual Studio, można także zobaczyć, która strona wzorcowa jest obecnie używana. Nazwa aktywnej strony wzorcowej pokazywana jest w prawym górnym rogu widoku *Design*. Jeżeli spróbujemy kliknąć na wyszarzonym obszarze reprezentującym część dziedziczoną ze strony wzorcowej, wtedy będzie można zobaczyć, że kursor się zmienia. Sygnalizuje to, że wykonanie tej operacji jest niemożliwe. Pokazano to na rysunku 5.8 (kursor znajduje się nad słowem „firmy”).



Rysunek 5.7

Rysunek 5.8

Wszystkie wspólne obszary zdefiniowane w stronie wzorcowej są wyszarzone. Obszary zawartości oznaczone w stronie wzorcowej za pomocą kontrolki serwerowej `<asp:ContentPlaceHolder>` są pokazane wyraźnie i umożliwiają wstawienie dodatkowych elementów. Do tych obszarów zawartości można wstawić dowolne elementy. Odbywa się to tak samo, jak w przypadku zwykłych stron *.aspx*. Przykład wykorzystania strony wzorcowej w stronie z zawartością przedstawiono na listingu 5.3.

Listing 5.3. Strona z zawartością korzystająca ze strony wzorcowej Wrox.master

VB

```
<%@ Page Language="VB" MasterPageFile="~/Wrox.master" %>

<script runat="server">
   Protected Sub Button1_Click(ByVal sender As Object,
      ByVal e As System.EventArgs)

         Label1.Text = "Witaj, " & TextBox1.Text & "!"
   End Sub
```

```
</script>

<asp:Content ID="Content1" ContentPlaceHolderId="ContentPlaceHolder1"
 runat="server">
    <b>Wpisz swoje imię:</b><br />
    <asp:Textbox ID="TextBox1" runat="server" />
    <br />
    <br />
    <asp:Button ID="Button1" runat="server" Text="Zatwierdź"
     OnClick="Button1_Click" /><br />
    <br />
    <asp:Label ID="Label1" runat="server" Font-Bold="True" />
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderId="ContentPlaceHolder2"
 runat="server">
        <asp:Image ID="Image1" runat="server" ImageUrl="wrox.gif" />
</asp:Content>
```

C#

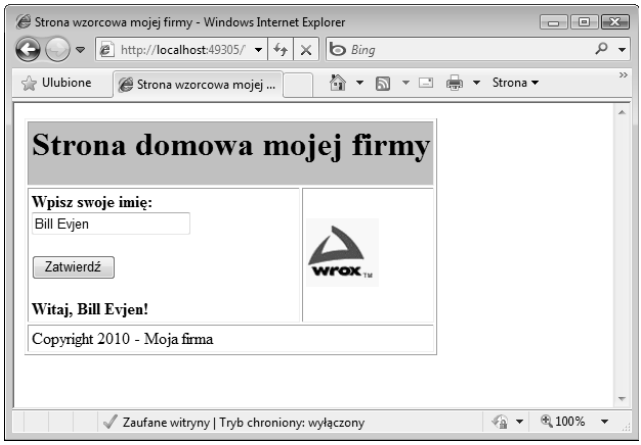
```
<%@ Page Language="C#" MasterPageFile="~/Wrox.master" %>

<script runat="server">
    protected void Button1_Click(object sender, System.EventArgs e)
    {
        Label1.Text = "Witaj, " + TextBox1.Text + "!";
    }
</script>
```

Od razu można zauważyć różnice. Jak już wspomnieliśmy, strona nie posiada znacznika `<form id="form1" runat="server">` ani żadnego otwierającego lub zamykającego znacznika `<html>`. Te znaczniki nie są umieszczone na tej stronie, ponieważ znajdują się na stronie wzorcowej. Warto także zwrócić uwagę na nową kontrolkę serwerową `<asp:Content>`.

```
<asp:Content ID="Content1" ContentPlaceHolderId="ContentPlaceHolder1"
 runat="server">
    ...
</asp:Content>
```

Kontrolka serwerowa `<asp:Content>` jest zdefiniowanym obszarem zawartości, który pokrywa się z określoną kontrolką serwerową `<asp:ContentPlaceHolder>` strony wzorcowej. W pokazanym przykładzie można zauważyć, że kontrolka serwerowa `<asp:Content>` łączy się z kontrolką serwerową `<asp:ContentPlaceHolder>` strony wzorcowej posiadającą wartość `ID` równą `ContentPlaceHolder1`. Na stronie z zawartością nie trzeba się martwić ustawianiem położenia zawartości, ponieważ obszar zawsze jest definiowany przez stronę wzorcową. W związku z tym jedynym zmartwieniem jest umieszczenie właściwych elementów wewnątrz dostępnych sekcji. Pozostałą część pracy należy pozostawić stronie wzorcowej.

Tak samo jak w przypadku pracy ze zwykłymi stronami *.aspx*, tak i w stronach z zawartością można tworzyć dowolne procedury obsługi zdarzeń. W tym przypadku wykorzystujemy tylko jedną procedurę obsługi zdarzenia — wywoływana jest w momencie kliknięcia przycisku zatwierdzającego formularz. Utworzona strona *.aspx*, która korzysta ze strony wzorcowej oraz kilku wstawionych elementów, pokazana jest na rysunku 5.9.



Rysunek 5.9

Łączenie różnych typów stron i języków

Jest jedna interesująca rzecz: podczas korzystania ze stron wzorcowych nie ma obowiązku korzystania z jednego określonego modelu pisania kodu (code-inline lub code-behind), nie ma też obowiązku stosowania jednego określonego języka. Można śmiało łączyć te elementy w aplikacji, ponieważ wszystko będzie działało doskonale.

Można zatem wykorzystać wcześniej utworzoną stronę wzorcową, jeśli się wie, że była utworzona z wykorzystaniem modelu code-inline, a następnie utworzyć stronę z zawartością za pomocą modelu code-behind. Na listingu 5.4 pokazano stronę, która została utworzona w modelu code-behind.

Listing 5.4. Strona z zawartością korzystająca z modelu code-behind

.aspx (VB)

```
<%@ Page Language="VB" MasterPageFile="~/Wrox.master" AutoEventWireup="false"
    CodeFile="MyContentPage.aspx.vb" Inherits="MyContentPage" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderId="ContentPlaceHolder1"
 runat="server">
    <b>Wpisz swoje imię:</b><br />
    <asp:Textbox ID="TextBox1" runat="server" />
    <br />
    <br />
    <asp:Button ID="Button1" runat="server" Text="Zatwierdź" /><br />
    <br />
    <asp:Label ID="Label1" runat="server" Font-Bold="True" />
</asp:Content>

<asp:Content ID="Content3" ContentPlaceHolderId="ContentPlaceHolder2"
 runat="server">
       <asp:Image ID="Image1" runat="server" ImageUrl="wrox.gif" />
</asp:Content>
```

VB (plik code-behind)

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

    Protected Sub Button1_Click(ByVal sender As Object, _
       ByVal e As System.EventArgs) Handles Button1.Click

        Label1.Text = "Witaj, " & TextBox1.Text & "!"
    End Sub

End Class
```

C# (plik code-behind)

```
public partial class MyContentPage : System.Web.UI.Page
{
    protected void Button1_Click(object sender, EventArgs e)
    {
        Label1.Text = "Witaj, " + TextBox1.Text + "!";
    }
}
```

Pomimo że strona wzorcowa korzysta z modelu code-inline, można z łatwością tworzyć strony z zawartością (na przykład takie, jak ta pokazana na listingu 5.4), które wykorzystują model code-behind. Strony w dalszym ciągu będą działały doskonale.

Samo łączenie dwóch modeli pisania kodu to nie wszystko, co można zrobić przy stosowaniu stron wzorcowych. Można także mieszać języki programowania używane podczas pisania stron wzorcowych i stron z właściwą zawartością. To, że strona wzorcowa utworzona jest za pomocą języka C#, nie oznacza wcale, że język C# musi być używany na wszystkich stronach, które tej strony wzorcowej używają. Strony z zawartością mogą być tworzone także w języku Visual Basic. Jako dobry przykład stwórzmy stronę wzorcową w języku C#, która korzysta ze zdarzenia `Page_Load`, a następnie dodajmy do tego stronę z zawartością, która została napisana w języku Visual Basic. Po zrobieniu wszystkiego jak należy spróbujmy uruchomić stronę. Działa ona doskonale. Oznacza to, że nawet wtedy, gdy strona wzorcowa napisana jest w jednym z dostępnych języków .NET, grupy programistyczne korzystające podczas tworzenia aplikacji ze strony wzorcowej mogą używać takiego języka .NET, jakiego tylko chcą. Warto pokochać otwartość, którą oferuje nam platforma .NET.

Określanie, której strony wzorcowej użyć

Pokazaliśmy już, że wskazanie wykorzystywanej strony wzorcowej z poziomu strony jest łatwe. W dyrektywie `Page` strony z zawartością wystarczy zastosować atrybut `MasterPageFile`:

```
<%@ Page Language="VB" MasterPageFile="~/Wrox.master" %>
```

Oprócz wskazywania używanej strony wzorcowej na poziomie strony istnieje drugi sposób wykonania tego samego zadania. Stronę wzorcową można określić za pomocą pliku konfiguracyjnego aplikacji *web.config*. Pokazano to na listingu 5.5.

Listing 5.5. Wskazywanie strony wzorcowej w pliku web.config

```
<configuration>
   <system.web>
      <pages masterPageFile="~/Wrox.master" />
   </system.web>
</configuration>
```

Wskazanie strony wzorcowej w pliku *web.config* spowoduje, że każda strona z zawartością tworzona w aplikacji będzie dziedziczyła po określonej stronie wzorcowej. Jeżeli stronę zadeklarujemy w pliku *web.config*, wtedy będzie można tworzyć dowolną ilość stron z zawartością i wszystkie one będą korzystały z podanej strony wzorcowej. Po wskazaniu strony w ten sposób dyrektywę `Page` można skonstruować w następujący sposób:

```
<%@ Page Language="VB" %>
```

Można w łatwy sposób przesłonić to ustawienie dla całej aplikacji, deklarując na stronie z zawartością inną stronę wzorcową:

```
<%@ Page Language="VB" MasterPageFile="~/MyOtherCompany.master" %>
```

Wskazanie strony wzorcowej w pliku *web.config* nie oznacza, że *wszystkie* strony *.aspx* muszą korzystać z tej strony wzorcowej. Jeżeli stworzymy i uruchomimy zwykłą stronę Web Form, środowisko ASP.NET będzie wiedziało, że nie jest to strona z zawartością. Strona ta zostanie przetworzona tak samo, jak zwykła strona *.aspx*.

Jeżeli zachodzi potrzeba zastosowania strony wzorcowej tylko do określonego zestawu stron (na przykład do stron umieszczonych w określonym katalogu aplikacji), wtedy można użyć elementu `<location>` w pliku *web.config*. Pokazano to na listingu 5.6.

Listing 5.6. Wskazywanie strony wzorcowej dla określonego katalogu w pliku web.config

```
<configuration>

    <location path="AdministrationArea">
        <system.web>
           <pages masterPageFile="~/WroxAdmin.master" />
        </system.web>
    </location>

</configuration>
```

Dodając w pliku *web.config* sekcję `<location>`, można wskazać, że określony katalog (`AdministrationArea`) będzie używał innego pliku strony wzorcowej. Zrealizowane jest to poprzez użycie atrybutu `path` elementu `<location>`. Wartością atrybutu `path` może być tak jak tutaj nazwa katalogu, ale może to być także określona strona — na przykład *AdminPage.aspx*.

Praca z tytułem strony

Tworząc w aplikacji strony z zawartością, należy zwrócić uwagę na jedną rzecz. Domyślnie wszystkie takie strony posiadają tytuł zadeklarowany na stronie wzorcowej. Pierwotnie używaliśmy strony wzorcowej z tytułem `Strona wzorcowa mojej firmy`. Każda strona z zawartością tworzona na podstawie tej strony wzorcowej także miała tytuł `Strona wzorcowa mojej firmy`. Można to zmienić, korzystając z atrybutu `Title` dyrektywy `@Page` na stronie z zawartością. Można także programowo zmodyfikować tytuł wybranych stron z zawartością. Aby tego doko-

nać, należy w kodzie strony z zawartością użyć obiektu `Master`. Obiekt `Master` posiada wygodną właściwość o nazwie `Title`. Wartością tej właściwości jest tytuł strony używany przez strony z zawartością. Można go zmienić w sposób pokazany na listingu 5.7.

Listing 5.7. Kod zmieniający tytuł strony z zawartością

VB
```
<%@ Page Language="VB" MasterPageFile="~/Wrox.master" %>

<script runat="server">
    Protected Sub Page_LoadComplete(ByVal sender As Object, _
       ByVal e As System.EventArgs)

        Master.Page.Title = "Strona została wygenerowana: " & _
           DateTime.Now.ToString()
    End Sub
</script>
```

C#
```
<%@ Page Language="C#" MasterPageFile="~/Wrox.master" %>

<script runat="server">
    protected void Page_LoadComplete(object sender, EventArgs e)
    {
        Master.Page.Title = "Strona została wygenerowana: " +
           DateTime.Now.ToString();
    }
</script>
```

Praca z kontrolkami i właściwościami strony wzorcowej

Pracując ze stronami wzorcowymi z poziomu stron z zawartością, można w łatwy sposób dostać się do kontrolek i właściwości udostępnionych przez stronę wzorcową. Strona wzorcowa wskazywana przez stronę z zawartością posiada właściwość `Master`. Dzięki tej właściwości można pobrać wartości kontrolek lub innych właściwości umieszczonych na samej stronie wzorcowej.

Aby zobaczyć przykład tego mechanizmu, stwórzmy na stronie wzorcowej GUID (unikatowy identyfikator), który będzie pobierany przez stronę z zawartością korzystającą z tej strony wzorcowej. Dla celów przykładu skorzystajmy ze strony wzorcowej pokazanej na listingu 5.1. Dodajmy jednak do niej kontrolkę serwerową `Label` oraz zdarzenie `Page_Load`. Pokazano to na listingu 5.8.

Listing 5.8. Strona wzorcowa tworząca identyfikator GUID podczas pierwszego żądania

VB
```
<%@ Master Language="VB" %>

<script runat="server">
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
        If Not Page.IsPostBack Then
            Label1.Text = System.Guid.NewGuid().ToString()
        End If
    End Sub
</script>

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
```

```
    <title>Strona wzorcowa mojej firmy</title>
    <asp:ContentPlaceHolder id="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <form id="form1" runat="server">
        <table cellpadding="3" border="1">
            <tr bgcolor="silver">
                <td colspan="2">
                    <h1>Strona domowa mojej firmy</h1>
                    <b>GUID użytkownika:  
                        <asp:Label ID="Label1" runat="server" /></b>
                </td>
            </tr>
            <tr>
                <td>
                    <asp:ContentPlaceHolder ID="ContentPlaceHolder1"
                     runat="server">
                    </asp:ContentPlaceHolder>
                </td>
                <td>
                    <asp:ContentPlaceHolder ID="ContentPlaceHolder2"
                     runat="server">
                    </asp:ContentPlaceHolder>
                </td>
            </tr>
            <tr>
                <td colspan="2">
                    Copyright 2010 - Moja firma
                </td>
            </tr>
        </table>
    </form>
</body>
</html>
```

C#

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        Label1.Text = System.Guid.NewGuid().ToString();
    }
}
```

Teraz na stronie wzorcowej znajduje się kontrolka `Label`, do której można uzyskać dostęp z poziomu strony z zawartością. Jest kilka sposobów na wykonanie tego zadania. Pierwszy z nich to wykorzystanie metody `FindControl()`, która jest udostępniana przez stronę wzorcową. Taki sposób rozwiązania problemu pokazano na listingu 5.9.

Listing 5.9. Pobieranie wartości Text kontrolki Label na stronie z zawartością

VB

```
<%@ Page Language="VB" MasterPageFile="~/wrox.master" %>

<script runat="server" language="vb">
    Protected Sub Page_LoadComplete(ByVal sender As Object, _
```

```
       ByVal e As System.EventArgs)

        Label1.Text = CType(Master.FindControl("Label1"), Label).Text
    End Sub

    Protected Sub Button1_Click(ByVal sender As Object, _
       ByVal e As System.EventArgs)

        Label2.Text = "Witaj, " & TextBox1.Text & "!"
    End Sub
</script>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderId="ContentPlaceHolder1"
 runat="server">
    <b>Twój numer GUID ze strony wzorcowej to:<br />
    <asp:Label ID="Label1" runat="server" /></b><p>
    <b>Wpisz swoje imię:</b><br />
    <asp:Textbox ID="TextBox1" runat="server" />
    <br />
    <br />
    <asp:Button ID="Button1" runat="server" Text="Zatwierdź"
     OnClick="Button1_Click" /><br />
    <br />
    <asp:Label ID="Label2" runat="server" Font-Bold="True" /></p>
</asp:Content>

<asp:Content ID="Content3" ContentPlaceHolderId="ContentPlaceHolder2"
 runat="server">
        <asp:Image ID="Image1" runat="server" ImageUrl="Wrox.gif" />
</asp:Content>
```

C#

```
<%@ Page Language="C#" MasterPageFile="~/wrox.master" %>

<script runat="server">

    protected void Page_LoadComplete(object sender, EventArgs e)
    {
        Label1.Text = (Master.FindControl("Label1") as Label).Text;
    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        Label2.Text = "<b>Witaj " + TextBox1.Text + "!</b>";
    }
</script>
```

W przykładzie pokazanym na listingu 5.8 strona wzorcowa tworzy identyfikator GUID i zapamiętuje go w postaci wartości tekstowej w kontrolce serwerowej `Label`. Sama kontrolka jest także umieszczona na stronie wzorcowej. Wartością `ID` kontrolki `Label` jest `Label1`. Strona wzorcowa generuje identyfikator GUID tylko podczas obsługi pierwszego żądania tej konkretnej strony. Od tego momentu strona generowana jest wraz z kontrolką uzupełnioną tą wartością.

Interesujące w stronie z zawartością jest to, że kod został wstawiony w procedurze obsługi zdarzenia `Page_LoadComplete`. W ten sposób można pobrać wartość identyfikatora GUID umieszczoną na stronie wzorcowej. To zdarzenie w ASP.NET zgłaszane jest zaraz po zdarzeniu `Page_Load`. Porządek zdarzeń omówiony jest w dalszej części rozdziału, tu jedynie należy wspomnieć, że zdarzenie `Page_Load` na stronie z zawartością wywoływane jest zawsze wcześniej niż zdarzenie `Page_Load` na stronie wzorcowej. Aby pobrać utworzony identyfikator GUID (powstaje w procedurze obsługi zdarzenia `Page_Load` strony wzorcowej), należy się do niego odwołać w procedurze obsługi zdarzenia, która wywoływana jest później niż zdarzenie `Page_Load` — i to właśnie w tym miejscu przydatne okazuje się zdarzenie `Page_LoadComplete`. Kontrolka `Label` uzupełniana jest wobec tego wewnątrz procedury obsługi zdarzenia `Page_LoadComplete` strony z zawartością. Należy zwrócić uwagę, że kontrolka `Label` na stronie z zawartością posiada ten sam identyfikator `ID`, co kontrolka `Label` na stronie wzorcowej. Nie ma to jednak większego znaczenia. Można je odróżnić, korzystając z właściwości `Master`.

W taki sposób można pobrać nie tylko kontrolki serwerowe umieszczone na stronie wzorcowej. Można także uzyskać dostęp do dowolnych właściwości, które strona wzorcowa udostępnia. Przyjrzyjmy się stronie wzorcowej pokazanej na listingu 5.10. Używa ona własnej właściwości do modyfikacji sekcji `<h1>` na stronie.

Listing 5.10. Strona wzorcowa, która udostępnia własną właściwość

VB

```
<%@ Master Language="VB" %>

<script runat="server">
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
        If Not Page.IsPostBack Then
            Label1.Text = Guid.NewGuid().ToString()
        End If
    End Sub

    Dim m_PageHeadingTitle As String = "Moja firma"

    Public Property PageHeadingTitle() As String
        Get
            Return m_PageHeadingTitle
        End Get
        Set(ByVal Value As String)
            m_PageHeadingTitle = Value
        End Set
    End Property
</script>

<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title>Strona wzorcowa mojej firmy</title>
    <asp:ContentPlaceHolder id="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <form id="Form1" runat="server">
        <table cellpadding="3" border="1">
            <tr bgcolor="silver">
                <td colspan="2">
                    <h1><%= PageHeadingTitle %></h1>
                    <b>GUID użytkownika:  
```

```
                        <asp:Label ID="Label1" runat="server" /></b>
                </td>
            </tr>
            <tr>
                <td>
                    <asp:ContentPlaceHolder ID="ContentPlaceHolder1"
                     runat="server">
                    </asp:ContentPlaceHolder>
                </td>
                <td>
                    <asp:ContentPlaceHolder ID="ContentPlaceHolder2"
                     runat="server">
                    </asp:ContentPlaceHolder>
                </td>
            </tr>
            <tr>
                <td colspan="2">
                    Copyright 2010 - Moja firma
                </td>
            </tr>
        </table>
    </form>
</body>
</html>
```

C#

```
<%@ Master Language="C#" %>

<script runat="server">
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!Page.IsPostBack)
        {
            Label1.Text = System.Guid.NewGuid().ToString();
        }
    }

    string m_PageHeadingTitle = "Moja firma";

    public string PageHeadingTitle
    {
        get
        {
            return m_PageHeadingTitle;
        }
        set
        {
            m_PageHeadingTitle = value;
        }
    }
</script>
```

W powyższym przykładzie pokazano stronę wzorcową, która udostępnia własną właściwość `PageHeadingTitle`. Domyślną wartością właściwości jest „Moja firma”. Wartość właściwości umieszczana jest w kodzie w HTML-u strony wzorcowej, pomiędzy znacznikami `<h1>`. Dzięki temu domyślna wartość staje się nagłówkiem używanym przez stroną wzorcową. Pomimo że strona wzorcowa posiada już wartość używaną jako tytuł, to każda strona z zawartością korzystająca ze strony wzorcowej może nadpisać nagłówek `<h1>`. Cały proces pokazany jest na listingu 5.11.

Listing 5.11. Strona z zawartością, która nadpisuje właściwość strony wzorcowej

VB
```
<%@ Page Language="VB" MasterPageFile="~/Wrox.master" %>
<%@ MasterType VirtualPath="~/Wrox.master" %>

<script runat="server">
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
        Master.PageHeadingTitle = "Moja firma – Oddział X"
    End Sub
</script>
```

C#
```
<%@ Page Language="C#" MasterPageFile="~/Wrox.master" %>
<%@ MasterType VirtualPath="~/Wrox.master" %>

<script runat="server">
    protected void Page_Load(object sender, EventArgs e)
    {
        Master.PageHeadingTitle = "Moja firma – Oddział X";
    }
</script>
```

Z poziomu strony z zawartością można przypisać do właściwości udostępnianej przez stronę wzorcową pewną wartość. Umożliwia to właściwość `Master`. Jak widać, jest to całkiem proste. Należy pamiętać, że dostęp można uzyskać nie tylko do publicznych właściwości strony wzorcowej. Równie dobrze można wykorzystać metody, które zostały umieszczone na stronie wzorcowej.

Elementem, który umożliwia zastosowanie takich rozwiązań, jest dyrektywa `MasterType`. Dyrektywa `MasterType` pozwala utworzyć referencję ze ścisłą kontrolą typów do strony wzorcowej. Dzięki temu możliwe są odwołania do właściwości strony wzorcowej poprzez właściwość `Master`.

Wcześniej pokazaliśmy sposób uzyskania dostępu do umieszczonych na stronie wzorcowej kontrolek serwerowych za pomocą metody `FindControl()`. Metoda `FindControl()` działa dobrze, ale stosuje mechanizm późnego wiązania i w związku z tym jej wywołanie może się nie powieść, jeżeli kontrolka została usunięta ze znaczników strony. Należy zatem stosować technikę programowania defensywnego i zawsze sprawdzać, czy wartość zwrócona przez metodę `FindControl()` nie jest równa `null`. Korzystając z pokazanych mechanizmów (użycie publicznych właściwości zaprezentowano na listingu 5.10), można zastosować inny sposób udostępniania kontrolek serwerowych na stronie wzorcowej. Takie podejście okaże się bardziej efektywne.

Należy w tym celu udostępnić kontrolki serwerowe pod postacią publicznych właściwości. Pokazano to na listingu 5.12.

Listing 5.12. Udostępnianie kontrolek serwerowych strony wzorcowej poprzez publiczną właściwość

VB
```
<%@ Master Language="VB" %>

<script runat="server">
    Public Property MasterPageLabel1() As Label
        Get
            Return Label1
        End Get
        Set(ByVal Value As Label)
            Label1 = Value
```

```
       End Set
    End Property
</script>
```

C#

```
<%@ Master Language="C#" %>

<script runat="server">
    public Label MasterPageLabel
    {
        get
        {
            return Label1;
        }
        set
        {
            Label1 = value;
        }
    }
</script>
```

W tym przypadku publiczna właściwość o nazwie `MasterPageLabel1` umożliwia uzyskanie dostępu do kontrolki `Label`, której `ID` jest równe `Label1`. Można teraz utworzyć egzemplarz właściwości `MasterPageLabel1` na stronie z zawartością i nadpisać dowolny atrybut kontrolki serwerowej `Label`. Jeżeli trzeba na przykład zmienić rozmiar identyfikatora GUID, który jest tworzony przez stronę wzorcową i wyświetlany przez kontrolkę serwerową `Label1`, wtedy wystarczy zwyczajnie nadpisać atrybut `Font.Size` tej kontrolki `Label`. Pokazano to na listingu 5.13.

Listing 5.13. Nadpisywanie atrybutu kontrolki Label umieszczonej na stronie wzorcowej

VB

```
<%@ Page Language="VB" MasterPageFile="~/Wrox.master" %>
<%@ MasterType VirtualPath="~/Wrox.master" %>

<script runat="server">
   Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
      Master.MasterPageLabel1.Font.Size = 25
   End Sub
</script>
```

C#

```
<%@ Page Language="C#" MasterPageFile="~/Wrox.master" %>
<%@ MasterType VirtualPath="~/Wrox.master" %>

<script runat="server">
    protected void Page_Load(object sender, EventArgs e)
    {
        Master.MasterPageLabel1.Font.Size = 25;
    }
</script>
```

Takie podejście może być najbardziej efektywne spośród tych, które pozwalają pobrać referencję do dowolnej kontrolki serwerowej udostępnianej przez stronę wzorcową.

Określanie domyślnej zawartości na stronie wzorcowej

Jak już mogliśmy się przekonać, strona wzorcowa pozwala określić obszary zawartości, które mogą być wykorzystywane przez strony z zawartością. Strony wzorcowe mogą zawierać tylko jeden obszar zawartości, ale równie dobrze mogą obejmować wiele takich obszarów. Dość interesującym rozwiązaniem związanym z obszarem zawartości jest to, że podczas tworzenia stron wzorcowych można zdefiniować domyślną treść dla tych obszarów. Ta domyślna zawartość może być zostawiona na swoim miejscu i nie musi być nadpisywana przez stronę z zawartością. Na listingu 5.14 pokazano stronę wzorcową z domyślną treścią umieszczaną w obszarze zawartości.

Listing 5.14. Definiowanie domyślnej zawartości strony wzorcowej

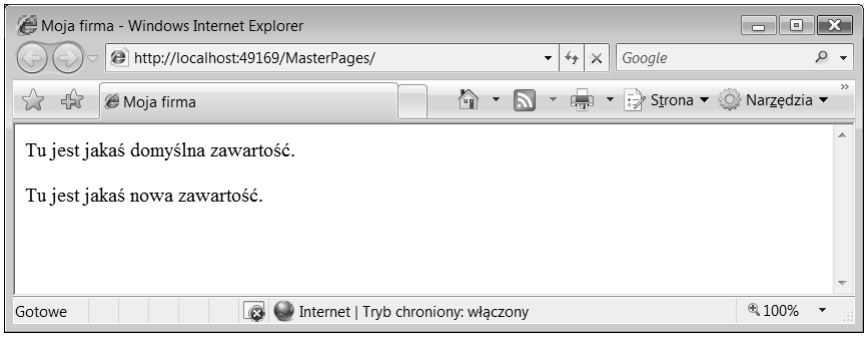
```
<%@ Master Language="VB" %>

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Moja firma</title>
    <asp:ContentPlaceHolder id="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <form id="form1" runat="server">
        <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
        Tu jest jakaś domyślna zawartość.
        </asp:ContentPlaceHolder><p>
        <asp:ContentPlaceHolder ID="ContentPlaceHolder2" runat="server">
        Tu jest jeszcze więcej domyślnej zawartości.
        </asp:ContentPlaceHolder></p>
    </form>
</body>
</html>
```

Aby umieścić domyślną treść w obszarze zawartości na stronie wzorcowej, wystarczy na samej stronie wzorcowej umieścić kontrolkę serwerową `ContentPlaceHolder`. Każda strona z zawartością dziedzicząca po tej stronie wzorcowej dziedziczy także domyślną treść. Na listingu 5.15 pokazano stronę z zawartością, która przesłania tylko jeden obszar zawartości strony wzorcowej.

Listing 5.15. Przesłanianie domyślnej zawartości na stronie z zawartością

```
<%@ Page Language="VB" MasterPageFile="~/MasterPage.master" %>

<asp:Content ID="Content3" ContentPlaceHolderId="ContentPlaceHolder2"
 runat="server">
    Tu jest jakaś nowa zawartość.
</asp:Content>
```

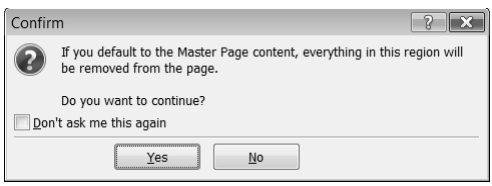
Pokazany kod pozwala utworzyć stronę z jednym obszarem zawartości, który dziedziczony jest po samej stronie wzorcowej, oraz z drugim obszarem, pochodzącym ze strony z zawartością (zobacz rysunek 5.10).



Rysunek 5.10

Kolejnym interesującym udogodnieniem jest to, że podczas pracy z obszarami zawartości w widoku projektanta Visual Studio 2010 pojawiają się inteligentne znaczniki, które ułatwiają pracę z domyślną treścią.

Rozpoczynając pracę ze stronami z zawartością, można zauważyć, że w kontrolkach serwerowych `Content` na początku pojawia się domyślna zawartość. Można zmienić tę zawartość, klikając inteligentny znacznik i wybierając z menu podręcznego opcję *Create Custom Content*. Opcja pozwala przesłonić zawartość strony wzorcowej i wstawić swoją własną zawartość. Po umieszczeniu własnych elementów wewnątrz obszaru zawartości inteligentny znacznik pokaże inną opcję — *Default to Master's Content*. Opcja pozwala przywrócić domyślną zawartość strony wzorcowej i usunąć wszystko, co zostało do tej pory umieszczone w obszarze zawartości. Skutkuje to powrotem do zawartości domyślnej. Po wybraniu tej opcji pojawi się ostrzeżenie informujące o tym, że wszystkie elementy wstawione w kontrolce serwerowej zostaną usunięte. Pokazano to na rysunku 5.11.

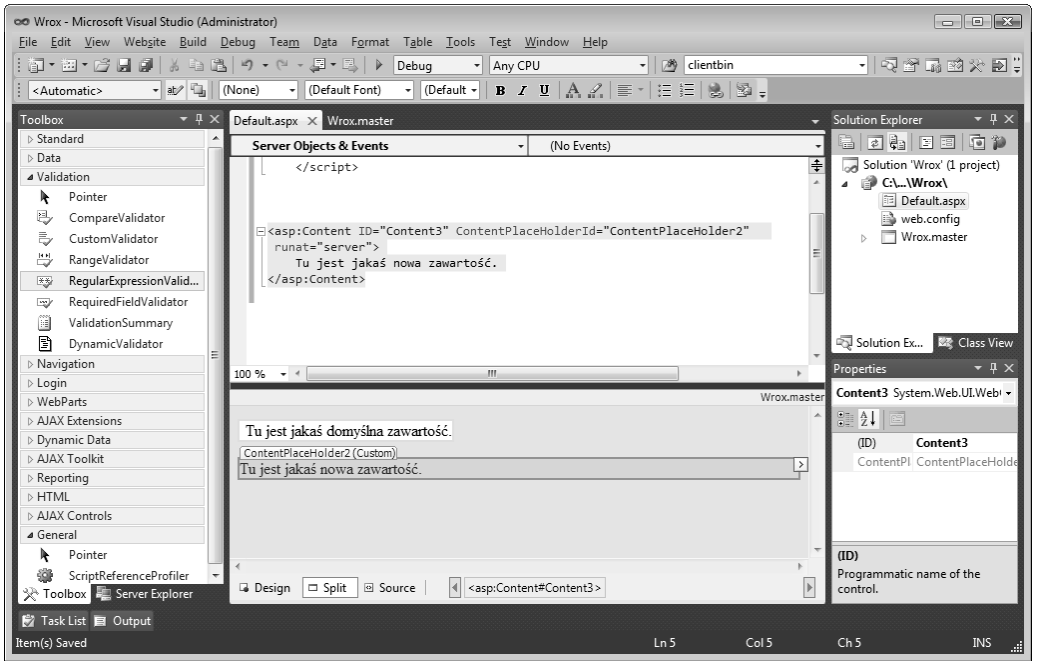


Rysunek 5.11

Po zmianie domyślnej zawartości kontrolki `Content` strona będzie wyglądała podobnie do tej z rysunku 5.12.

Programowe przypisywanie strony wzorcowej

Z poziomu każdej strony z zawartością można w łatwy sposób programowo przypisać stronę wzorcową. Wykonuje się to za pomocą właściwości `Page.MasterPageFile`. Taka opcja może być użyta bez względu na to, czy inna strona wzorcowa została już przypisana w dyrektywie `@Page`.



Rysunek 5.12

by wykonać to zadanie, można skorzystać z właściwości `Page.MasterPageFile` w zdarzeniu `PreInit`. Zdarzenie `PreInit` jest pierwszym zdarzeniem w całym cyklu przetwarzania strony, w którym można uzyskać dostęp do właściwości strony. Z tego powodu jest to najlepsze miejsce, w którym do stron z zawartością można przypisać stronę wzorcową. `PreInit` to warte uwagi zdarzenie. Nabiera ono szczególnego znaczenia podczas pracy ze stronami wzorcowymi, ponieważ jest to jedyny moment, gdy zmiana może mieć wpływ zarówno na stronę wzorcową, jak i stronę z zawartością, zanim zostaną one połączone w jedną stronę właściwą. Na listingu 5.16 pokazano sposób programowego przypisania strony wzorcowej z poziomu strony z zawartością.

Listing 5.16. Korzystanie z procedury obsługi zdarzenia Page_Prelnit w celu programowego przypisania strony wzorcowej

VB

```
<%@ Page Language="VB" %>

<script runat="server">
    Protected Sub Page_PreInit(ByVal sender As Object, ByVal e As System.EventArgs)
        Page.MasterPageFile = "~/MyMasterPage.master"
    End Sub
</script>
```

C#

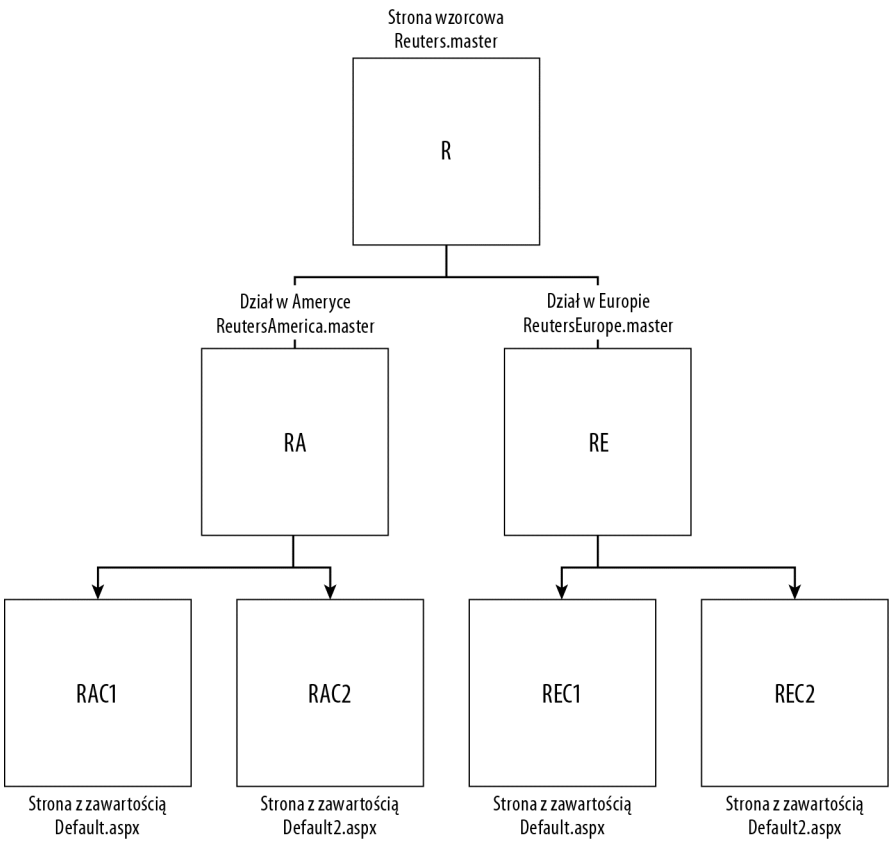
```
<%@ Page Language="C#" %>

<script runat="server">
    protected void Page_PreInit(object sender, EventArgs e)
    {
        Page.MasterPageFile = "~/MyMasterPage.master";
    }
</script>
```

W tym przypadku strona jest generowana dynamicznie. Strona wzorcowa przypisywana jest do strony z zawartością na początku procesu konstrukcji strony. Należy zwrócić szczególną uwagę na to, że strona z zawartością musi się spodziewać kontrolek `Content`, w przeciwnym razie pojawi się błąd.

Osadzanie stron wzorcowych

Mamy nadzieję, że widoczna jest już potęga stron wzorcowych w obszarze tworzenia aplikacji sieciowych korzystających z jednego schematu. Do tej pory tworzyliśmy pojedynczą stronę wzorcową, z której korzystały strony z zawartością. Jednak większość portali firm i organizacji nie składa się z tylko dwóch warstw. W większych organizacjach istnieją różne oddziały i grupy, które mogą korzystać z różnych odmian stron wzorcowych. Jedna strona wzorcowa może wtedy zostać umieszczona w innej stronie wzorcowej. W ASP.NET jest to możliwe.

Przypuśćmy, że Reuters tworzy stronę wzorcową przeznaczoną do wykorzystania w intranecie całej firmy. Cała agencja Reuters może utworzyć jedną stronę wzorcową dla wszystkich w firmie, ale każdy dział agencji Reuters może udostępnić własne wzorce dla poszczególnych grup intranetu, które pozostają pod jej kontrolą. Dział Reutersa w Europie i dział Reutersa w Ameryce mogą mieć swoje własne strony wzorcowe. Pokazano to na rysunku 5.13.



Rysunek 5.13

Aby utworzyć strony działów Reutersa dla Europy i Reutersa dla Ameryki, programiści tworzą zwyczajne strony wzorcowe, które dziedziczą po globalnej stronie wzorcowej. Pokazano to na listingu 5.17.

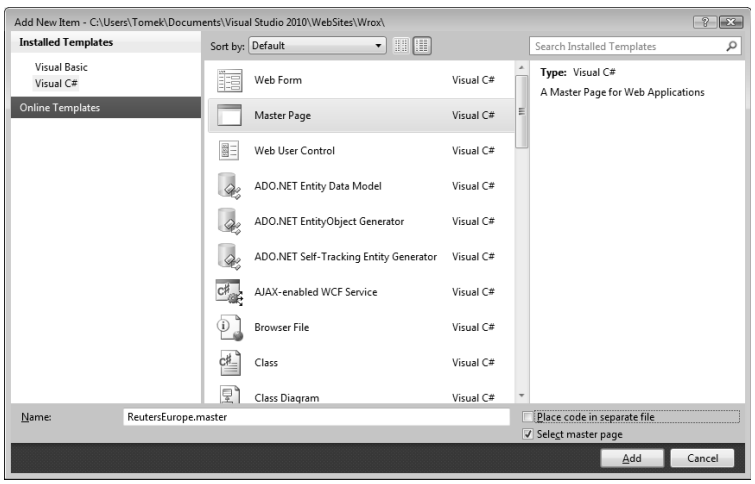
Listing 5.17. Główna strona wzorcowa

```
<%@ Master Language="VB" %>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Reuters</title>
    <asp:ContentPlaceHolder id="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <form id="form1" runat="server">
        <p><asp:Label ID="Label1" runat="server" BackColor="LightGray"
           BorderColor="Black" BorderWidth="1px" BorderStyle="Solid"
           Font-Size="XX-Large"> Główna strona wzorcowa Reuters </asp:Label></p>
        <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
        </asp:ContentPlaceHolder>
    </form>
</body>
</html>
```

Plik ReutersMain.master

To prosta strona wzorcowa, ale doskonale nadaje się do pokazania funkcji osadzania. Główna strona wzorcowa jest używana globalnie w całej firmie. Obejmuje ona kontrolkę serwerową `ContentPlaceHolder`. Wartością `ID` kontrolki jest `ContentPlaceHolder1`.

Podczas tworzenia podstrony wzorcowej lub osadzonej strony wzorcowej wykonuje się te same zadania i w ten sam sposób, który był stosowany przy tworzeniu zwykłej strony wzorcowej. Z okna dialogowego *Add New Item* wybierzmy opcję *Master Page* i upewnijmy się, że zaznaczona jest opcja *Select master page*. Pokazano to na rysunku 5.14. W ten sposób po raz kolejny przejdziemy do okna dialogowego, które pozwala wybrać stronę wzorcową.



Rysunek 5.14

Na listingu 5.18 pokazano, w jaki sposób można korzystać ze strony wzorcowej w pliku podstrony wzorcowej.

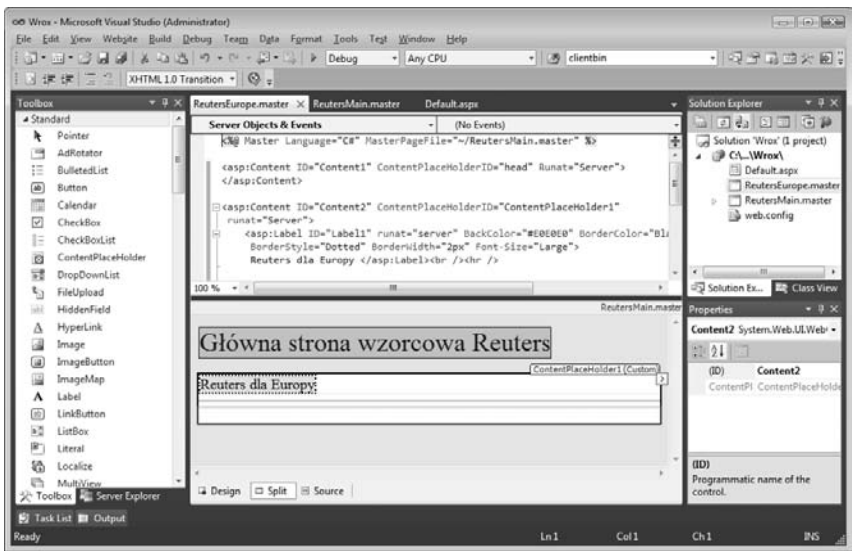
Listing 5.18. Podstrona wzorcowa

```
<%@ Master Language="VB" MasterPageFile="~/ReutersMain.master" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"
 runat="Server">
    <asp:Label ID="Label1" runat="server" BackColor="#E0E0E0" BorderColor="Black"
     BorderStyle="Dotted" BorderWidth="2px" Font-Size="Large">
     Reuters dla Europy </asp:Label><br /><hr />

        <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
        </asp:ContentPlaceHolder>
</asp:Content>
```

Plik ReutersEurope.master

Patrząc na kod zaprezentowanej strony, można zauważyć, że nie różni się ona bardzo od typowej strony *.aspx*, która korzysta ze strony wzorcowej. Atrybut `MasterPageFile` jest ten sam. Umieszczony jest on jednak w dyrektywie `@Master`, podczas gdy zwykła strona używa dyrektywy `@Page`. Kontrolka `Content2` także używa atrybutu `ContentPlaceHolderId` kontrolki `Content`. Atrybut łączy obszar zawartości z obszarem zawartości `ContentPlaceHolder1`, zdefiniowanym w głównej stronie wzorcowej.

Jedną z funkcji ASP.NET jest możliwość podglądu osadzonych stron wzorcowych bezpośrednio w widoku Design środowiska Visual Studio 2010. Wersje starsze od środowiska Visual Studio 2008 wyświetlały błąd, gdy próbowano pokazać osadzoną stronę wzorcową. Na rysunku 5.15 przedstawiono osadzoną stronę wzorcową w widoku Design środowiska Visual Studio 2010.



Rysunek 5.15

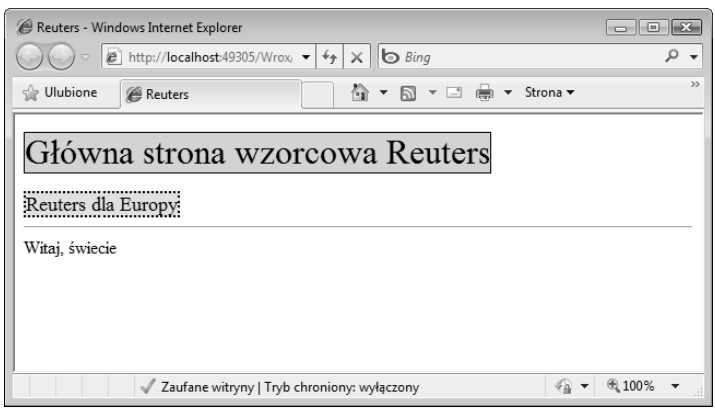
Wewnątrz podstrony wzorcowej pokazanej na listingu 5.18 można teraz używać tylu kontrolek serwerowych `ContentPlaceHolder`, ile tylko jest potrzebnych. Kontrolki te są dostępne dla stron z zawartością korzystających z tej strony wzorcowej. Na listingu 5.19 pokazano stronę z zawartością, która korzysta z podstrony wzorcowej `ReutersEurope.master`.

Listing 5.19. Strona z zawartością

Default.aspx

```
<%@ Page Language="VB" MasterPageFile="~/ReutersEurope.master" %>

<asp:Content ID="Content1" ContentPlaceHolderId="ContentPlaceHolder1"
 runat="server">
    Witaj, świecie
</asp:Content>
```

Jak można zauważyć, w pokazanej stronie wzorcowej wartością atrybutu `MasterPageFile` w dyrektywie `Page` jest utworzona podstrona wzorcowa. Dziedziczenie po stronie wzorcowej `ReutersEurope` powoduje, że wynikowa strona wzorcowa łączy w sobie obie strony (*ReutersMain.master* oraz *ReutersEurope.master*) w pojedynczą stronę wzorcową. Kontrolka `Content` strony z zawartością wskazuje na obszar zawartości również zdefiniowany przez podstronę wzorcową. Jak można zauważyć, w kodzie wykorzystano do tego atrybut `ContentPlaceHolderId`. W rezultacie otrzymujemy niezbyt ładną stronę końcową. Pokazano ją na rysunku 5.16.



Rysunek 5.16

Obserwując rezultat końcowy, można się przekonać, że podstrona wzorcowa działa całkiem nieźle.

Strony wzorcowe dostosowane do przeglądarek

W wielu przypadkach programiści tworzą aplikacje, które będą wyświetlane w wielu różnych przeglądarkach. Niektórzy odbiorcy aplikacji mogą korzystać z przeglądarki Microsoft Internet Explorer, niektórzy mogą używać Firefoksa lub programu Google Chrome. Inni użytkownicy mogą przeglądać strony aplikacji na urządzeniu Pocket PC lub w telefonie komórkowym Nokia.

Z tego powodu ASP.NET pozwala umieścić na stronie z zawartością wiele stron wzorcowych. W zależności od przeglądarki użytej przez użytkownika końcowego silnik ASP.NET wstawia właściwy plik strony wzorcowej. Powstają zatem strony wzorcowe specyficzne dla danej przeglądarki i w ten sposób zapewniane są użytkownikom końcowym najlepsze możliwe doświadczenia pod-

czas oglądania stron. Można dzięki temu wykorzystać możliwości oferowane przez konkretne przeglądarki. Sposób wykorzystania wielu stron wzorcowych pokazany jest na listingu 5.20.

Listing 5.20. Strona z zawartością, która może pracować z więcej niż jedną stroną wzorcową

```
<%@ Page Language="VB" MasterPageFile="~/Wrox.master"
       Mozilla:MasterPageFile="~/WroxMozilla.master"
       Opera:MasterPageFile="~/WroxOpera.master" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderId="ContentPlaceHolder1"
 runat="server">
    Witaj, świecie
</asp:Content>
```

Jak można zauważyć na pokazanym listingu strony z zawartością, może ona działać z trzema rodzajami stron wzorcowych. Pierwszy z nich korzysta z atrybutu `MasterPageFile`. To ustawienie domyślne, używane przez te przeglądarki, które nie spełniają kryteriów nałożonych przez kolejne opcje. Oznacza to, że jeżeli żądanie nie przychodzi od przeglądarki Opera lub Mozilla, wtedy używana jest domyślna strona wzorcowa *Wrox.master*. Jeżeli jednak żądanie przychodzi od przeglądarki Opera, wtedy używany jest plik *WroxOpera.master*. Pokazano to na rysunku 5.17.

Rysunek 5.17

Listę dostępnych przeglądarek można znaleźć w katalogu *C:\Windows\Microsoft.NET\Framework\v4.0.xxxxx\CONFIG\Browsers* na serwerze, na którym aplikacje będą umieszczane. Niektóre z opcji wypisano poniżej:

- avantgo
- cassio
- Default
- docomo
- ericsson
- EZWap
- gateway
- generic
- goAmerica
- ie
- Jataayu
- jphone
- legend
- MME
- mozilla
- netscape
- nokia
- openwave
- opera
- palm
- panasonic
- pie
- webtv
- winwap
- xiino

Można oczywiście dodać swoje własne pliki *.browser*, jeżeli zajdzie taka potrzeba.

Porządek wywoływania zdarzeń

Podczas pracy ze stronami wzorcowymi i stronami z zawartością korzysta się z tych samych zdarzeń (na przykład ze zdarzenia `Load`). Należy wiedzieć, które ze zdarzeń występują przed innymi. Tworzy się bowiem jedną stronę z dwóch klas i wymagany jest przy tym odpowiedni porządek. Gdy użytkownik wysyła z przeglądarki żądanie strony z zawartością, wtedy zdarzenia wywoływane są w następującej kolejności:

1. **Inicjalizacja kontrolek potomnych strony wzorcowej** — wszystkie kontrolki znajdujące się na stronie wzorcowej są inicjalizowane.
2. **Inicjalizacja kontrolek potomnych strony z zawartością** — wszystkie kontrolki znajdujące się na stronie z zawartością są inicjalizowane.
3. **Inicjalizacja strony wzorcowej** — inicjalizowana jest strona wzorcowa.
4. **Inicjalizacja strony z zawartością** — inicjalizowana jest strona z zawartością.
5. **Wczytywanie strony z zawartością** — wczytywana jest strona z zawartością (wywoływane jest zdarzenie `Page_Load`, a następnie zdarzenie `Page_LoadComplete`).
6. **Wczytywanie strony wzorcowej** — wczytywana jest strona wzorcowa (wywoływane jest zdarzenie `Page_Load`, a następnie zdarzenie `Page_LoadComplete`).
7. **Wczytywanie kontrolek potomnych strony wzorcowej** — na stronę wczytywane są kontrolki serwerowe strony wzorcowej.
8. **Wczytywanie kontrolek potomnych strony z zawartością** — na stronę wczytywane są kontrolki serwerowe strony z zawartością.

Podczas tworzenia aplikacji należy zwrócić uwagę na porządek występowania zdarzeń. Jeżeli trzeba użyć wartości kontrolek serwerowych umieszczonych na stronie wzorcowej w określonej stronie z zawartością, to nie można tego zrobić bezpośrednio z poziomu procedury obsługi zdarzenia `Page_Load` strony z zawartością. Dzieje się tak, ponieważ zdarzenie wywoływane jest przed zakończeniem działania procedury obsługi zdarzenia `Page_Load` strony wzorcowej. Opisany problem zmusił twórców ASP.NET do wprowadzenia nowego zdarzenia `Page_LoadComplete`. Zdarzenie `Page_LoadComplete` występuje zaraz po zdarzeniu `Page_Load`. Można skorzystać z tego porządku w celu pobrania wartości ze strony wzorcowej, pomimo że zawartość strony po wywołaniu zdarzenia nie jest jeszcze uzupełniona.

Buforowanie stron wzorcowych

Podczas pracy z typowymi stronami *.aspx* można ustawić sposób buforowania danych wyjściowych za pomocą następującej konstrukcji (lub jej odmiany):

```
<%@ OutputCache Duration="10" Varybyparam="None" %>
```

Dzięki temu strona będzie przechowywana w pamięci podręcznej serwera przez 10 sekund. Wielu programistów korzysta z techniki buforowania danych wyjściowych w celu zwiększenia wydajności swoich stron ASP.NET. Takie rozwiązanie ma także sens w przypadku stron z danymi, które nie tracą swojej ważności zbyt szybko.

W jaki sposób można zastosować buforowanie danych wyjściowych stron ASP.NET w przypadku stron wzorcowych? Po pierwsze, nie można zastosować buforowania wyłącznie w odniesieniu do stron wzorcowych. Nie można umieścić dyrektywy `OutputCache` na samej

stronie wzorcowej. Jeżeli spróbujemy coś takiego zrobić, wtedy podczas drugiego pobrania strony pojawi się błąd. Aplikacja nie będzie mogła odnaleźć strony w pamięci podręcznej.

Aby możliwa była współpraca mechanizmu buforowania ze stronami wzorcowymi, należy umieścić dyrektywę `OutputCache` na stronie z zawartością. W ten sposób buforowana będzie treść zarówno strony z zawartością, jak i strony wzorcowej (należy pamiętać, że w tym miejscu jest to już jedna strona). Dyrektywa `OutputCache` umieszczona na stronie wzorcowej nie spowoduje, że pojawi się błąd. Strona nie będzie jednak buforowana. Dyrektywa działa tylko ze stronami z zawartością.

Inną nową i ciekawą funkcją środowiska ASP.NET 4 w obszarze buforowania jest możliwość sprawdzania stanu na poziomie kontrolek. Choć natychmiast przychodzi na myśl możliwość kontrolowania w ten sposób stanu takich kontrolek jak `GridView` i innych obejmujących rozbudowany stan, technikę tę można zastosować także do kontrolek `ContentPlaceHolder`.

Można na przykład napisać kod podobny do poniższego:

```
<asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server"
 ViewStateMode="Disabled">

</asp:ContentPlaceHolder>
```

Przy takich ustawieniach kontrolka `ContentPlaceHolder1` nie będzie korzystać ze stanu, nawet jeśli będą go używać pozostałe elementy strony. Opcja `ViewStateMode` przyjmuje wartości `Disabled`, `Enabled` i `Inherit`. Ustawienie `Disabled` powoduje wyłączenie obsługi stanu kontrolki, opcja `Enabled` włącza tę funkcję, a opcja `Inherit` spowoduje zastosowanie wartości określonej w dyrektywie `@Page`. Wyłączenie obsługi stanu prowadzi do poprawy wydajności stron.

ASP.NET AJAX i strony wzorcowe

Wiele większych aplikacji ASP.NET w dzisiejszych czasach korzysta ze stron wzorcowych oraz możliwości, jakie daje ta technologia. Pozwala ona tworzyć portale na podstawie określonego wzorca. ASP.NET 4 obejmuje instalowaną domyślnie technologię ASP.NET AJAX. Okaże się wkrótce, że strony wzorcowe współpracują z technologią AJAX całkiem nieźle.

UWAGA

Technologia ASP.NET AJAX omówiona jest w rozdziale 18. tej książki.

Każda strona, która będzie korzystała z technologii AJAX, musi obejmować kontrolkę `ScriptManager`. Jeżeli tworzona strona z zawartością korzystająca z AJAX-a dziedziczy po stronie wzorcowej, wtedy kontrolka `ScriptManager` musi być umieszczona na stronie wzorcowej.

UWAGA

Należy pamiętać, że na stronie może znajdować się tylko jedna kontrolka `ScriptManager`.

Skonfigurowanie strony wzorcowej pod kątem obsługi technologii AJAX nie jest trudne. Wystarczy dodać do strony wzorcowej kontrolkę `ScriptManager`. Przykład takiego rozwiązania przedstawia listing 5.21.

Listing 5.21. Strona wzorcowa z obsługą technologii AJAX

```
<%@ Master Language="VB" %>

<script runat="server">

</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <asp:ContentPlaceHolder id="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:ScriptManager ID="ScriptManager1" runat="server" />
        <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">

        </asp:ContentPlaceHolder>
    </div>
    </form>
</body>
</html>
```

Jak można zauważyć na listingu 5.21, jedyną różnicą pomiędzy stroną wzorcową w technologii AJAX a standardową stroną wzorcową jest obecność w kodzie kontrolki serwerowej `ScriptManager`. Technikę tę należy stosować wtedy, gdy strona wzorcowa korzysta z pewnych udogodnień technologii AJAX, a nawet wtedy, gdy strony z zawartością w ogóle nie korzystają z AJAX-a.

Kontrolka `ScriptManager` na stronie wzorcowej jest przydatna także wtedy, gdy istnieją pewne wspólne fragmenty kodu w języku JavaScript, wykorzystywane na wszystkich stronach aplikacji. Na listingu 5.22 pokazano, w jaki sposób można łatwo umieścić kod w języku JavaScript na wszystkich stronach korzystających ze strony wzorcowej.

Listing 5.22. Włączanie skryptów poprzez stronę wzorcową

```
<%@ Master Language="VB" %>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <asp:ContentPlaceHolder id="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:ScriptManager ID="ScriptManager1" runat="server">
            <Scripts>
```

```
            <asp:ScriptReference Path="myScript.js" />
        </Scripts>
    </asp:ScriptManager>
    <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">

    </asp:ContentPlaceHolder>
</div>
</form>
</body>
</html>
```

Dzięki zastosowaniu rozwiązania pokazanego w przykładzie plik *myScript.js* będzie umieszczany na każdej stronie z zawartością, która korzysta ze strony wzorcowej z obsługą AJAX-a. Jeżeli strona z zawartością także musi korzystać z technologii AJAX, wtedy nie można już tak zwyczajnie dodać do niej kontrolki `ScriptManager`. Strona z zawartością musi w takim przypadku korzystać z kontrolki `ScriptManager`, która jest już obecna na stronie wzorcowej.

W związku z powyższym, jeżeli strona z zawartością musi dołączyć do kontrolki `ScriptManager` dodatkowe składniki, wtedy może skorzystać z kontrolki serwerowej `ScriptManagerProxy` i za jej pomocą dostać się do kontrolki ze strony wzorcowej. Korzystanie z kontrolki `ScriptManagerProxy` pozwala dodać do kontrolki `ScriptManager` elementy, które są specyficzne dla danego egzemplarza strony z zawartością.

Na listingu 5.23 pokazano, w jaki sposób strona z zawartością może dodać kolejne skrypty do strony poprzez kontrolkę `ScriptManagerProxy`.

Listing 5.23. Wstawianie dodatkowych elementów za pomocą kontrolki ScriptManagerProxy

```
<%@ Page Language="VB" MasterPageFile="~/AjaxMaster.master" %>

<asp:Content ID="Content1" ContentPlaceHolderId="head" runat="server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderId="ContentPlaceHolder1"
 runat="server">

 <asp:ScriptManagerProxy ID="ScriptManagerProxy1" runat="server">
    <Scripts>
        <asp:ScriptReference Path="myOtherScript.js" />
    </Scripts>
 </asp:ScriptManagerProxy>
</asp:Content>
```

W tym przypadku strona z zawartością korzysta z kontrolki `ScriptManagerProxy` w celu dodania na stronę kolejnych skryptów. Kontrolka `ScriptManagerProxy` działa dokładnie tak samo jak główna kontrolka `ScriptManager`. Różnica polega na tym, że kontrolka `ScriptManagerProxy` przewidziana jest do pracy ze stronami z zawartością dziedziczącymi po stronach wzorcowych. Kontrolka `ScriptManagerProxy` będzie współpracować z kontrolką `ScriptManager` i wykona wszystkie wymagane operacje.

Podsumowanie

W czasie tworzenia aplikacji, w której prawie każda strona korzysta ze wspólnego nagłówka, stopki lub sekcji nawigacyjnej, dobrym rozwiązaniem jest zastosowanie stron wzorcowych. Strony wzorcowe są łatwe w implementacji i pozwalają wprowadzić zmiany do każdej strony aplikacji poprzez modyfikację pojedynczego pliku. Wystarczy sobie wyobrazić, o ile łatwiejsze jest takie zadanie przy aplikacji, która zawiera tysiące stron.

W niniejszym rozdziale opisano strony wzorcowe w ASP.NET i wyjaśniono sposób tworzenia stron wzorcowych w aplikacjach sieciowych. Oprócz podstawowych zagadnień poruszono także temat porządku wywoływania zdarzeń, buforowanie oraz dostosowanie stron wzorcowych do konkretnych przeglądarek. Na koniec mała rada. Jeżeli w aplikacji występuje pewien szablon, naturalnym odruchem powinno być wykorzystanie stron wzorcowych — technika ta daje olbrzymie możliwości.

Zaawansowane programowanie

ASP.NET 4 z wykorzystaniem C# i VB

Platforma ASP.NET to główny konkurent języka Java w zakresie tworzenia aplikacji internetowych oraz dynamicznych stron internetowych. Każda jej wersja dostarcza wiele interesujących ulepszeń, a wśród nich te najważniejsze — pozwalające na zdjęcie z programisty obowiązku pisania dużych ilości nudnego kodu, bez którego jeszcze niedawno aplikacja nie mogłaby istnieć.

Niniejsza książka została napisana przez grupę wyjątkowych autorów. Bill Evjen to najaktywniejszy promotor technologii .NET, Scott Hanselman to główny menedżer w jednym z działów firmy Microsoft, prowadzący szkolenia dotyczące ASP.NET na całym świecie, a Devin Rader to pracownik firmy Infragistics. Ta doborowa trójka stworzyła świetny podręcznik, w całości poświęcony ASP.NET. Znajdziesz w nim informacje na temat stosowania języków VisualBasic oraz C# do tworzenia dynamicznych stron. Dowiesz się, jak wykorzystać kontrolki serwerowe, budować aplikacje z wykorzystaniem wzorca MVC oraz tchnąć życie w strony za pomocą technologii AJAX. Te i wiele innych bezcennych informacji, porad i wskazówek odkryjesz dzięki tej wyjątkowej książce!

Sprawdź, jaka moc drzemie w platformie ASP.NET!

- **Produktywność programistów**
- **Infrastruktura i tworzenie aplikacji ASP.NET**
- **Środowisko ADO.NET**
- **Kompilacja aplikacji ASP.NET**
- **Monitorowanie stanu aplikacji**
- **Kontrolki serwerowe**
- **Wykorzystanie stron wzorcowych**
- **Kompozycje i skórki**
- **Wykonywanie zapytań z wykorzystaniem LINQ**
- **Wykorzystanie formatu XML**
- **Bezpieczeństwo aplikacji ASP.NET**
- **Instrumentacja**

Nr katalogowy: 5767

Księgarnia internetowa: http://helion.pl

Zamówienia telefoniczne: 0 801 339900 0 601 339900

Helion

Sprawdź najnowsze promocje: http://helion.pl/promocje
Książki najchętniej czytane: http://helion.pl/bestsellery
Zamów informacje o nowościach: http://helion.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

helion.pl księgarnia internetowa

Cena 199,00 zł

ISBN 978-83-246-2846-9

9 788324 628469

Informatyka w najlepszym wydaniu