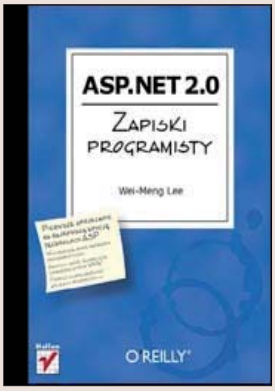
ASP.NET 2.0. Zapiski programisty



Autor: Wei-Meng Lee
Tłumaczenie: Robert Górczyński
ISBN: 83-246-0247-X
Tytuł oryginału: ASP.NET 2.0: A Developers Notebook
Format: B5, stron: 392

Pierwsze spojrzenie na najnowszą edycję technologii ASP

- Wykorzystaj nowe narzędzia programistyczne
- Zastosuj nowe technologie tworzenia witryn WWW
- Podnieś bezpieczeństwo aplikacji internetowych

Najnowsza wersja ASP.NET w połączeniu z najnowszą wersją środowiska programistycznego Visual Studio daje programistom aplikacji internetowych ogrom nowych możliwości. Dzięki nim aplikacje mogą być wydajniejsze i bezpieczniejsze, a proces ich tworzenia – znacznie krótszy. Za pomocą stron wzorcowych można błyskawicznie stworzyć spójnie wyglądającą witrynę WWW, profile zapewnią przechowywanie informacji o użytkownikach, a nowe kontrolki Grid View pozwolą na szybki dostęp do danych i wyświetlanie ich na stronie. A to tylko niektóre z nowych funkcji ASP.NET 2.0.

Aby dowiedzieć się o pozostałych, sięgnij po książkę „ASP.NET 2.0. Zapiski programisty". Jeśli wolisz poznawać nowe technologie, wykorzystując je w praktyce, a nie czytając długie teoretyczne wywody, to jest to właśnie książka dla Ciebie. Znajdziesz tu notatki programistów, którzy pracowali z ASP.NET 2.0 już wtedy, gdy w prasie pojawiały się pierwsze nieśmiałe zapowiedzi tej technologii – programistów, którzy wiedzą, że dla praktyków takie pozornie chaotyczne notatki okażą się nieocenioną pomocą przy poznawaniu nowych możliwości ASP.NET 2.0.

- Tworzenie projektu internetowego w Visual Studio 2005
- Definiowanie stron wzorcowych dla witryny
- Korzystanie z Web Parts Framework do budowania portali internetowych
- Zastosowanie kontrolki Grid View do pobierania i wyświetlania danych
- Tworzenie kont użytkowników za pomocą Create User Wizard
- Poprawianie wydajności działania witryny
- Personalizacja aplikacji za pomocą profilów
- Lokalizacja aplikacji

Poznaj niezwykłe możliwości najnowszej wersji ASP .NET

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

Spis treści

ROZDZIAŁ 2.

Strony wzorcowe i nawigacja witryny

W tym rozdziale:

- ✓ Tworzenie strony wzorcowej dla swojej witryny
- ✓ Użycie strony wzorcowej jako szablonu strony z zawartością
- ✓ Modyfikowanie strony wzorcowej w trakcie działania
- ✓ Tworzenie mapy witryny dla swojej witryny internetowej
- ✓ Wyświetlanie hierarchicznych danych przy użyciu kontrolki `TreeView`
- ✓ Programowe zapełnianie kontrolki `TreeView`
- ✓ Wyświetlanie rozwijanych menu za pomocą kontrolki `Menu`

Dzięki Visual Studio 2005 ASP.NET 2.0 obsługuje obecnie wizualne dziedziczenie strony — mechanizm podobny do dziedziczenia Windows Forms. Z pomocą ASP.NET 2.0 możesz teraz utworzyć pojedynczą stronę wzorcową zawierającą wspólne elementy dla stron na Twojej witrynie. Następnie możesz tworzyć strony internetowe, które będą dziedziczyć ze strony wzorcowej i w ten sposób zapewnisz spójny wygląd i działanie całej witryny.

Oprócz tego mechanizmu ASP.NET 2.0 oferuje nowe kontrolki ułatwiające nawigację na stronie. Te kontrolki, znane jako `SiteMapPath` i `Menu`, pozwolą Ci na dołączenie do Twojej witryny odnośników nawigacyjnych bez potrzeby pisania dużej ilości kodu.

Tworzenie strony wzorcowej dla swojej witryny

Większość witryn internetowych, które oglądasz w dzisiejszych czasach ma spójny wygląd i działanie oraz, prawdopodobnie, powtórzone na każdej stronie logo firmy i menu nawigacyjne. Dobrym przykładem jest tutaj witryna O'Reilly — na każdej stronie wyświetla się w górnej części znane logo O'Reilly, natomiast po lewej stronie menu nawigacyjne, co zostało pokazane na rysunku 2.1.

Rysunek 2.1. Większość witryn internetowych ma na stronach wspólne nagłówki i menu nawigacyjne

Jako programista sieciowy ASP.NET 1.x bez wątpienia nauczyłeś się, w jaki sposób używać kontrolek Web User do hermetyzacji wszystkich nagłówków oraz menu nawigacyjnych wykorzystywanych na swojej witrynie i jak osadzić je na każdej ze stron. Oczywiście wadą takiego rozwiązania jest to, że jeśli chcesz dokonać modyfikacji dowolnej z tych kontrolek, z reguły musisz ręcznie edytować każdą stronę, aby zmienić jej ułożenie.

WSKAZÓWKA

W celu zapewnienia stronom nagłówków zalecane jest używanie stron wzorcowych zamiast wykorzystywania kontrolek Web User. Dzięki takiemu rozwiązaniu, w razie jakichkolwiek zmian, trzeba jedynie zmodyfikować stronę wzorcową, a następne zostaną już zaktualizowane automatycznie.

W ASP.NET 2.0 nowa funkcja, znana jako *Master Pages* (strony wzorcowe), zajmuje się ograniczeniami wykorzystania kontrolek Web User do tworzenia nagłówków oraz informacji menu nawigacyjnych. W ASP.NET 2.0 możesz po prostu skonstruować stronę wzorcową, która będzie zawierała informacje z nagłówka Twojej strony. Następnie budujesz każdą kolejną stronę witryny, korzystając w pierwszej kolejności z dziedziczenia ze strony wzorcowej witryny.

Świetnie! Teraz strony wzorcowe znacznie ułatwiają utrzymanie spójnego wyglądu i działania całej witryny internetowej. Nie trzeba więc dłużej zmagać się z kontrolkami Web User!

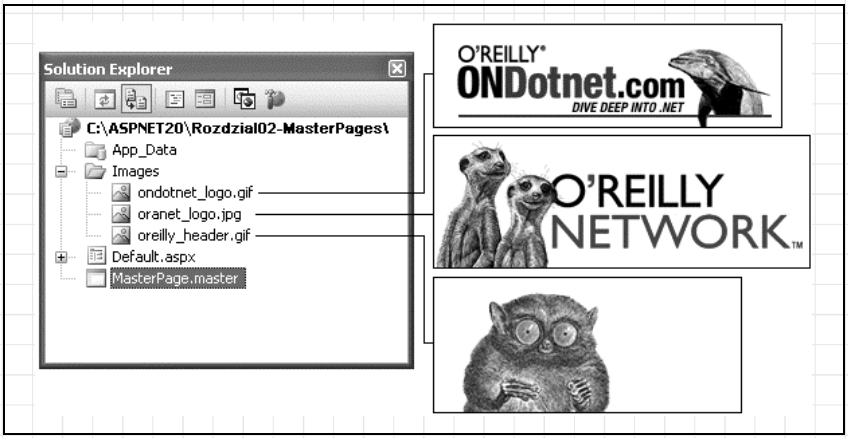
Jak to osiągnąć?

Aby zrozumieć, w jaki sposób działają strony wzorcowe, zbudujesz w kilku kolejnych ćwiczeniach aplikację sieciową używającą stron wzorcowych. W tym ćwiczeniu utworzymy stronę wzorcową i zapełnimy ją kilkoma kontrolkami.

1. W Visual Studio 2005 utwórz nową aplikację sieciową ASP.NET 2.0 i zapisz ją jako *C:\ASPNET20\Rozdzial02-MasterPages*.
2. Do projektu dodaj katalog o nazwie *Images* (kliknij prawym przyciskiem myszy nazwę projektu w oknie *Solution Explorer* i wybierz *Add Folder/Regular Folder*). Przekopiuj obrazki pokazane na rysunku 2.2 do nowego katalogu *C:\ASPNET20\Rozdzial02-MasterPages\Images*. Użyjesz tych obrazków do zbudowania swojej strony wzorcowej.

WSKAZÓWKA

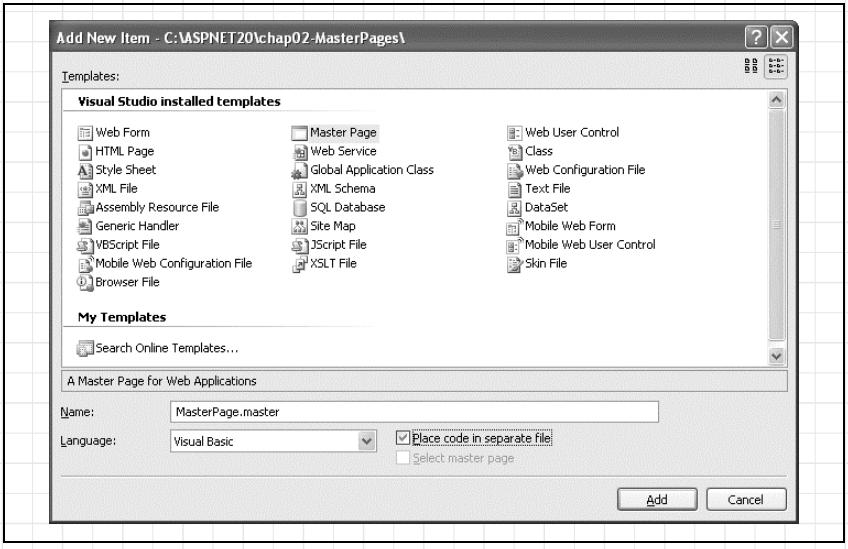
Wspomniane obrazki można pobrać pod adresem: *ftp://ftp.helion.pl/przyklady/aspnzp.zip*.



Rysunek 2.2. Obrazki w katalogu Images

3. Teraz utworzysz stronę wzorcową. Kliknij prawym przyciskiem myszy nazwę projektu w oknie *Solution Explorer*, a następnie z menu wybierz pozycję *Add New Item...*.

4. W oknie dialogowym *Add New Item* zaznacz szablon *Master Page* i użyj domyślnej nazwy — *MasterPage.master*, jak pokazano na rysunku 2.3. Kliknij przycisk *Add*, aby utworzyć stronę.



Rysunek 2.3. Dodanie strony wzorcowej do projektu

5. W oknie *Solution Explorer* kliknij dwukrotnie *MasterPage.master*, a zobaczysz pustą stronę internetową zawierającą pojedynczą kontrolkę `ContentPlaceHolder`.

Kontrolka ContentPlaceHolder reprezentuje obszar zablokowany dla stron z zawartością (stron, które dziedziczą ze strony wzorcowej) i służy do zapełnienia go kontrolkami.

6. Zapełnij treścią, obrazkami, odnośnikami stronę *MasterPage.master* przez przeciągnięcie z paska narzędziowego i upuszczenie na stronie dwóch kontrolek typu `Image`, jednej typu `Panel` oraz czterech typu `LinkButton`, jak to pokazano na rysunku 2.4. Ustaw w następujący sposób właściwości każdej z tych kontrolek:

   Kontrolki typu `Image`

   Nazwij pierwszą kontrolkę `imgLogo` i ustaw jej właściwość `ImageURL` jako *Images\oreilly_header.gif*, aby wyświetlała logo O'Reilly. Drugiej kontrolce nadaj nazwę `imgTitle` i ustaw jej właściwość `ImageURL` jako *Images\oranet_logo.jpg*. Będzie ona wyświetlała logo O'Reilly Network.

   Kontrolka `Panel`

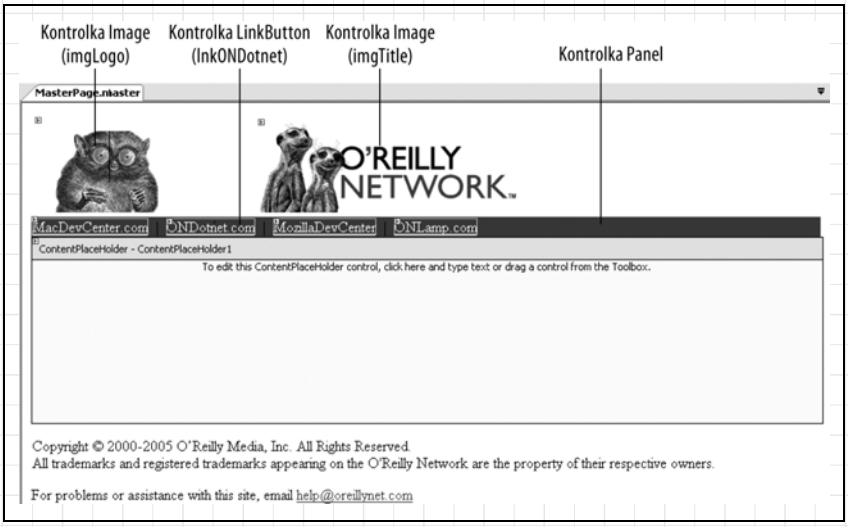
   Użyj myszy, aby rozciągnąć jej obszar na całą stronę, a następnie ustaw jej właściwość `Backcolor` jako `Maroon`.

   Kontrolki `LinkButton`

   Upuść cztery kontroli `LinkButton` na kontrolce `Panel`. Nazwij je tak, jak pokazano na rysunku 2.4 (wykorzystując ich właściwości `Text`). Ponieważ będziesz klikał jedynie jeden z tych odnośników, nazwij drugą kontrolkę `lnkONDotnet`. Ustaw jej właściwość `PostBackUrl` na *ONDotnet.aspx*.

7. Wpisz zawartość stopki na dole ekranu, tuż pod kontrolką `ContentPlaceHolder`. Aby to zrobić, umieść kursor na końcu kontrolki `ContentPlaceHolder` i naciśnij klawisz *Enter*. Możesz następnie rozpocząć wpisywanie informacji o prawach autorskich, jak to zostało pokazane na rysunku 2.4.

8. To już wszystko. Właśnie utworzyłeś swoją stronę wzorcową. W następnym ćwiczeniu zobaczysz, w jaki sposób z niej korzystać.



Rysunek 2.4. Zapełnianie strony wzorcowej

A co...

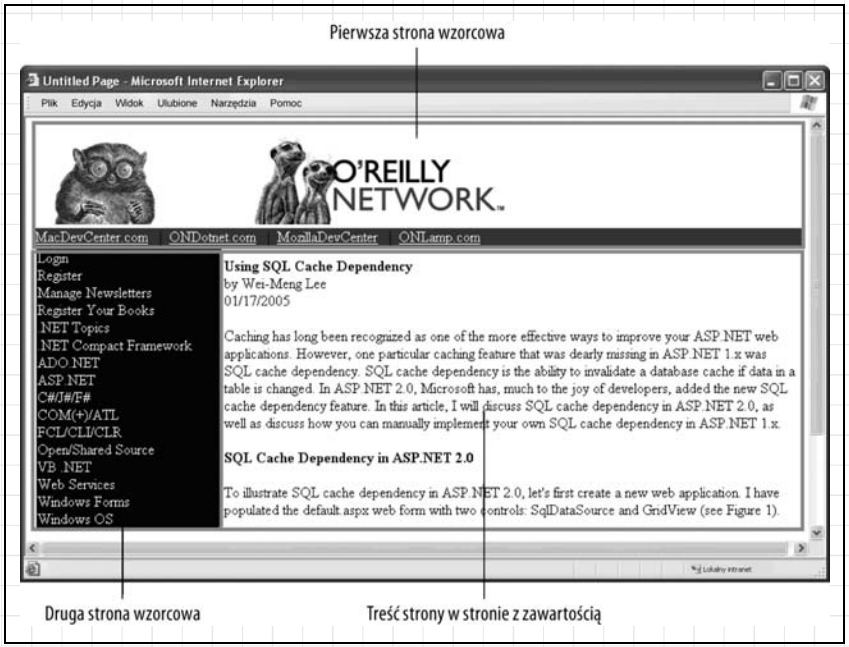
...z zagnieżdżonymi stronami wzorcowymi?

Strona wzorcowa zagnieżdżona wewnątrz innej strony wzorcowej bywa użyteczna w sytuacji, kiedy chcesz zmienić wygląd i działanie określonych obszarów swojej witryny. Na przykład, załóżmy, że O'Reilly Network chce udostępnić unikatowe paski nawigacyjne dla użytkowników stron na poszczególnych witrynach o tematyce sieciowej, takich jak ONDotnet.com lub MozillaDevCenter. Rysunek 2.5 pokazuje, w jaki sposób mogłoby to zostać zrealizowane przy użyciu strony zawierającej dwie strony wzorcowe, z których druga jest zagnieżdżona w pierwszej.

Aby dowiedzieć się, w jaki sposób zagnieździć jedną stronę wewnątrz innej, wykonaj następujące kroki:

1. Dodaj nową stronę wzorcową do projektu (*C:\ASPNET20\Rozdzial02-MasterPages*) i nazwij ją `MasterPage2.master`.

2. Druga strona wzorcowa będzie składała się z tabeli zawierającej jeden wiersz i dwie kolumny. Pierwsza kolumna zawiera kontrolkę `Content` (`Content1`), która wypełni kontrolkę `ContentPlaceHolder` na pierwszej stronie wzorcowej (`ContentPlaceHolder1`) tekstem (dla uproszczenia, w tym przykładzie użyjesz tekstu zamiast odnośników) takim,



Rysunek 2.5. Zagnieżdżone strony wzorcowe

jak *Logowanie*, *Rejestracja* itd. Druga kolumna zawiera kontrolkę `ContentPlaceHolder` (`ContentPlaceHolder2`) dla stron z zawartością, umożliwiającą dołączenie do niej innych kontrolek. Przełącz się do widoku *Source View* i dodaj do strony *MasterPage2.master* kod przedstawiony na listingu 2.1.

O stronie z zawartością dowiesz się więcej w następnym ćwiczeniu.

WSKAZÓWKA

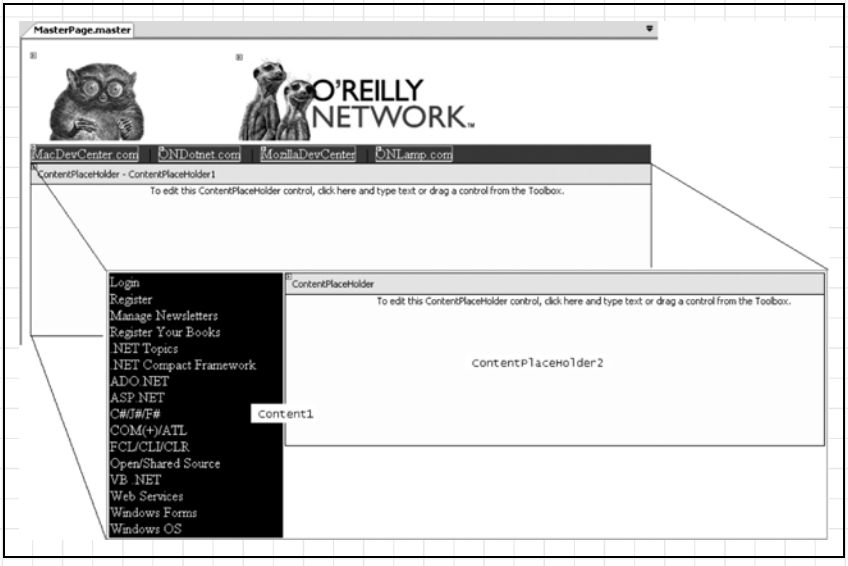
Visual Studio 2005 nie obsługuje wizualnej edycji lub tworzenia zagnieżdżonych stron wzorcowych. Tego typu zadania będziesz musiał wykonać w widoku *Source View*.

Listing 2.1. Tworzenie zagnieżdżonej strony wzorcowej

```
<%@ Master Language="VB" MasterPageFile="~/MasterPage.master"
   CodeFile="MasterPage2.master.vb" AutoEventWireup="false"
   Inherits="MasterPage2" %>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1"
      Runat="Server">
<table>
   <tr>
      <td width="200" bgcolor="black">
```

```
        <font color="white">
                Login<br />Register<br />Manage Newsletters<br />
            Register Your Books<br />.NET Topics<br />
            .NET Compact Framework<br />ADO.NET<br />
            ASP.NET<br />C#/J#/F#<br />COM(+)/ATL<br />
            FCL/CLI/CLR<br />Open/Shared Source<br />
            VB .NET<br />Web Services<br />Windows Forms<br />
            Windows OS<br />
        </font>
        </td>
        <td>
           <asp:contentplaceholder id="ContentPlaceHolder2" runat="server">

           </asp:contentplaceholder>
        </td>
     </tr>
  </table>
  </asp:Content>
```

3. Na rysunku 2.6 pokazano związki między dwiema stronami wzorcowymi.



Rysunek 2.6. Związki między dwiema stronami wzorcowymi

W kolejnym ćwiczeniu dowiesz się, w jaki sposób korzystać z utworzonych przed chwilą stron wzorcowych, w celu utrzymania spójnego wyglądu i działania Twoich stron z zawartością.

Użycie strony wzorcowej jako szablonu strony z zawartością

Używając stron wzorcowych i dziedziczenia wizualnego, możesz sprawić, że wszystkie Twoje strony z zawartością w ramach witryny będą mieć spójny wygląd i działanie.

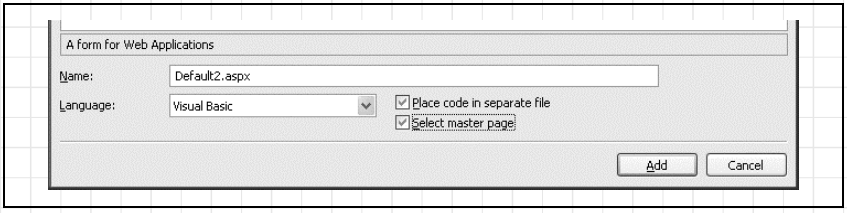
Jeśli utworzyłeś stronę wzorcową, możesz dla swojej witryny konstruować strony z zawartością, które jako szablonu będą używać strony wzorcowej.

Jak to osiągnąć?

W tym ćwiczeniu do swojego projektu dodasz strony z zawartością oraz nadasz im spójny wygląd i działanie dzięki użyciu stron wzorcowych utworzonych w poprzednim podrozdziale „Tworzenie strony wzorcowej dla swojej witryny". Następnie do strony z zawartością dołączysz kontrolki i zobaczysz, w jaki sposób ASP.NET łączy zawartość stron wzorcowych i stron z zawartością w trakcie działania.

Strona z zawartością jest formularzem Web Form, który używa strony wzorcowej.

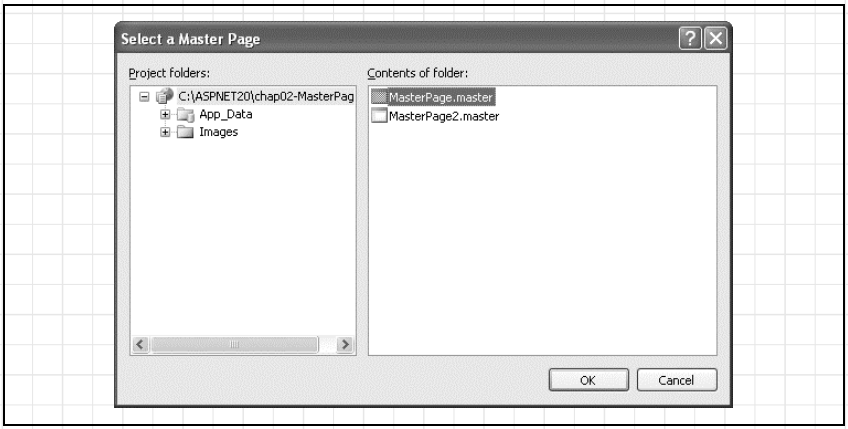
1. Po pierwsze, utwórz kilka stron na swojej witrynie internetowej. Używając projektu, który powstał w poprzednim ćwiczeniu, dodaj nowy formularz Web Form. Kliknij prawym przyciskiem myszy nazwę projektu w oknie *Solution Explorer* i wybierz z menu opcję *Add New Item...*. W oknie dialogowym *Add New Item* zaznacz formularz Web Form, a następnie użyj jego domyślnej nazwy (*Default2.aspx*). Upewnij się, że zostało zaznaczone pole *Select master page* w dolnej części okna dialogowego (rysunek 2.7).



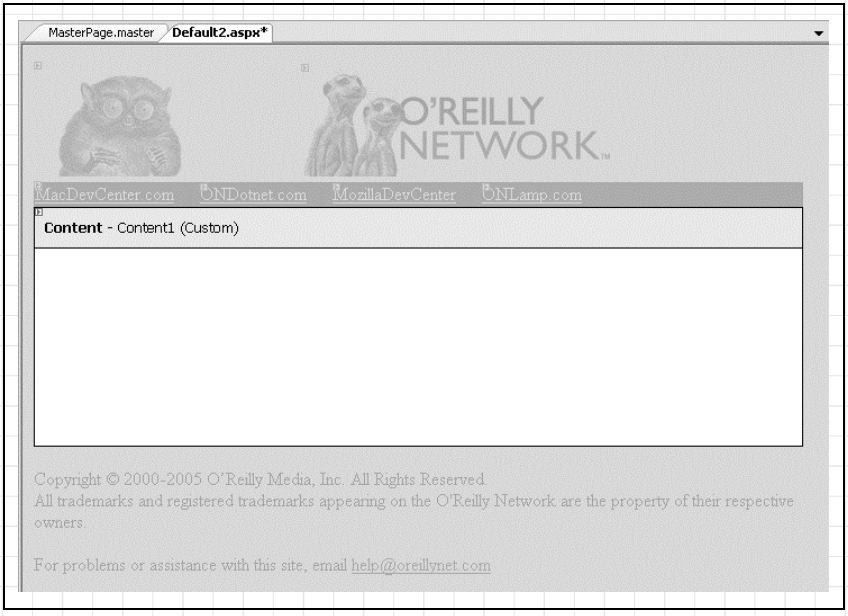
Rysunek 2.7. Tworzenie strony z treścią przez zaznaczenie strony wzorcowej

Strona z treścią może posiadać tylko jedną stronę wzorcową.

2. Zostaniesz poproszony o wybór strony wzorcowej, której chcesz użyć z formularzem. Zaznacz *MasterPage.master* (rysunek 2.8).
3. Kliknij przycisk *OK*, a Visual Studio wyświetli nową stronę, *Default2.aspx*, z zaznaczoną na szaro zawartością strony wzorcowej (rysunek 2.9). Wskazuje to, że zawartość strony wzorcowej nie może być



Rysunek 2.8. Zaznaczanie strony wzorcowej



Rysunek 2.9. Tworzenie nowej strony z treścią

edytowana w formularzu *Default2.aspx*. Zauważ, że strona została utworzona przy użyciu kontrolki `Content`.

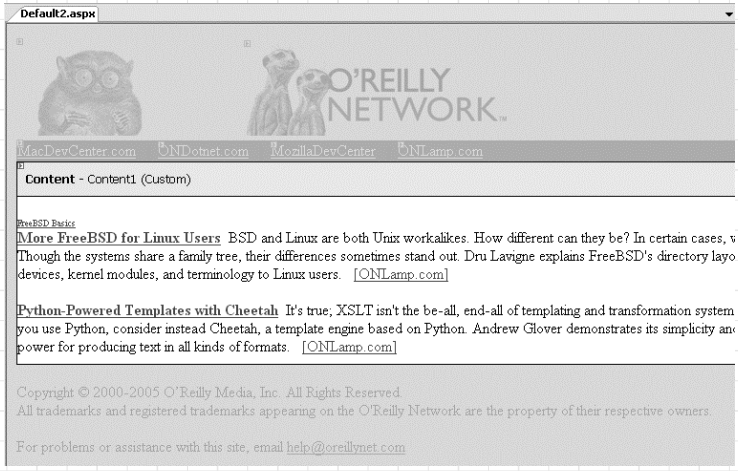
Kontrolka Content (zawartość) jest miejscem, które możesz zapełnić treścią strony.

4. Możesz teraz dostosować nową stronę *Default2.aspx* do swoich potrzeb poprzez dodanie kontrolek do kontrolki `Content`. Zwróć uwagę, że nie możesz modyfikować strony wzorcowej na tej stronie.

WSKAZÓWKA

Aby przeprowadzić edycję strony wzorcowej, możesz albo kliknąć prawym przyciskiem myszy w obszarze zawartości strony wzorcowej i wybrać opcję *Edit Master*, albo po prostu przejść do okna *Solution Explorer* i dwukrotnie kliknąć stronę wzorcową. Każde z tych rozwiązań spowoduje wczytanie strony wzorcowej w trybie do edycji.

5. Dodanie zawartości do kontrolki `Content` to po prostu przeciągnięcie i upuszczenie na nią odpowiednich kontrolek. Możesz również wpisywać dane bezpośrednio do kontrolki `Content`. Spróbuj dodać pewien tekst do kontrolki `Content` (rysunek 2.10).

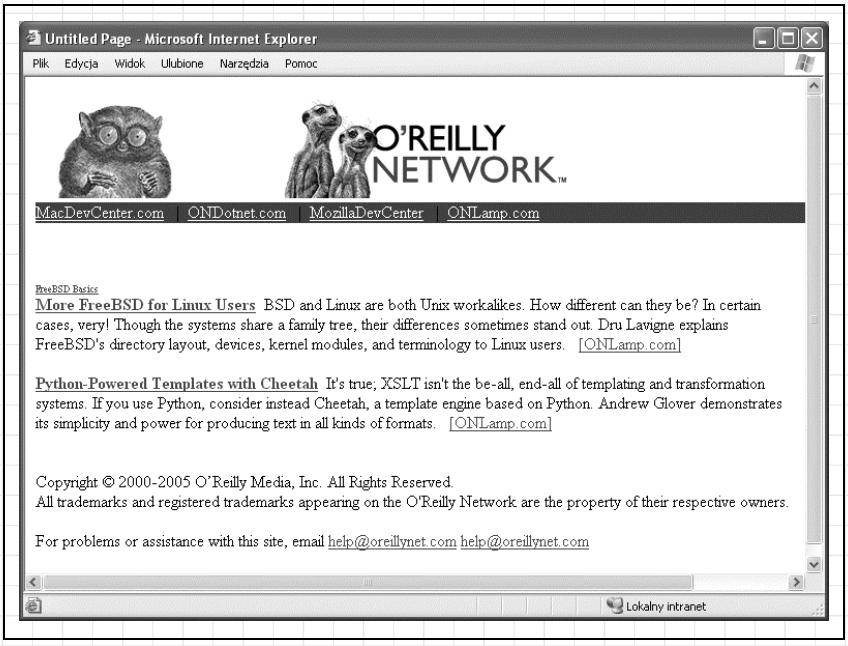


Rysunek 2.10. Zapełnianie strony treścią

6. W celu przetestowania strony naciśnij klawisz *F5*. Twoja strona powinna wyglądać jak ta pokazana na rysunku 2.11.

WSKAZÓWKA

Przypominam, że strona wzorcowa zawiera kontrolkę `ContentPlaceHolder`. Jeśli chcesz, możesz umieścić treść w kontrolce `ContentPlaceHolder` na stronie wzorcowej. Jednakże jeśli to zrobisz, zawartość wewnątrz `ContnetPlaceHolder` będzie pojawiać się na stronie, kiedy ta zostanie wczytana, dopóki strona z treścią nie nadpisze jej swoją własną kontrolką `Content`.



Rysunek 2.11. Wyświetlenie strony z zawartością w przeglądarce Internet Explorer

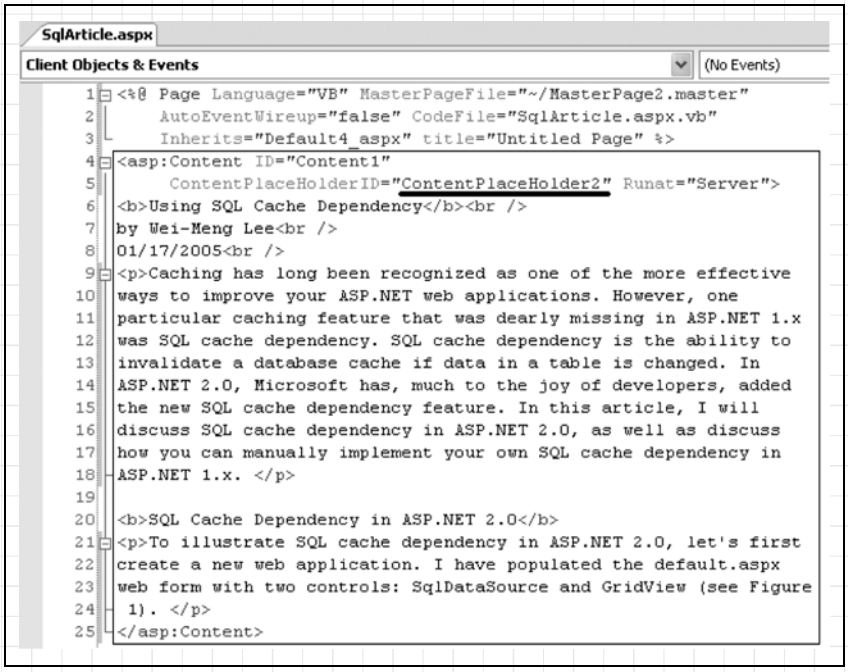
Jak to działa?

To, co utworzyłeś jest formularzem Web Form (*Default2.aspx*), który dziedziczy ze strony wzorcowej (*MasterPage.master*). Korzyść z używania stron wzorcowych polega na tym, że możesz utworzyć szablon zawartości strony, a inne formularze Web Form mogą z niego dziedziczyć. Jeżeli zawartość strony wzorcowej zostanie zmieniona, te zmiany zostaną również automatycznie uwzględnione na każdej stronie, która z niej dziedziczy.

Zauważ, że jeżeli strona korzysta z zagnieżdżonej strony wzorcowej takiej jak *MasterPage2.master*, nie będziesz mógł jej wyświetlić w widoku *Design View* (ponieważ wizualna edycja zagnieżdżonych stron wzorcowych nie jest obsługiwana w Visual Studio 2005). Zagnieżdżoną stronę wzorcową możesz edytować jedynie w widoku *Source View*.

Aby zobaczyć, jak wygląda strona, która korzysta z zagnieżdżonej strony wzorcowej, dodaj do projektu nowy formularz Web Form (nazwij go `SqlArticle.aspx`) i zaznacz *MasterPage2.master* jako jej stronę wzorcową.

Zapełnij stronę *SqlArticle.aspx*, wprowadzając kod HTML i tekst pokazany na rysunku 2.12. Zwróć uwagę, że musisz ręcznie wypełnić kontrolkę `Content`.



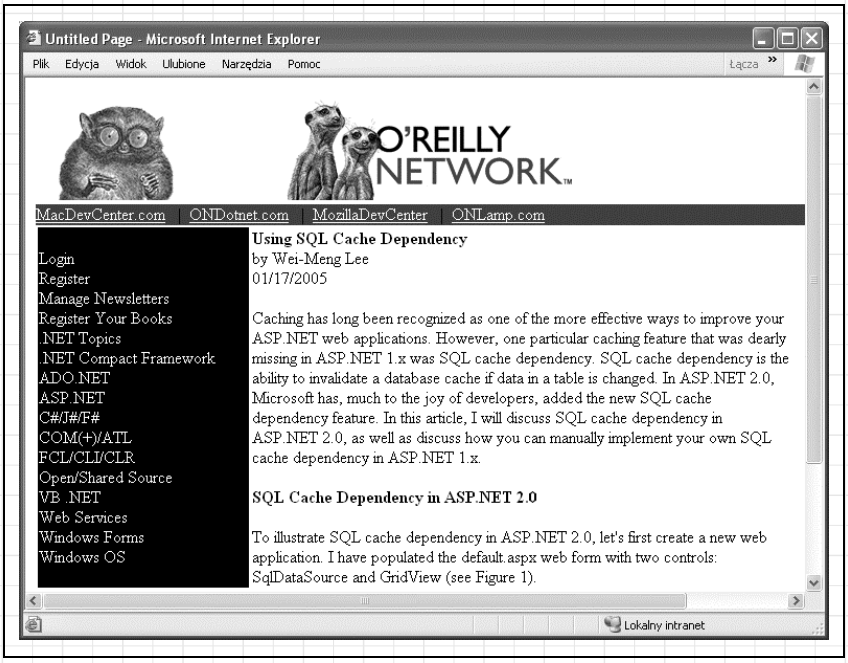
Rysunek 2.12. Edycja w widoku Source View strony, która korzysta z zagnieżdżonej strony wzorcowej

Kiedy strona zostanie wczytana w przeglądarce Internet Explorer (po naciśnięciu klawisza *F5*), zobaczysz połączoną zawartość dwóch stron wzorcowych (rysunek 2.13).

A co...

...z przekształceniem istniejących stron internetowych na strony z zawartością?

Gdy spojrzysz na kod źródłowy normalnego formularza Web Form (takiego, który nie korzysta ze strony wzorcowej), zauważysz, że zawiera on zarówno zwykłe fragmenty kodu HTML, które oczekujesz znaleźć na stronie ASP.NET, jak i element `<form>`:



Rysunek 2.13. Wczytywanie strony, która korzysta z zagnieżdżonej strony wzorcowej

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
Inherits="Default_aspx" %>

<!DOCTYPE html PUBLIC "-//WW3C//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">
   <title>Untitled Page</title>
</head>
<body>
   <form id="form1" runat="server">
   <div>

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

Jednakże strona z treścią nie zawiera zwykłych elementów, które czynią ją stroną ASP.NET (takich jak `<html>`, `<body>` lub `<form>`). Zamiast tego zobaczysz element `<asp:Content>` przedstawiający kontrolkę `Content` i jej właściwości. Kontrolka `<asp:Content>` zawiera wszelkie kontrolki (i zawartość), które upuszczasz na kontrolkę `Content`.

```
<%@ Page Language="VB" MasterPageFile="~/MasterPage.master"
AutoEventWireup="false" CodeFile="Default3.aspx.vb"
Inherits="Default3_aspx" title="Untitled Page"
%>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1"
Runat="server">
...
...
</asp:Content>
```

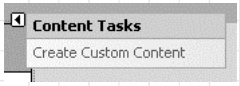
Aby przekształcić formularz Web Form na stronę z zawartością, musisz po prostu dodać atrybut `MasterPageFile` do jej dyrektywy `Page`, a następnie usunąć inne elementy HTML ze strony. Oto przykład dyrektywy `Page` wykonującej to zadanie:

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

Dyrektywa `Page` przekształca stronę *Default.aspx* z tego ćwiczenia na stronę z zawartością. A co z kontrolką `Content`, która jest potrzebna do wyświetlenia unikatowej zawartości strony? Możesz ją dodać w kodzie, na przykład w taki sposób:

```
<%@ Page Language="VB" MasterPageFile="~/MasterPage.master"
AutoEventWireup="false" CodeFile="Default.aspx.vb"
Inherits="Default_aspx" %>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1"
   runat="server">
</asp:Content>
```

Ewentualnie, gdy klikniesz odnośnik *Create Custom Content* w menu *Content Tasks*, które znajduje się po prawej stronie kontrolki `Content` (rysunek 2.14).



Rysunek 2.14. Tworzenie nowej kontrolki Content

WSKAZÓWKA

Jeżeli nie umieścisz kontrolki `Content` w kodzie, wciąż będziesz widział kontrolkę `Content` w widoku *Design View*. Nie będziesz w stanie dodać kontrolek do kontrolki `Content`, dopóki nie klikniesz odnośnika *Create Custom Content* w menu *Content Task* kontrolki `Content`.

...z zastosowaniem strony wzorcowej dla całej witryny?

Aby wszystkie strony na witrynie domyślnie użyły określonej strony wzorcowej, po prostu dodaj element `<pages>` do pliku *Web.config* dla danej witryny:

```
<system.web>
   <pages masterPageFile="MasterPage.master" />
   ...
```

Strona, która nie ma określonej wartości atrybutu `MasterPageFile` w swojej dyrektywie `Page`, będzie domyślnie używała strony wzorcowej *MasterPage.master*. W celu unieważnienia domyślnego ustawienia strony wzorcowej w pliku *Web.config*, po prostu dodaj atrybut `MasterPageFile` do dyrektywy `Page` strony internetowej, o której mowa i przypisz jej nazwę do innego pliku.

...z wykrywaniem typu przeglądarki i przełączaniem stron wzorcowych w locie?

Jeżeli Twoja aplikacja jest skierowana do użytkowników różnych przeglądarek, możesz utworzyć oddzielne strony wzorcowe, aby każda z nich była zoptymalizowana dla określonej przeglądarki. W takim przypadku będziesz musiał być w stanie dynamicznie przełączać swoje strony wzorcowe w trakcie działania. Realizacja takiego zadania wymaga dodania następującego kodu w zdarzeniu `PreInit` strony z zawartością:

```
Protected Sub Page_PreInit(ByVal sender As Object, _
      ByVal e As System.EventArgs) _
      Handles Me.PreInit
   If (Request.Browser.IsBrowser("IE")) Then
      MasterPageFile = "MasterPage.master"
   ElseIf (Request.Browser.IsBrowser("Mozilla")) Then
      MasterPageFile = "MasterPage_FireFox.master"
   Else
      MasterPageFile = "MasterPage_Others.master"
   End If
End Sub
```

W powyższym kodzie zakłada się, że posiadasz trzy różne strony wzorcowe: *MasterPage.master* (dla przeglądarki Internet Explorer), *MasterPage_FireFox.master* (dla przeglądarki Firefox) oraz *MasterPage_Others.master* (dla innych przeglądarek). Na stronie sprawdza się typ przeglądarki, a następnie stosuje odpowiednią stronę wzorcową.

Poza przełączaniem programowym strony wzorcowej istnieje również taki sposób:

```
<%@ Page Language="VB"
   MasterPageFile="~/MasterPage_Others.master"
   ie:MasterPageFile="~/MasterPage.master"
   mozilla:MasterPageFile="~/MasterPage_FireFox.master"
   ...
```

Więcej informacji

Scott Guthrie (Product Manager ASP.NET w Microsofcie) napisał blog o tym, w jaki sposób zmieniać strony wzorcowe w zależności od typu przeglądarki. Tekst (w języku angielskim) jest dostępny pod adresem: *http://blogs.msdn.com/scottgu/archive/2004/11/20/267362.aspx*.

Modyfikacja strony wzorcowej w trakcie działania

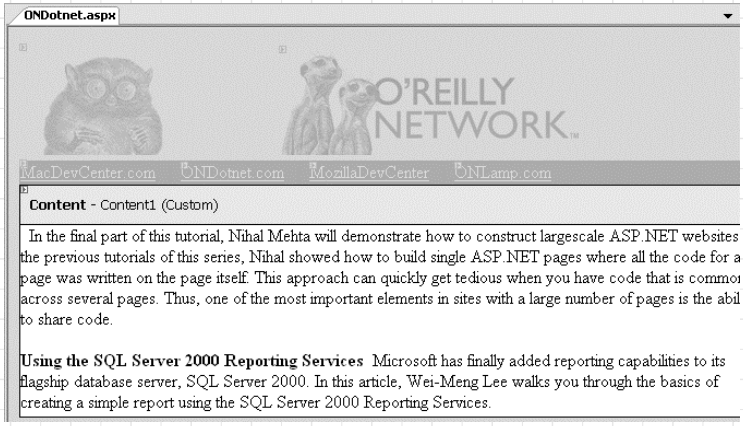
Możesz modyfikować zawartość strony wzorcowej, tak aby pasowała tematem do wyświetlanej strony.

Kiedy formularz Web Form korzystający ze strony wzorcowej jest wczytywany w trakcie działania, wyświetla zawartość strony wzorcowej razem z własną zawartością. Jednakże zdarzają się sytuacje, gdy chcesz zmodyfikować część strony wzorcowej w trakcie wczytywania określonej strony internetowej. Na przykład, na witrynie O'Reilly Network (*http://www.oreillynet.com/*) na stronach należących do podwitryny ONDotnet wyświetla się w nagłówkach logo ONDotnet (skaczący delfin) zamiast podstawowego logo O'Reilly Network. Na stronach należących do innych podwitryn, np. Perl.com czy ONLamp.com, jest podobnie.

Jak to osiągnąć?

Aby dowiedzieć się, w jaki sposób strona wzorcowa może być modyfikowana w locie, dodamy stronę do naszego poprzedniego projektu (zobacz podrozdział „Użycie strony wzorcowej jako szablonu strony z informacjami"). Strona ta będzie używać *MasterPage.master* i zmieni po załadowaniu obrazek strony wzorcowej z logo O'Reilly Network na obrazek z logo ONDotnet.

1. Do projektu utworzonego w poprzednim ćwiczeniu dodaj nowy formularz Web Form (kliknij prawym przyciskiem myszy nazwę projektu w oknie *Solution Explorer*, wybierz opcję *Add New Item...*, a następnie zaznacz *Web Form*). Upewnij się, że zostało zaznaczone pole *Select master page* w dolnej części okna dialogowego. Formularzowi Web Form nadaj nazwę *ONDotnet.aspx* i kliknij przycisk *Add*.
2. Zaznacz *MasterPage.master* jako stronę wzorcową.
3. Zapełnij formularz Web Form *ONDotnet.aspx* tekstem (rysunek 2.15).



Rysunek 2.15. Formularz Web Form ONDotnet.aspx z dołączonym tekstem

4. Przełącz się do widoku *Source View* i za dyrektywą `Page` wstaw dyrektywę `MasterType`. Dyrektywa `MasterType` pozwala Ci na wskazanie ściśle określonego odwołania do strony wzorcowej, zatem możesz uzyskać dostęp do jej zawartości (takich jak publiczne właściwości i metody):

```
<%@ Page Language="VB" MasterPageFile="~/MasterPage.master"
   AutoEventWireup="false" CodeFile="OnDotnet.aspx.vb"
   Inherits="ONDotnet_aspx" title="Untitled Page" %>
<%@ MasterType VirtualPath="~/MasterPage.master" %>
...
```

5. W ukrytym kodzie strony wzorcowej (*MasterPage.master.vb*) wstaw publiczną metodę `setLinkColor()`. Dzięki dodaniu do strony wzorcowej tej publicznej metody będziesz mógł na stronach z informacjami programowo ustawić kolor różnych kontrolek `LinkButton`:

```
Imports System.Drawing
Partial Class MasterPage_master
   Inherits System.Web.UI.MasterPage

   Public Sub setLinkColor(ByVal ctrl As String, _
         ByVal lnkColor As Color)
      Select Case ctrl
         Case "MacDevCenter"
            lnkMacDevCenter.ForeColor = lnkColor
         Case "ONDotnet"
            lnkONDotnet.ForeColor = lnkColor
         Case "MozillaDevCenter"
            lnkMozillaDevCenter.ForeColor = lnkColor
         Case "ONLamp"
            lnkONLamp.ForeColor = lnkColor
      End Select
   End Sub
End Class
```

Będziesz musiał użyć modyfikatora public przy definiowaniu metody setLinkColor(), w przeciwnym razie strona z informacjami nie będzie w stanie go wywołać.

6. W ukrytym kodzie dla *ONDotnet.aspx* (*ONDotnet.aspx.vb*) umieść kod następującego zdarzenia `Page_Load`:

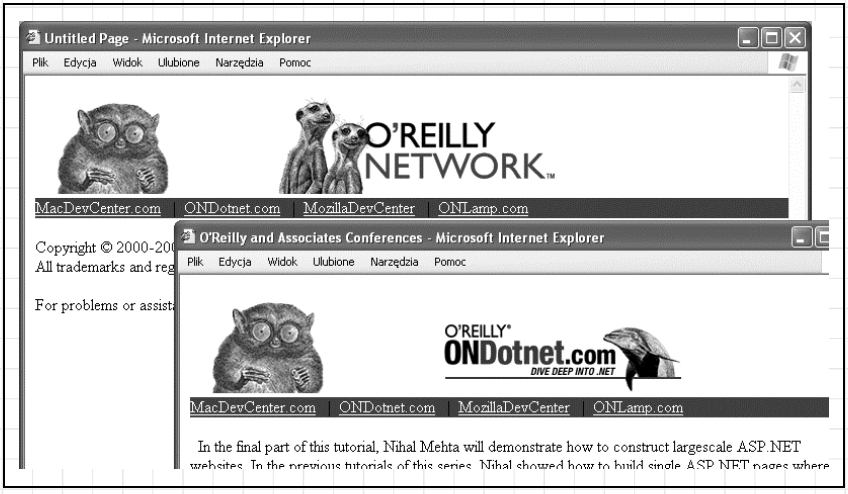
```
Sub Page_Load(ByVal sender As Object, _
      ByVal e As System.EventArgs) _
      Handles Me.Load
   '---Zmień kolor odnośnika.
   Master.setLinkColor("ONDotnet", Drawing.Color.Yellow)

   '---Zmień tytuł.
   Master.Page.Header.Title = _
      "Konferencje O'Reilly i partnerów"

   '---Zmień obrazek.
   Dim masterImage As Image
   masterImage = CType(Master.FindControl("imgTitle"), _
         Image)
   If Not (masterImage Is Nothing) Then
      masterImage.ImageUrl = "Images/ondotnet_logo.gif"
   End If
End Sub
```

7. Użyjesz właściwości `Master`, aby pobrać odwołanie do strony wzorcowej. W pierwszej kolejności zmienisz kolor kontrolki ONDotnet.com, `LinkButton`, przez wywołanie metody `setLinkColor()`. Następnie przy użyciu właściwości *Master.Page.Header.Title* zmienisz tytuł strony. Na końcu zmienisz obrazek logo na stronie wzorcowej, korzystając z metody `FindControl()`.

WSKAZÓWKA

Właściwość `Master` jest specjalną właściwością ujawnioną przez formularz Web Form, ponieważ obsługuje dostęp do strony wzorcowej. W rzeczywistości każda strona może uzyskać dostęp do właściwości `Master`, jednakże jest to sensowne jedynie w przypadku, gdy strona korzysta ze strony wzorcowej.

8. Naciśnij klawisz *F5* i przetestuj aplikację. W pierwszej kolejności załaduj *Default2.aspx*, a następnie kliknij odnośnik do ONDotnet.com, by wczytać *ONDotnet.aspx*. Na rysunku 2.16 pokazano różnice między tymi dwiema stronami.



Rysunek 2.16. Wyświetlenie stron Default2.aspx oraz ONDotnet.aspx

Jak to działa?

Istnieją dwa sposoby modyfikowania zawartości strony wzorcowej w trakcie działania. Możesz to osiągnąć:

- Wywołując publiczne metody i ustawiając publiczne właściwości strony wzorcowej.
- Używając metody `FindControl()` (właściwości `Master`) do odnalezienia kontrolek na stronie, a następnie wywołując metody i ustawiając właściwości tych kontrolek.

OSTRZEŻENIE

Właściwość `Master` jest poprawna jedynie na stronach, które odwołują się do strony wzorcowej w atrybucie `MasterPageFile` dyrektywy `Page`. Jeśli spróbujesz odwołać się do właściwości `Master` na stronie, która nie zawiera odwołania do strony wzorcowej, to zostanie zgłoszony wyjątek `NullReferenceException`.

Gdzie tylko jest to możliwe, powinieneś używać pierwszej metody, która ujawnia publiczne właściwości i metody strony wzorcowej. Takie podejście jest bardziej efektywnym sposobem zmiany kontrolek na stronie wzorcowej, ponieważ korzysta się tu z wczesnych wiązań i przez to kontrolki są ściśle określone.

Użycie wczesnych wiązań, gdy tylko jest to możliwe, znacznie upraszcza programowanie i ogranicza możliwość powstawania błędów.

By mogło dojść do użycia wczesnych wiązań, strona wzorcowa musi ujawnić swoje publiczne metody i właściwości (jak w kroku 5.), tak aby strona z treścią mogła je ustawić lub uzyskać do nich dostęp w trakcie działania. W naszym ćwiczeniu ujawniłeś publiczną metodę o nazwie `setLinkColor()` na stronie *MasterPage.master*. Zmiana koloru kontrolki `LinkButton` na stronie wzorcowej w trakcie działania wymaga wywołania metody `setLinkColor()` w zdarzeniu `Page_Load` strony z treścią.

W przypadku drugiej metody odnajdujesz kontrolki, które chcesz modyfikować na stronie wzorcowej przy użyciu metody `FindControl()` właściwości `Master`. Następnie dostarczasz nazwę i typ kontrolki przeznaczonej do modyfikacji. Gdy kontrolka zostanie zlokalizowana, możesz zmienić jej właściwości tak, jakby była ona obiektem lokalnym. Technika ta została przedstawiona w poniższym fragmencie kodu, który lokalizuje kontrolkę `Image` na stronie wzorcowej i zastępuje ją nowym plikiem *.gif* zawierającym logo ONDotnet.

```
Dim masterImage As Image
masterImage = CType(Master.FindControl("imgTitle"), _
      Image)
If Not (masterImage Is Nothing) Then
   masterImage.ImageUrl = "Images/ondotnet_logo.gif"
End If
```

Metoda FindControl() używa późnych wiązań, podczas gdy zastosowanie właściwości publicznych posiada zalety wczesnych wiązań i jest ściśle określone.

Technika ta wykorzystuje późne wiązania w uzyskaniu dostępu do kontrolek na stronie wzorcowej, ponieważ musisz wyraźnie przekonwertować kontrolki znalezione w trakcie działania na pożądany typ.

A co...

...z uzyskaniem dostępu do dynamicznie tworzonych kontrolek podczas ładowania strony wzorcowej?

Na przykład, strona wzorcowa może wyświetlać dodatkowe informacje z ostatniej chwili dzięki dynamicznemu tworzeniu kontrolki `Label`. Liczba generowanych nowych kontrolek jest uzależniona od ilości informacji z ostatniej chwili.

Zaletą uczynienia metod i właściwości publicznymi na stronie wzorcowej jest to, że możesz użyć wczesnych wiązań, co oznacza możliwość skorzystania z pomocy mechanizmu *IntelliSense* podpowiadającego ich nazwy oraz polegania na kompilatorze sprawdzającym ich typy w trakcie kompilacji.

Ujawnienie publicznych właściwości nie jest możliwe dla kontrolek, które są generowane dynamicznie na stronie wzorcowej. Dzieje się tak, ponieważ musisz zdecydować, które metody i właściwości ujawnić w trakcie projektowania.

W takim przypadku powinieneś uciec się do późnych wiązań, korzystając z opisanej wcześniej w punkcie „Jak to działa?" metody `FindControl()`.

Więcej informacji

Aby dowiedzieć się więcej na temat wczesnych i późnych wiązań oraz zalet i wad każdego z tych rozwiązań, przejrzyj następujący temat pomocy w bibliotece MSDN: *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbcn7/html/vaconearlylatebinding.asp*.

Twoi użytkownicy nie zgubią się nigdy więcej, jeśli utworzysz mapę witryny dla swojej witryny internetowej!

Tworzenie mapy witryny dla swojej witryny internetowej

Jeśli Twoja witryna internetowa nie zawiera tylko jednej strony internetowej, musisz znaleźć sposób na umożliwienie użytkownikom nawigacji pomiędzy stronami na witrynie. W ASP.NET 2.0 znajdują się cztery nowe

kontrolki, które ułatwiają wdrożenie nawigacji na witrynie. Tymi kontrolkami są:

`SiteMapDataSource`
: Kontrolka, która przedstawia mapę witryny. Mapa witryny jest plikiem XML, który opisuje logiczną strukturę witryny internetowej oraz związki pomiędzy stronami. Kontrolka `SiteMapDataSource` nie jest widoczna po załadowaniu strony. Jej rolą jest działanie jako dostawca informacji o witrynie (pobranych z pliku mapy witryny) do kontrolki `SiteMapPath`.

`SiteMapPath`
: Wizualna kontrolka przedstawiająca ścieżkę do bieżącej strony.

`Menu`
: Wizualna kontrolka wyświetlająca informacje w formie menu. Kontrolka `Menu` obsługuje wiązanie danych do mapy witryny.

`TreeView`
: Wizualna kontrolka wyświetlająca informacje w hierarchicznej strukturze. Kontrolka `TreeView` obsługuje wiązanie danych do mapy witryny.

W pozostałych ćwiczeniach tego rozdziału nauczysz się, jak używać tych kontrolek, aby ułatwić użytkownikom nawigację na Twojej witrynie. W pierwszej kolejności przyjrzyjmy się bliżej kontrolkom `SiteMapDataSource` oraz `SiteMapPath`.

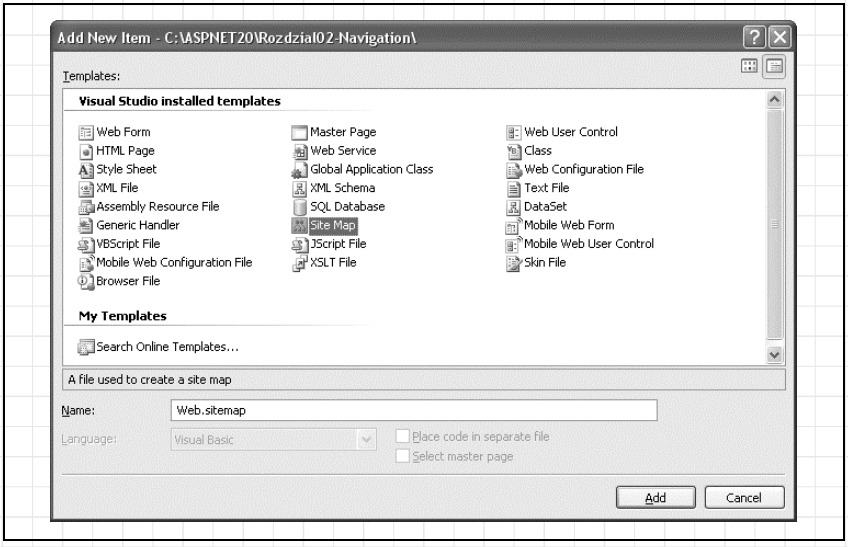
Jak to osiągnąć?

W tym ćwiczeniu dowiesz się, w jaki sposób użyć kontrolki `SiteMapPath` wraz z plikiem *Web.sitemap* do wyświetlenia ścieżek nawigacji na stronach Twojej witryny. Po pierwsze, utworzysz aplikację sieciową korzystającą ze strony wzorcowej. Kiedy strona wzorcowa jest używana jako szablon dla wszystkich innych stron na witrynie, jest ona dobrym miejscem do umieszczenia mapy witryny i kontrolek, które pozwalają użytkownikom na odnalezienie się. Dzieje się tak, ponieważ tego typu informacja będzie widoczna na każdej stronie odwiedzanej przez użytkownika. Utworzysz plik mapy witryny opisujący logiczne związki pomiędzy stronami witryny, a następnie dodasz kontrolkę `SiteMapPath` do strony wzorcowej.

Pozwoli to użytkownikom zobaczyć, w którym miejscu w hierarchii się znajdują oraz przenosić się „w górę" i „w dół" ścieżki ich bieżącej strony.

Mapa witryny jest dokumentem XML opisującym logiczny rozkład witryny internetowej.

1. Uruchom Visual Studio 2005 i utwórz nową aplikację sieciową ASP.NET 2.0. Zapisz ten projekt pod nazwą: *C:\ASPNET20\Rozdzial02-Navigation*.
2. W oknie *Solution Explorer* dołącz mapę witryny do aplikacji przez kliknięcie prawym przyciskiem myszy nazwy projektu, a następnie wybranie z menu opcji *Add New Item/Site Map* (rysunek 2.17).



Rysunek 2.17. Dołączenie mapy witryny do projektu

3. Zapełnij plik *Web.sitemap* — otwórz go i wpisz kod przedstawiony na listingu 2.2.

WSKAZÓWKA

Pamiętaj o sprawdzeniu, czy Twój dokument XML *Web.sitemap* jest dobrze zbudowany (tzn. czy dokument odpowiada zasadom XML-a). Najłatwiejszym sposobem przeprowadzenia weryfikacji jest załadowanie dokumentu w przeglądarce Internet Explorer — jeżeli nie będzie dobrze zbudowany, przeglądarka Cię o tym poinformuje.

Listing 2.2. Plik Web.sitemap

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap>
   <siteMapNode url="main.aspx"
      title="Strona główna Windows Mobile"
      description=" Strona główna Windows Mobile " roles="">
      <siteMapNode url="whatiswinmobile.aspx"
         title="Czym jest Windows Mobile"
         description=" Czym jest Windows Mobile " roles="">
      <siteMapNode url="windowsmobileoverview.aspx"
         title="Ogólny opis Windows Mobile"
         description=" Ogólny opis Windows Mobile"
         roles="" />
      <siteMapNode url="comparedevices.aspx"
         title="Porównaj urządzenia"
         description=" Porównaj urządzenia " roles="" />
      <siteMapNode url="software.aspx"
         title="Poznaj oprogramowanie"
         description=" Poznaj oprogramowanie "
         roles="" />
      <siteMapNode url="faq.aspx"
         title="FAQ"
         description=" FAQ "
         roles="" />
   </siteMapNode>

   <siteMapNode url="devices.aspx" title="Urządzenia"
      description=" Urządzenia " roles="" />
      <siteMapNode url="downloads.aspx" title="Centrum pobierania"
         description=" Centrum pobierania " roles="" />
      <siteMapNode url="howto.aspx"
         title="Pomoc"
         description=" Pomoc " roles="" />
      <siteMapNode url="communities.aspx"
         title="Społeczność"
         description=" Społeczność " roles="" />
      <siteMapNode url="events.aspx"
         title="Nowości i wydarzenia"
         description=" Nowości i wydarzenia" roles="" />
      <siteMapNode url="Administrator/Admin.aspx"
         title="Administrator"
         description=" Administrator " roles="" />
   </siteMapNode>
</siteMap>
```

Plik *Web.sitemap* po prostu opisuje logiczny rozkład witryny. Nie ma zupełnie znaczenia, gdzie fizycznie są przechowywane różne pliki. Zwróć uwagę, że schemat dokumentu jest prosty, elementy są zagnieżdżone i logicznie pogrupowane.

4. Każdy element `<siteMapNode>` posiada cztery atrybuty:

`title`

Tekst wyświetlany w każdym węźle kontrolki `SiteMapPath`, `TreeView` lub `Menu`.

`description`

Tekst pomocy wyświetlany po najechaniu myszą na węzeł w kontrolce `SiteMapPath`, `TreeView` lub `Menu`.

`roles`

Rola lub role, dla których dany węzeł jest wyświetlany.

`url`

Adres URL, do którego będziemy przeniesieni po kliknięciu węzła w kontrolce `SiteMapPath`, `TreeView` lub `Menu`.

WSKAZÓWKA

Zwróć uwagę, że wartości każdego atrybutu `url` w elemencie `<siteMapNode>` muszą być unikatowe.

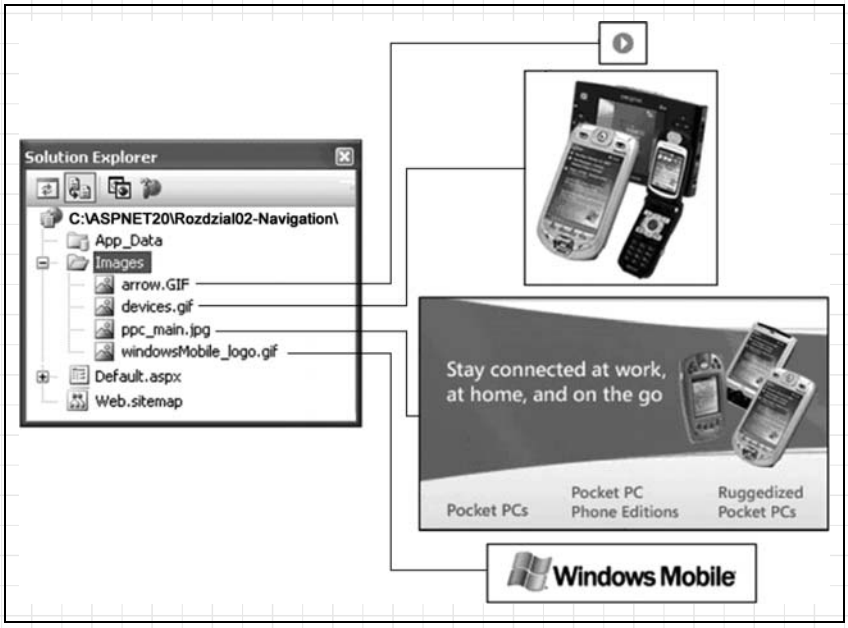
---

5. Dodaj nowy katalog do projektu i nazwij go `Images` (kliknij prawym przyciskiem myszy nazwę projektu w oknie *Solution Explorer* i wybierz *Add Folder/Regular Folder*). Skopiuj obrazki pokazane na rysunku 2.18 do katalogu *C:\ASPNET20\Rozdzial02-Navigation\Images*. Wykorzystasz je na stronie wzorcowej dla tej witryny.

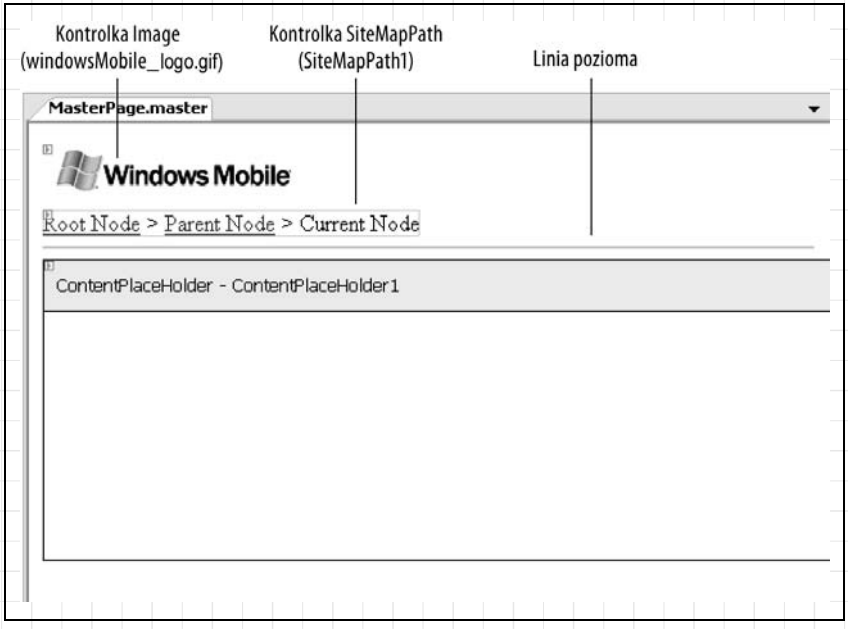
6. Następnym krokiem jest dołączenie nowej strony wzorcowej do projektu. Zapełnij stronę wzorcową kontrolkami pokazanymi na rysunku 2.19. Strona wzorcowa jest idealnym miejscem do umieszczenia kontrolki `SiteMapPath`, ponieważ wszystkie kontrolki umieszczone na tej stronie są widoczne na każdej stronie, która z niej dziedziczy. W rezultacie otrzymasz ścieżkę nawigacji witryny, wyświetloną przez kontrolkę `SiteMapPath` i widoczną na wszystkich stronach.

7. Dodaj nowy formularz Web Form do projektu i nazwij go `Main.aspx`. Upewnij się, że opcja *Select master page* została zaznaczona i wybierz stronę wzorcową utworzoną w poprzednim kroku (rysunek 2.20).

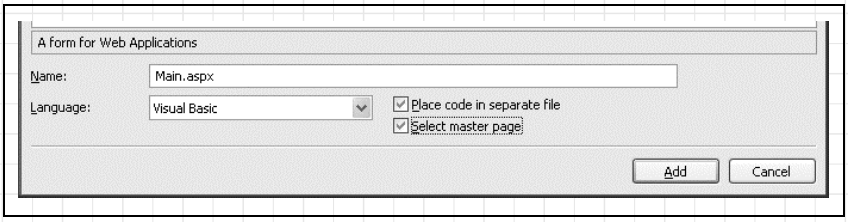
8. Zapełnij stronę *Main.aspx* zawartością pokazaną na rysunku 2.21.



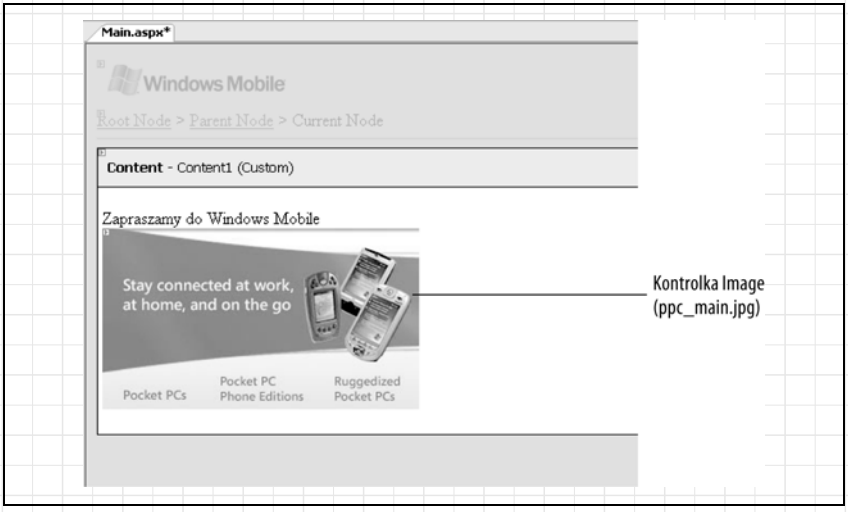
Rysunek 2.18. Obrazki w katalogu Images



Rysunek 2.19. Zapełnienie strony wzorcowej



Rysunek 2.20. Wybór strony wzorcowej dla formularza Web Form



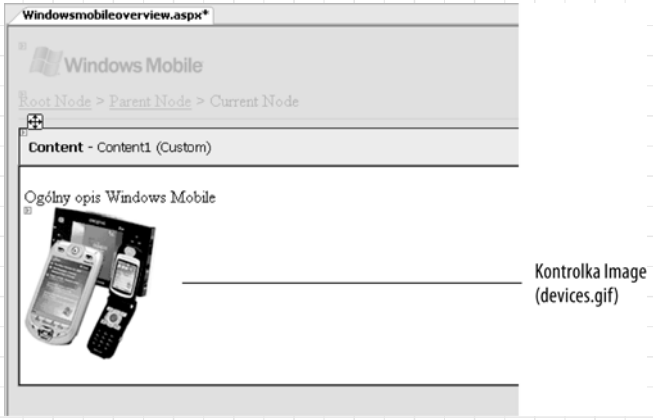
Rysunek 2.21. Strona Main.aspx

9. Następnie dodaj inny formularz Web Form do witryny i nazwij go `Windowsmobileoverview.aspx`. Zapełnij formularz zawartością pokazaną na rysunku 2.22.

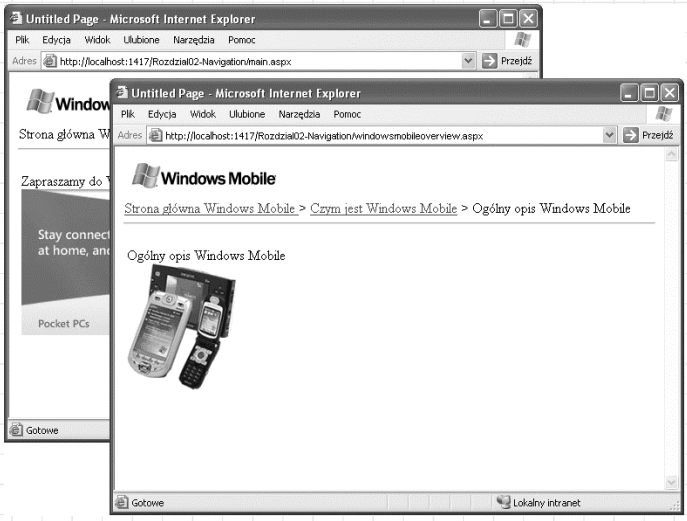
10. Aby przetestować strony, naciśnij klawisz *F5* i wczytaj stronę *Main.aspx* oraz *Windowsmobileoverview.aspx*. Powinna być widoczna ścieżka nawigacyjna pokazana na rysunku 2.23.

Jak to działa?

Kontrolka `SiteMapPath` domyślnie odczytuje zawartość pliku *Web.sitemap* i wyświetla ścieżkę nawigacyjną witryny prowadzącą do aktualnie wyświetlanej strony.



Rysunek 2.22. Strona Windowsmobileoverview.aspx



Rysunek 2.23. Testowanie kontrolki SiteMapPath

WSKAZÓWKA

Pamiętaj, że kontrolka `SiteMapPath` wyświetla swoją informację o ścieżce na podstawie tego, co znajduje się w pliku *Web.sitemap*. Nie musi to koniecznie oznaczać, że taka jest fizyczna organizacja witryny. Jeżeli zawartość pliku *Web.sitemap* nie odzwierciedla Twojej aktualnej struktury witryny, informacje wyświetlane przez kontrolkę `SiteMapPath` będą również nieprawidłowe.

Zauważ, że w ASP.NET 2.0 jest dostępny tylko jeden dostawca mapy witryny: `XmlSiteMapProvider`. Odczytuje on informacje na temat mapy witryny ze zdefiniowanego dokumentu XML, którym domyślnie jest plik *Web.sitemap*.

Jeśli musisz zmienić domyślną nazwę pliku mapy witryny, możesz to zrobić poprzez plik *Web.config*. Listing 2.3 pokazuje, w jaki sposób możesz dokonać wyrejestrowania domyślnego `XmlSiteMapProvider` i ponownego zarejestrowania go z inną nazwą pliku mapy witryny: *My.sitemap* (możesz również zarejestrować nowego dostawcę mapy witryny, na przykład napisanego samodzielnie przez Ciebie).

Listing 2.3. Zmiana domyślnej mapy witryny w pliku Web.config

```
<configuration>
   <system.web>
      <siteMap>
         <providers>
            <remove name="AspNetXmlSiteMapProvider" />
            <add name="AspNetXmlSiteMapProvider"
                  type="System.Web.XmlSiteMapProvider,
                  System.Web, Version=1.2.3400.0,
                  Culture=neutral,
                  PublicKeyToken=b03f5f7f11d50a3a"
                  siteMapFile="My.sitemap" />
         </providers>
      </siteMap>
   </system.web>
</configuration>
```

W rozdziale 5. przeanalizujemy bardziej szczegółowo model dostawcy, który jest nowością w ASP.NET 2.0.

WSKAZÓWKA

Jeżeli utworzyłeś swój własny obiekt `SiteMapProvider` dostawcy mapy, zmień właściwość `SiteMapProvider` kontrolki `SiteMapPath` na nazwę Twojego obiektu.

A co...

...z dostosowaniem do własnych potrzeb kontrolki SiteMapPath?

Domyślny widok kontrolki `SiteMapPath` wyświetla ścieżkę nawigacyjną, używając łańcucha zawartego w tytule atrybutu elementu `<siteMapNode>`, który znajduje się w pliku mapy witryny. Każda ścieżka nawigacyjna jest rozdzielona przez znak >.

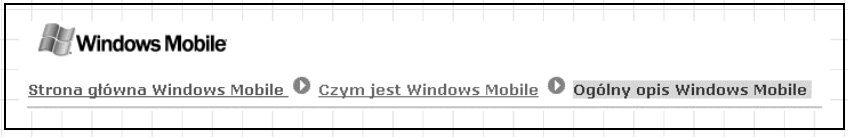
Możesz określić własny wygląd i działanie kontrolki SiteMapPath przez dodanie do niej informacji o stylu. Dostosowanie do własnych potrzeb kontrolki SiteMapPath przeprowadzasz, przełączając się do widoku *Source View* strony, która zawiera kontrolkę (w naszym przykładzie jest to strona wzorcowa).

Listing 2.4 przedstawia, w jaki sposób unieważnić domyślny znak rozdzielający i zamienić go na kontrolkę Image (użyty przez nas obrazek został zapisany w katalogu *Images* aplikacji i nosi nazwę *arrow.gif*). Kod powoduje również podświetlenie bieżącej ścieżki za pomocą jasnozielonego koloru tła.

Listing 2.4. Dostosowanie do własnych potrzeb kontrolki SiteMapPath

```
<asp:SiteMapPath ID="SiteMapPath2"
   Font-Name="garamond"
   Font-Size="12pt" RunAt="server">
   <CurrentNodeStyle Height="24px"
      BackColor="LightGreen"
      Font-Bold="true" />
   <NodeStyle Height="24px" />
   <PathSeparatorTemplate>
      <ItemTemplate>
         <asp:Image ID="Image2"
            ImageUrl="~/Images/arrow.gif"
            RunAt="server" />
      </ItemTemplate>
   </PathSeparatorTemplate>
</asp:SiteMapPath>
```

Na rysunku 2.24 zostało pokazane, jak wygląda kontrolka SiteMapPath po dokonanych przez nas modyfikacjach.



Rysunek 2.24. Dostosowanie do własnych potrzeb wyglądu i działania kontrolki SiteMapPath

Możesz również programowo zmodyfikować właściwości kontrolki SiteMapPath, korzystając z następujących właściwości:

PathDirection

Ustawia porządek, w jakim będą wyświetlane ścieżki. Domyślną wartością jest RootToCurrent — ścieżki wyświetlane są od najwyższej do bieżącej. Aby wyświetlić ścieżki w odwrotnej kolejności, ustaw wartość na CurrentToRoot.

PathSeparator

Ustawia znak rozdzielający ścieżki. Domyślnie jest to znak >.

RenderCurrentNodeAsLink

Wyświetla bieżącą ścieżkę jako odnośnik. Domyślnie ma ustawioną wartość False.

...z programowym uzyskaniem dostępu do informacji o mapie witryny?

Jeżeli nie używasz kontrolki SiteMapPath, możesz programowo uzyskać dostęp do informacji o mapie witryny przez *Site Map API* (znajdujące się w przestrzeni nazw System.Web.SiteMap). Listing 2.5 (umieszczony w ukrytym kodzie strony wzorcowej) przedstawia, w jaki sposób wyświetlić programowo nawigację witryny.

Listing 2.5. Programowe wyświetlanie informacji o witrynie

```
Partial Class MasterPage_master

   Sub Page_Load(ByVal sender As Object, _
         ByVal e As System.EventArgs) _
         Handles Me.Load

      Dim myCurrentNode As SiteMapNode

      '---Pobiera bieżący węzeł.
      myCurrentNode = SiteMap.CurrentNode

      '---Wyświetla ścieżkę nawigacyjną---
      Dim path As String = myCurrentNode.Title
      While myCurrentNode.ParentNode IsNot Nothing
         myCurrentNode = myCurrentNode.ParentNode
         path = " > " & path
         path = myCurrentNode.Title & path
      End While

      Response.Write(path)
   End Sub

End Class
```

Oprócz właściwości `CurrentNode` istnieją dodatkowe właściwości dające dostęp do informacji o mapie witryny. Są to:

- `SiteMap.CurrentNode.ParentNode`
- `SiteMap.CurrentNode.PreviousSibling`
- `SiteMap.CurrentNode.RootNode`
- `SiteMap.CurrentNode.NextSibling`

...z ukrywaniem węzłów przed użytkownikami, którzy nie zostali uwierzytelnieni?

Jeżeli dokładnie przeanalizujesz plik *Web.sitemap*, zauważysz w nim atrybut `role`. W naszym przykładzie atrybut `role` jest ustawiony jako pusty łańcuch. Domyślnie kontrolka `SiteMapDataSource` powoduje, że wszystkie informacje o węzłach w pliku *Web.sitemap* są widzialne. Jednakże zdarzają się przypadki, w których chciałbyś ograniczyć wyświetlanie określonych węzłów do określonej grupy użytkowników. Na przykład, zwykli użytkownicy nie powinni zobaczyć ścieżki do strony administracyjnej, tak więc kontrolka `SiteMapDataSource` powinna pozwolić na ukrycie (lub przycięcie) węzłów. Cecha ta jest znana jako *security trimming*. Obiekt dostawcy mapy `XmlSiteMapProvider` domyślnie wyłącza mechanizm *security trimming*. Aby włączyć tę opcję, musisz wykonać następujące kroki:

1. Ustawić dla atrybutu `securityTrimmingEnabled` wartość `True`. Wykonanie tego zadania wymaga wyrejestrowania, a następnie ponownego zarejestrowania dostawcy `AspNetXmlSiteMapProvider` w pliku *Web.config*.

```
<system.web>
   <siteMap>
      <providers>
         <remove name="AspNetXmlSiteMapProvider" />
         <add name="AspNetXmlSiteMapProvider"
               type="System.Web.XmlSiteMapProvider,
               System.Web, Version=1.2.3400.0,
               Culture=neutral,
               PublicKeyToken=b03f5f7f11d50a3a"
               securityTrimmingEnabled="true"
               siteMapFile="Web.sitemap" />
```

```
        </providers>
      </siteMap>
    </system.web>
```

2. Następnie określ węzeł, dla którego chcesz włączyć *security trimming*. Używając pliku *Web.sitemap* utworzonego w tym ćwiczeniu, ustaw dla atrybutu `role` wartość `admin` dla ostatniego węzła. W tym przypadku kontrolka `SiteMapPath` wyświetli węzeł `Administrator` (prowadzący do strony *Admin.aspx* umieszczonej w katalogu */Administrator*) jedynie wówczas, gdy użytkownik będzie należał do roli `admin`:

```
...
<siteMapNode url="Administrator/Admin.aspx"
    title="Administrator"
    description=" Administrator" roles="admin" />
</siteMapNode>
```

3. W ostatnim kroku musisz skonfigurować katalog */Administrator* tak, aby uniemożliwiał dostęp wszystkim użytkownikom, a następnie zezwolić na dostęp do strony *Admin.aspx* jedynie tym, którzy należą do roli `admin`. Wykonasz to zadanie, dodając plik *Web.config* zawierający kod przedstawiony na listingu 2.6 do katalogu */Administrator*.

Listing 2.6. Plik Web.config przeznaczony do katalogu Administrator

```
<configuration
xmlns="http://schemas.microsoft.com/.NetConfiguration/v2.0">
   <system.web>
      <compilation debug="false"/>
      <!-- blokuje dostęp wszystkim użytkownikom -->
      <authorization>
         <deny users="*" />
      </authorization>
   </system.web>

<!-- pozwala jedynie użytkownikom o roli admin na dostęp do strony
Admin.aspx -->
   <location path="Admin.aspx">
      <system.web>
         <authorization>
            <allow roles="Admin" />
         </authorization>
      </system.web>
   </location>
</configuration>
```

Zajrzyj do rozdziału 5., aby poznać więcej szczegółów dotyczących ról i członkostwa.

Kiedy już wprowadzisz te zmiany, jedynie użytkownicy należący do roli `admin` będą mieli dostęp do węzła `Administrator`.

Więcej informacji

Więcej informacji dotyczących autoryzacji w ASP.NET uzyskasz, zapoznając się z tematem pomocy biblioteki MSDN pod adresem: *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconaspnetauthorization.asp*.

Aby dowiedzieć się więcej na temat dostawców map witryny, sprawdź temat pomocy biblioteki MSDN „ASP.NET Site Navigation Providers".

Jeżeli dołączony dostawca `AspNetXMLSiteMapProvider` nie spełnia Twoich oczekiwań i chciałbyś przechowywać informacje o mapie witryny w innej postaci niż XML (na przykład w bazie danych Oracle), zapoznaj się z tematem pomocy MSDN „Implementing a Site Map Provider".

Wyświetlanie hierarchicznych danych przy użyciu kontrolki TreeView

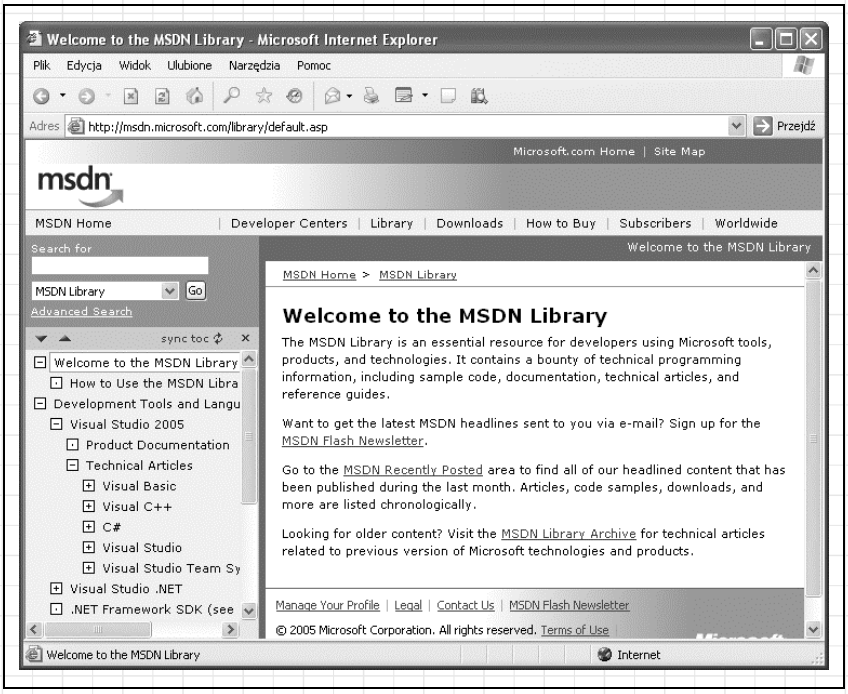
Wreszcie kontrolka TreeView jest dostarczana razem z pakietem Visual Studio.

Jeżeli kiedykolwiek odwiedzałeś witrynę MSDN, bez wątpienia zauważyłeś na stronie głównej po lewej stronie menu nawigacyjne w postaci drzewa (rysunek 2.25).

Informacje te są wyświetlane przy użyciu kontrolki `TreeView`, która oferuje alternatywny w stosunku do kontrolki `SiteMapPath` sposób nawigacji w Twojej witrynie. Mapy w postaci drzewa są idealne do wyświetlania hierarchicznych informacji i, w odróżnieniu od kontrolki `SiteMapPath`, pozwalają na przeglądanie całej witryny, a nie tylko ścieżki prowadzącej do bieżącej strony.

Jak to osiągnąć?

Korzystając z projektu utworzonego w ostatnim ćwiczeniu, dodajmy do witryny kontrolkę `TreeView`. Dokonasz tego przez dodanie kontrolki do strony wzorcowej i dołączenie jej do Twojej mapy witryny, tak aby użytkownicy mogli przeglądać Twoją witrynę, klikając elementy kontrolki `TreeView`.



Rysunek 2.25. Nawigacja w postaci drzewa na witrynie MSDN

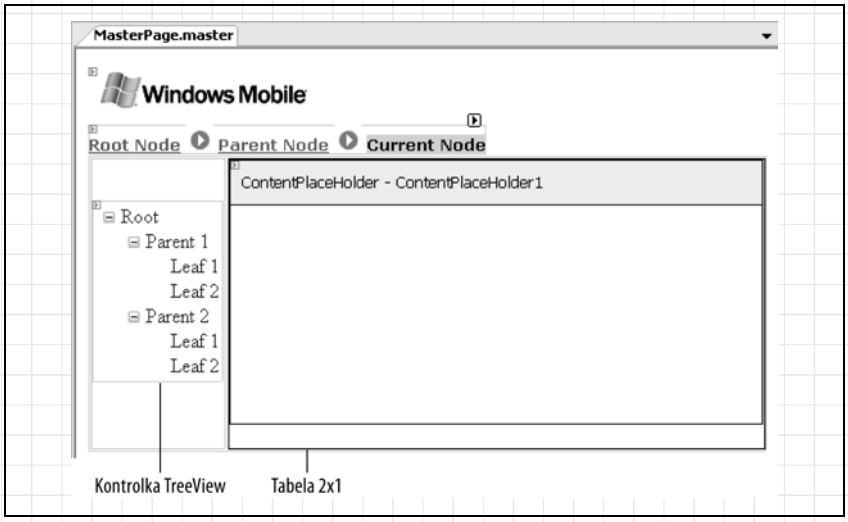
1. Na stronie wzorcowej *MasterPage.master* dodaj tabelę o wymiarach 2×1 (menu *Layout/Insert Table*). Dodaj kontrolkę `TreeView` do lewej komórki tabeli, natomiast do prawej komórki tabeli przeciągnij kontrolkę `ContentPlaceHolder` (rysunek 2.26).

2. Z menu *TreeView Tasks* kontrolki `TreeView` wybierz *<New data source...>* (rysunek 2.27).

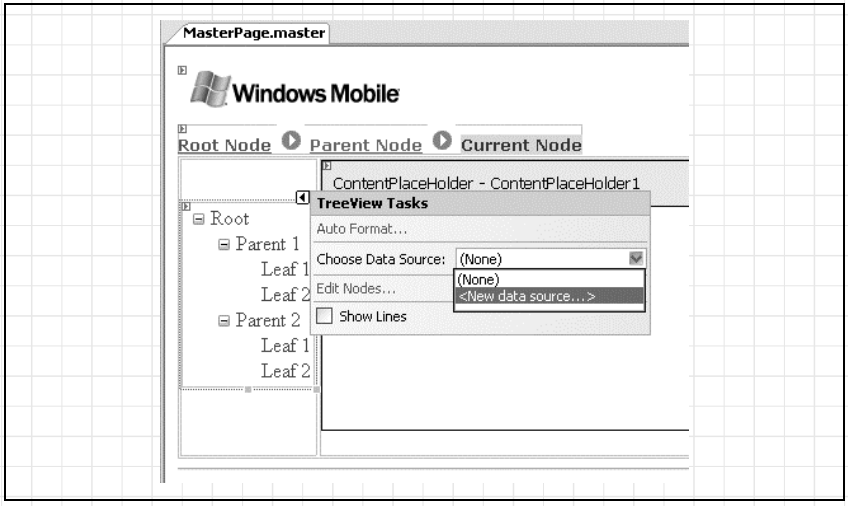
3. W oknie konfiguracyjnym *Data Source Configuration Wizard* zaznacz opcję `Site Map`, a następnie kliknij przycisk *OK* (rysunek 2.28).

4. Zastosuj temat MSDN (używając odnośnika „Auto Format...” w menu *TreeView Tasks* pokazanym na rysunku 2.29) dla kontrolki `TreeView` i ustaw właściwość `BackColor` kontrolki `TreeView` jako `Gray` (`#E0E0E0`).

   Kontrolka `SiteMapDataSource` przedstawia mapę witryny. Jak to zostało wyjaśnione w podrozdziale „Tworzenie mapy witryny dla swojej witryny internetowej”, mapa witryny opisuje logiczną strukturę witryny internetowej i związki między należącymi do niej stronami.



Rysunek 2.26. Zapełnianie strony wzorcowej



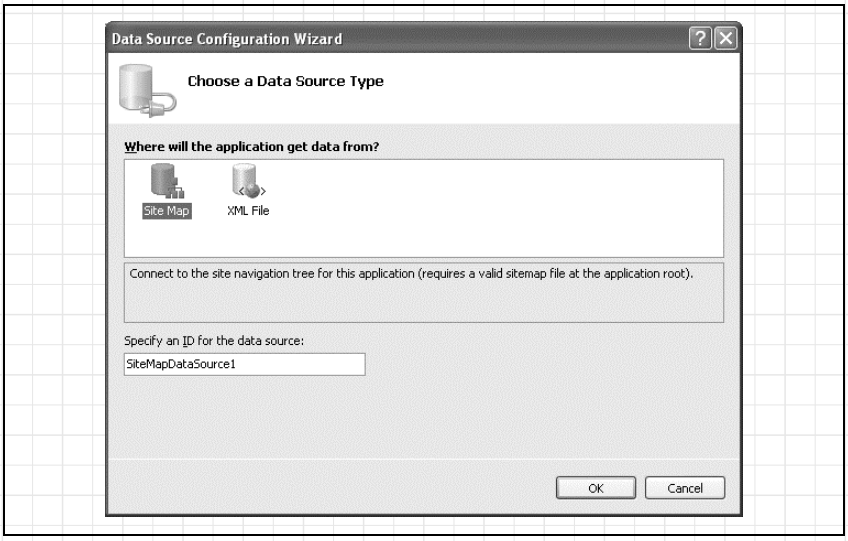
Rysunek 2.27. Dodanie kontrolki TreeView do strony wzorcowej

WSKAZÓWKA

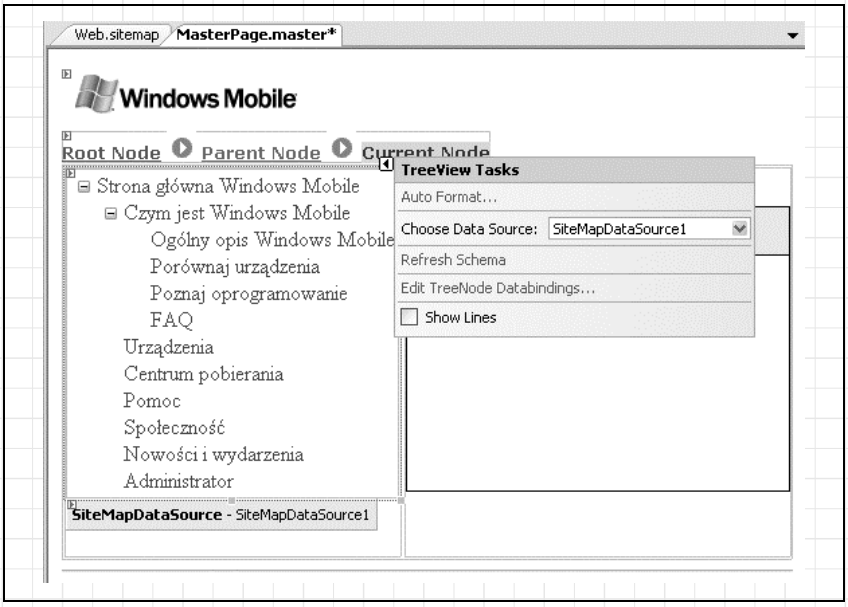
Visual Studio 2005 obsługuje różne style formatowania dla kontrolki `TreeView`. Na rysunku 2.30 znajduje się kilka przykładów.

---

5. Strona wzorcowa powinna teraz wyglądać jak na rysunku 2.31.

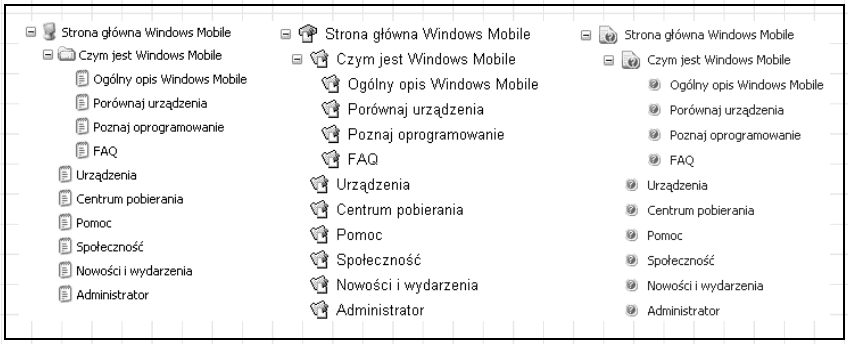


Rysunek 2.28. Wybór opcji Site Map jako źródła danych

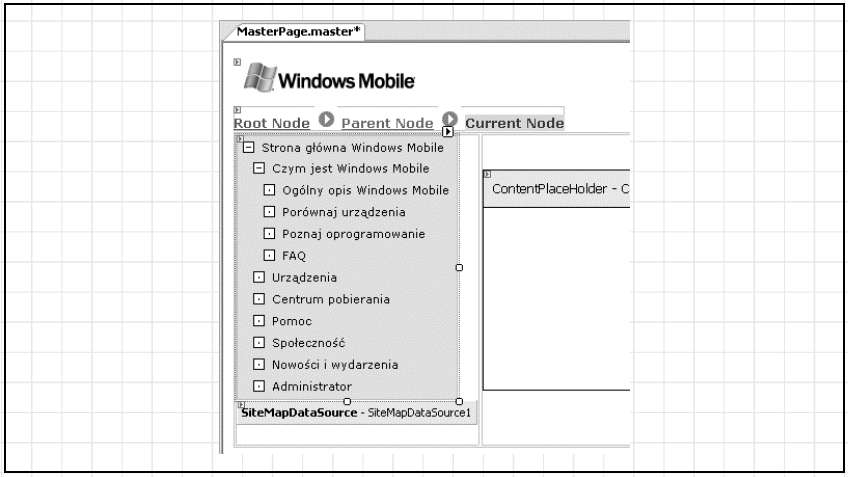


Rysunek 2.29. Kontrolka TreeView

6. Naciśnij klawisz *F5*, aby zobaczyć jak wygląda kontrolka `TreeView`. Powinieneś być teraz w stanie poruszać się pomiędzy wszystkimi stronami swojej witryny przez klikanie poszczególnych elementów na kontrolce `TreeView` (rysunek 2.32).



Rysunek 2.30. Różne style formatowania obsługiwane przez TreeView



Rysunek 2.31. Kontrolka TreeView na stronie wzorcowej

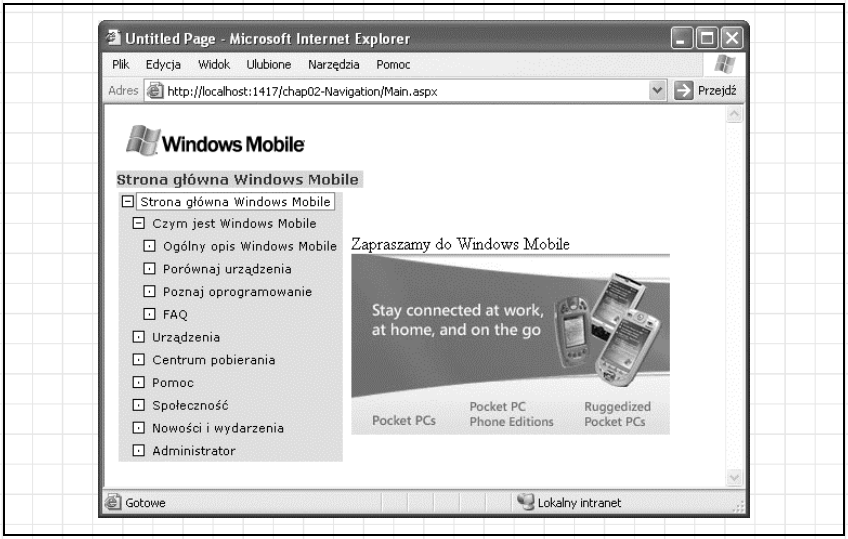
WSKAZÓWKA

Jeżeli na swojej stronie korzystasz z ramek, możesz użyć właściwości `Target` kontrolki `TreeView` do wskazania nazwy ramki, która ma wyświetlić stronę, kiedy dany element zostanie kliknięty.

A co...

...z użyciem swoich własnych dokumentów XML dla kontrolki TreeView zamiast mapy witryny?

Oprócz pliku mapy witryny z kontrolką `TreeView` możesz również używać własnego dokumentu XML, dołączonego do kontrolki `TreeView`.



Rysunek 2.32. Testowanie kontrolki TreeView

Na przykład, być może chcesz utworzyć zarys książki, a następnie połączyć go z kontrolką `TreeView`, aby umożliwić użytkownikom poruszanie się po książce przy użyciu kontrolki `TreeView`.

W celu skonfigurowania kontrolki `TreeView`, aby wykorzystywała plik XML, wykonaj następujące kroki:

1. Dodaj dokument XML do projektu, klikając prawym przyciskiem myszy nazwę projektu w oknie *Solution Explorer*, a następnie wybierając opcję *Add New Item...* i zaznaczając element *XML File*. Utworzonemu plikowi nadaj nazwę `Book.xml`.
2. Zapełnij plik *Book.xml* kodem XML przedstawionym na listingu 2.7.

Listing 2.7. Plik Book.xml

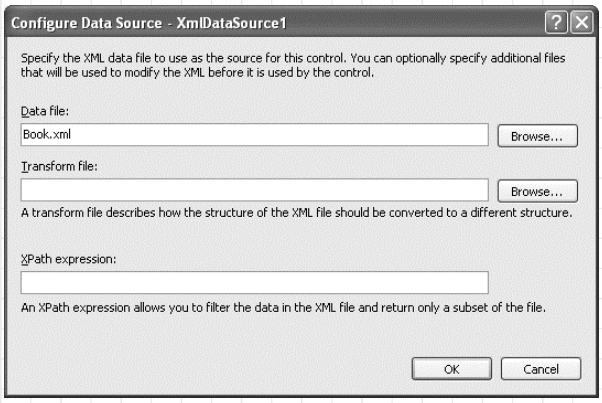
```
<Book>
   <Content Title="Przewodnik po .NET Compact Framework "
         URL="booktop.aspx">
      <Content Title="Część I" URL="PartI.aspx">
         <Content Title="Technologia .NET Framework a urządzenia mobilne"
               URL="PartIMain.aspx">
            <Content Title=".NET CLR" URL="PartI1.aspx" />
            <Content Title="Urządzenia mobilne z Windows "
                  URL="PartI2.aspx" />
            <Content Title="Narzędzia i obsługa języków"
                  URL="PartI3.aspx" />
```

```
            <Content Title="Witaj Pocket PC" URL="PartI4.aspx" />
            <Content Title="Debugowanie aplikacji"
                  URL="PartI5.aspx" />
            <Content Title="Rozwiązywanie problemów" URL="PartI6.aspx" />
         </Content>
      </Content>
      <Content Title="Część II" URL="PartIIMain.aspx">
         <Content Title="Projektowanie interfejsu użytkownika"
         URL="PartII.aspx">
            <Content Title="Używanie kontrolek Windows Forms"
                  URL="PartII1.aspx" />
            <Content Title="Kwestie projektowania aplikacji dla telefonów"
                  URL="PartII2.aspx" />
         </Content>
      </Content>
      <Content Title="Część III" URL="PartIIIMain.aspx">
         <Content Title="Projekty" URL="PartIII.apx">
            <Content Title="Projekt A: Przelicznik walut"
                  URL="PartIIIA.aspx" />
            <Content Title="Projekt B: System zamawiania książek"
                  URL="PartIIIB.apx" />
            <Content Title="Projekt C: Czat Bluetooth"
                  URL="PartIIIC.apx" />
         </Content>
      </Content>
      <Content Title="Część IV" URL="PartIVMain.aspx">
         <Content Title="Wdrażanie aplikacji .NET Compact Framework"
               URL="PartIV.aspx">
            <Content Title="Pakowanie plików CAB" URL="PartIV1.aspx" />
            <Content Title="Rozpowszechnianie .NET Compact Framework"
                  URL="PartIV2.aspx" />
            <Content Title="Wdrażanie aplikacji do telefonów komórkowych"
                  URL="PartIV3.aspx" />
         </Content>
      </Content>
   </Content>
</Book>
```

Zauważ, że możesz określić plik XSLT do przekształcenia dokumentu XML z jednego formatu na inny. Możesz również użyć wyrażenia XPath do filtrowania części dokumentu XML.

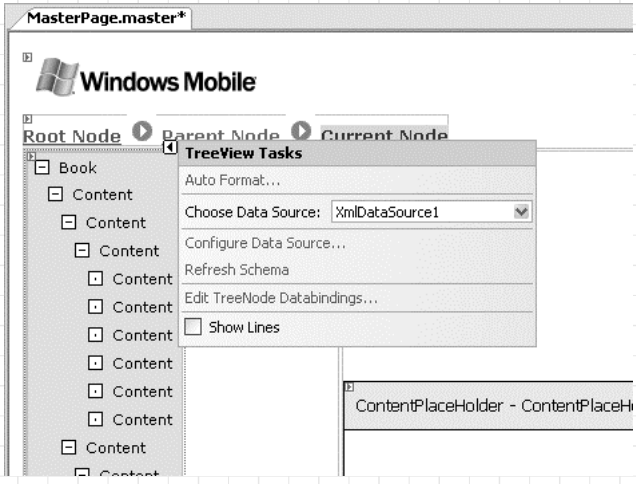
3. Z menu *TreeView Task* kontrolki `TreeView` wybierz opcję *<New data source...>* i w oknie *Data Source Configuration Wizard*, które się pojawi wskaż *XML File* jako źródło danych. Pozostaw domyślną nazwę *XmlDataSource1* i kliknij przycisk *OK*, by przejść do kolejnego okna dialogowego.

4. Zostaniesz poproszony o wskazanie pliku XML, który ma zostać użyty jako źródło dla kontrolki `TreeView` (rysunek 2.33). Wybierz ostatnio dodany plik *Book.xml*.



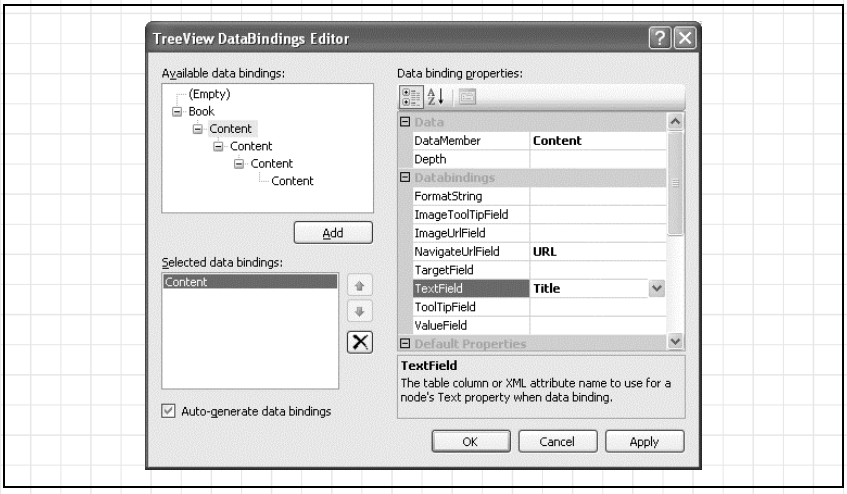
Rysunek 2.33. Określenie źródła XmlDataSource

5. Kontrolka `TreeView` powinna teraz wyglądać podobnie do tej, która została pokazana na rysunku 2.34. Będziesz musiał wyedytować węzły łączenia danych, w celu wyświetlenia przez kontrolkę `TreeView` wybranych przez siebie informacji. Kliknij odnośnik *Edit TreeNode Databindings...* z menu *TreeView Tasks*.



Rysunek 2.34. Kontrolka TreeView

6. W polu *Available DataBindings* (rysunek 2.35) zaznacz element `Content` (został podświetlony na rysunku) i kliknij przycisk *Add*. Ustaw właściwości `DataBinding` jak poniżej:



Rysunek 2.35. Wykorzystanie edytora DataBindings kontrolki TreeView

`DataMember`

Element w dokumencie XML używany do połączenia z węzłem w kontrolce `TreeView`. Ustaw tę właściwość jako `Content`.

`NavigateUrlField`

Adres URL, do którego nastąpi przejście, kiedy zostanie kliknięty węzeł w kontrolce `TreeView`. Ustaw tę właściwość jako `URL`.

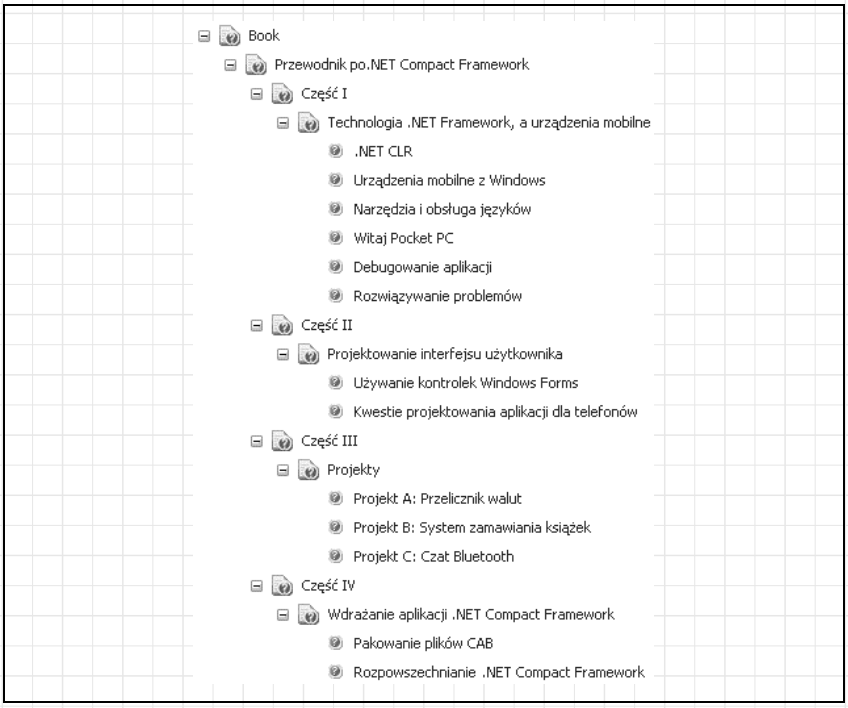
`TextField`

Atrybut używany do wyświetlenia w węźle kontrolki `TreeView`. Ustaw tę właściwość jako `Title`.

7. Kliknij przycisk *OK*.
8. Zastosuj temat FAQ dla kontrolki `TreeView` (używając odnośnika „Auto Format…" w menu *TreeView Tasks*).
9. Naciśnij klawisz *F5*, aby przetestować aplikację. Kontrolka `TreeView` będzie wyglądała jak ta pokazana na rysunku 2.36.

Więcej informacji

O alternatywnych kontrolkach `TreeView` w ASP.NET możesz przeczytać w następujących artykułach:



Rysunek 2.36. Kontrolka TreeView przyłączona do pliku XML

- *http://www.obout.com/obout/treeview/treeview.asp*
- *http://www.asp.net/ControlGallery/default.aspx?Category=38&tabindex=2*
- *http://www.telerik.com/Default.aspx?PageID=1828*

Zamiast statycznie łączyć kontrolkę TreeView z mapą witryny lub plikiem XML, skorzystaj z możliwości tworzenia ich dynamicznie w trakcie działania.

Programowe zapełnianie kontrolki TreeView

W innych ćwiczeniach tego rozdziału dowiedziałeś się, w jaki sposób kontrolka `TreeView` może zostać dołączona do pliku mapy witryny, jak również do dokumentów XML w trakcie projektowania. Dysponujesz także możliwością dynamicznego budowania swojej kontrolki `TreeView`, po jednym węźle. Na przykład, gdy chcesz wyświetlić określoną strukturę dysku, używając kontrolki `TreeView` lub posiadasz witrynę, która się

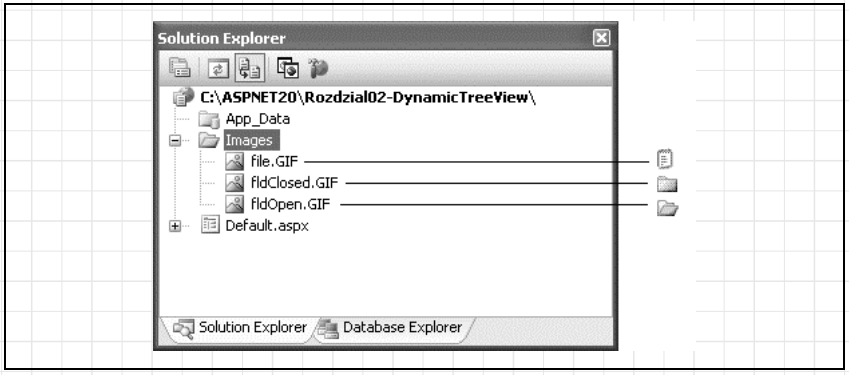
często zmienia, wówczas dynamiczne budowanie drzewa staje się bardziej efektywnym rozwiązaniem.

W tym ćwiczeniu użyjesz kontrolki `TreeView` do wyświetlenia bieżącej ścieżki aplikacji i jej podkatalogów.

Jak to osiągnąć?

W tym ćwiczeniu nauczysz się, jak dostosować do własnych potrzeb kontrolkę `TreeView` i dynamicznie zapełniać ją elementami. W szczególności, wyświetlisz bieżącą ścieżkę aplikacji i jej wszystkich podkatalogów w kontrolce `TreeView`.

1. Uruchom Visual Studio 2005 i utwórz nową aplikację sieciową ASP.NET. Zapisz projekt na dysku jako *C:\ASPNET20\Rozdział02-DynamicTreeView*.
2. Przeciągnij i upuść kontrolkę `TreeView` na domyślny formularz Web Form.
3. W oknie *Solution Explorer* kliknij prawym przyciskiem myszy nazwę projektu i wybierz z menu opcję *Add New Folder/Regular Folder*. Nazwij ten katalog `Images`, będziesz go wykorzystywał do przechowywania swoich plików graficznych.
4. Do katalogu *C:\ASPNET20\Rozdział02-DynamicTreeView\Images* dodaj obrazki pokazane na rysunku 2.37.

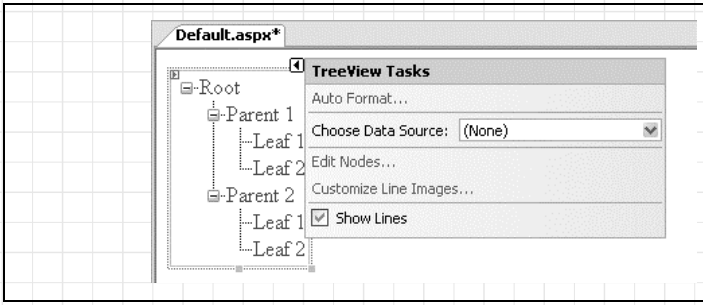


Rysunek 2.37. Obrazki w katalogu Images

WSKAZÓWKA

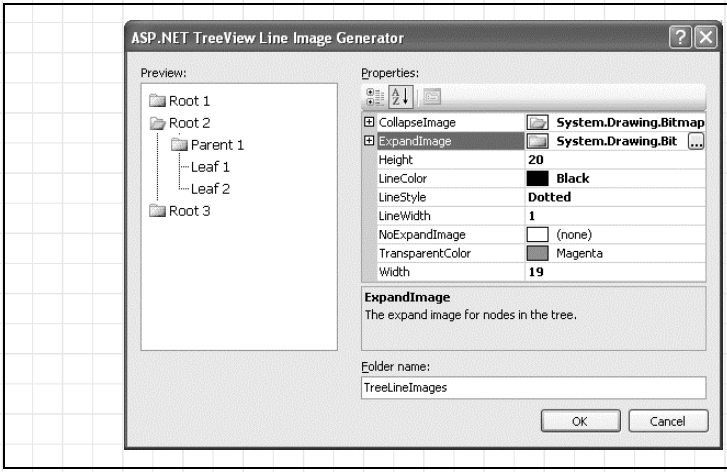
Wspomniane obrazki można pobrać pod adresem: *ftp://ftp.helion.pl/przyklady/aspnzp.zip*.

5. W oknie *TreeView Tasks* zaznacz pole wyboru *Show Lines*, a następnie kliknij odnośnik *Customize Line Images...* (rysunek 2.38).



Rysunek 2.38. Dostosowanie do własnych potrzeb kontrolki TreeView

6. W oknie dialogowym *ASP.NET TreeView Line Image Generator* ustaw następujące właściwości (rysunek 2.39):



Rysunek 2.39. Ustawianie właściwości kontrolki TreeView

- `CollapseImage`: *fldOpen.gif*
- `ExpandImage`: *fldClosed.gif*

7. Kliknij przycisk *OK*. Kiedy zostaniesz zapytany o utworzenie nowego katalogu o nazwie *TreeLineImages*, kliknij przycisk *Yes*. Katalog *TreeLineImages* zawiera serie obrazków użytych do wyświetlenia kontrolki `TreeView`. Gdy węzeł `TreeView` jest rozwinięty, będzie wyświetlony obrazek *fldOpen.gif*, natomiast kiedy węzeł jest zwinięty, będzie wyświetlony obrazek *fldClosed.gif*.

8. Kliknij dwukrotnie domyślny formularz Web Form, aby przejść do ukrytego kodu.

9. Dodaj kod metody `getSubDirectories()` przedstawionej na listingu 2.8. Metoda ta rekurencyjnie pobiera wszystkie podkatalogi w bieżącym katalogu i dodaje je jako węzły do kontrolki `TreeView`[1].

Listing 2.8. Metoda getSubDirectories()

```
Public Sub getSubDirectories(ByVal path As String, _
      ByVal node As TreeNode)
   '---Pobiera podkatalogi w bieżącym katalogu.
   Dim dirs As String() = Directory.GetDirectories(path)

   '---Brak podkatalogów.
   If dirs.Length = 0 Then
      Exit Sub
   Else
      '---W przypadku każdego podkatalogu, dodaj go do kontrolki TreeView
      ' i powtórz rekurencyjnie procedurę.
      Dim dir As String
      For Each dir In dirs
         '---Dodaj względną ścieżkę do kontrolki TreeView.
         Dim newNode As New TreeNode( _
               dir.Substring(dir.LastIndexOf("\") + 1))
         newNode.ToolTip = dir
         node.ChildNodes.Add(newNode)
         '---Odszukaj jego podkatalogi.
         getSubDirectories(dir, newNode)
         '---Odszukaj wszystkie pliki.
         getFiles(dir, newNode)
         '---Zamknij bieżący węzeł.
         newNode.CollapseAll()
      Next
   End If
End Sub
```

[1] Aby widoczny był obiekt `Directory`, należy dodać linię `Imports System.IO` na początku kodu. Importuje ona bibliotekę systemową dostarczającą obiekty zarządzające urządzeniami wejścia i wyjścia — *przyp. red.*

10. Dodaj kod metody `getFiles()` przedstawionej na listingu 2.9. Metoda `getFiles()`wyświetla listę wszystkich plików w bieżącym katalogu, a następnie dodaje pliki jako węzły w kontrolce `TreeView`.

Listing 2.9. Metoda getFiles()

```
Public Sub getFiles(ByVal path As String, ByVal node As TreeNode)
   Dim files As String() = Directory.GetFiles(path)
   Dim file As String

   '---Brak plików i podkatalogów w bieżącym katalogu.
   If files.Length = 0 And node.ChildNodes.Count = 0 Then
      Dim newNode As New TreeNode("Katalog jest pusty.")
      node.ChildNodes.Add(newNode)
      Exit Sub
   End If
   For Each file In files
      '---Dodaj plik do kontrolki TreeView.
      Dim newNode As New TreeNode(file.Substring(path.Length + 1))
      newNode.ToolTip = file
      newNode.ImageUrl = "Images\file.gif"
      node.ChildNodes.Add(newNode)
   Next
End Sub
```

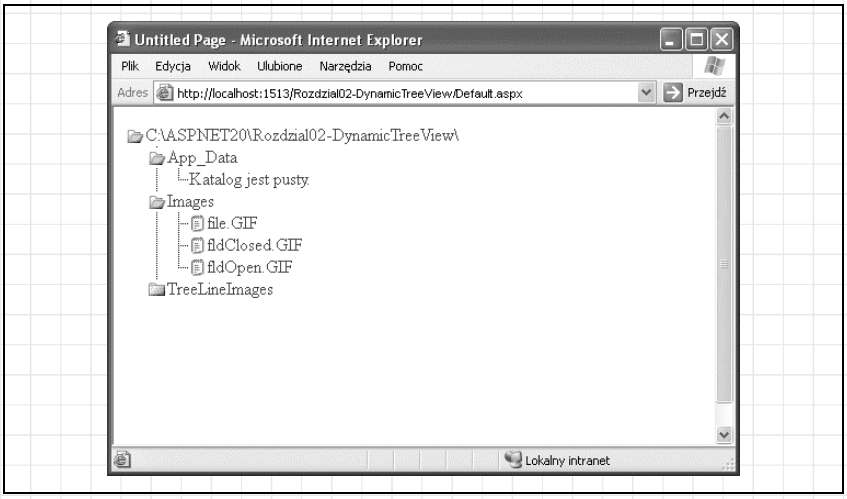
WSKAZÓWKA

Możesz określić obrazek użyty dla węzła we właściwości `ImageUrl`.

11. Na koniec dodaj następujący kod do zdarzenia `Page_Load`, co spowoduje, że kontrolka `TreeView` będzie zapełniana w trakcie ładowania się strony:

```
Protected Sub Page_Load(ByVal sender AS Object, _
      ByVal e As System.EventArgs) _
      Handles Me.Load
   If Not IsPostBack Then
      TreeView1.Nodes.Add(New _
         TreeNode(request.PhysicalApplicationPath))
      getSubDirectories(Request.PhysicalApplicationPath, _
         TreeView1.Nodes(0))
   End If
End Sub
```

12. W celu przetestowania aplikacji naciśnij klawisz *F5*. Powinieneś zobaczyć stronę podobną do tej z rysunku 2.40.



Rysunek 2.40. Wyświetlenie dynamicznie wygenerowanej kontrolki TreeView

A co...

...z programowym manipulowaniem kontrolką TreeView?

Aby programowo rozwinąć lub zwinąć wszystkie węzły, musisz użyć metod `ExpandAll()` i `CollapseAll()`, na przykład w taki sposób:

```
TreeView1.CollapseAll()
TreeView1.ExpandAll()
```

...z dostosowaniem zachowania pojedynczych elementów w kontrolce TreeView?

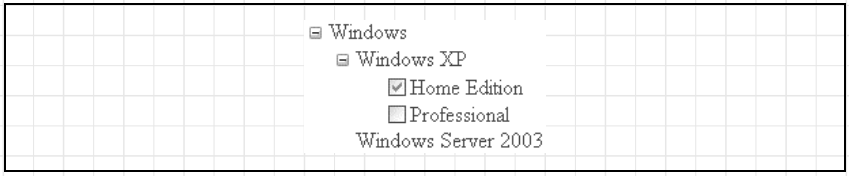
Jest kilka typów zachowań, które możesz ustawić dla każdego elementu w kontrolce `TreeView`. Niektóre z nich zostały przedstawione w widoku *Source View* kontrolki `TreeView` na listingu 2.10.

Listing 2.10. Ustawienie właściwości kontrolki TreeView

```
<asp:TreeView ID="TreeView2" runat="server">
   <Nodes>
      <asp:TreeNode SelectAction="Expand"
           Value="Windows" Text="Windows">
         <asp:TreeNode SelectAction="SelectExpand"
              Value="Windows XP" Text="Windows XP">
            <asp:TreeNode ShowCheckBox="True"
                 Value="Home Edition" Checked="True"
                 NavigateUrl="xphome.aspx"
                 Text="Home Edition">
```

```
            </asp:TreeNode>
            <asp:TreeNode ShowCheckBox="True"
                  Value="Professional"
                  NavigateUrl="xpProfessional.aspx"
                  Text="Professional">
            </asp:TreeNode>
          </asp:TreeNode>
          <asp:TreeNode Value="Windows 2003 Server"
                NavigateUrl="win2003.aspx"
                Text="Windows Server 2003">
          </asp:TreeNode>
        </asp:TreeNode>
    </Nodes>
</asp:TreeView>
```

Kiedy kontrolka `TreeView` zostanie wyświetlona w przeglądarce Internet Explorer, będzie wyglądała jak na rysunku 2.41.



Rysunek 2.41. Kontrolka TreeView po zabiegu dostosowywania

Jeżeli jesteś w stanie porównać zachowanie kontrolki `TreeView` z atrybutami przypisanymi do jej elementów na listingu 2.10, powinieneś zauważyć następujące zależności:

- Kliknięcie elementu *Windows* albo rozwinie, albo zwinie drzewo (`SelectAction="Expand"`). Powtórne odświeżenie nie występuje.
- Kliknięcie elementu *Windows XP* spowoduje rozwinięcie i pokazanie znajdujących się pod spodem elementów (jeśli jest zwinięty). Występuje ponowne odświeżanie (`SelectAction="SelectExpand"`). Możesz pobrać informacje o zaznaczonym elemencie w zdarzeniu `Page_Load` dzięki właściwości `SelectedNode` w kontrolce `TreeView`.
- Element *Home Edition* posiada wyświetlone obok pole wyboru (`ShowCheckBox="True"`), które jest zaznaczone (`Checked="True"`). Kliknięcie elementu spowoduje załadowanie strony *xphome.aspx* (`NavigateUrl="xphome.aspx"`).

- Element *Professional* również posiada wyświetlone obok pole wyboru (`ShowCheckBox="True"`). Kliknięcie elementu spowoduje załadowanie strony *xpProfessional.aspx* (`NavigateUrl="xpProfessional.aspx"`).
- Kliknięcie elementu *Windows Server 2003* spowoduje załadowanie strony *win2003.aspx*.

WSKAZÓWKA

Domyślną wartością `SelectAction` dla wszystkich elementów `TreeView` jest `Select`. W celu wyświetlenia strony docelowej w nowym oknie, ustaw właściwość `Target`.

Więcej informacji

Sprawdź tematy pomocy biblioteki MSDN dla kontrolki `TreeView`, aby uzyskać szczegółowe informacje na temat wszystkich właściwości i metod ujawnionych przez tę wszechstronną kontrolkę.

Wyświetlanie rozwijanych menu za pomocą kontrolki Menu

Obecnie w ASP.NET 2.0 możesz użyć kontrolki Menu do zbudowania menu po stronie klienta. Nigdy więcej skryptów po stronie klienta!

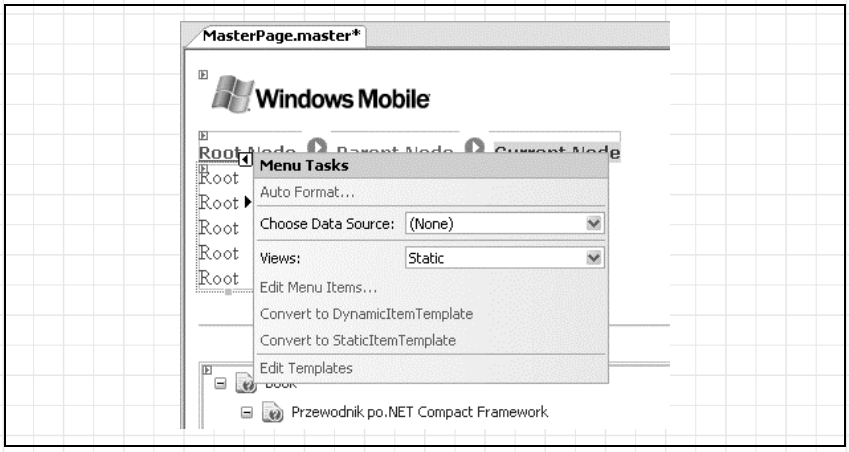
Oprócz kontrolki `TreeView` do wyświetlania informacji o witrynie, możesz użyć kontrolki `Menu` do automatycznego generowania w przeglądarce rozwijanych menu, które wymagają skryptów po stronie klienta.

Jak to osiągnąć?

W tym ćwiczeniu dodasz kontrolkę `Menu` do strony wzorcowej utworzonej we wcześniejszym ćwiczeniu „Wyświetlanie hierarchicznych danych przy użyciu kontrolki TreeView". Użytkownik będzie mógł używać kontrolki menu do poruszania się po witrynie.

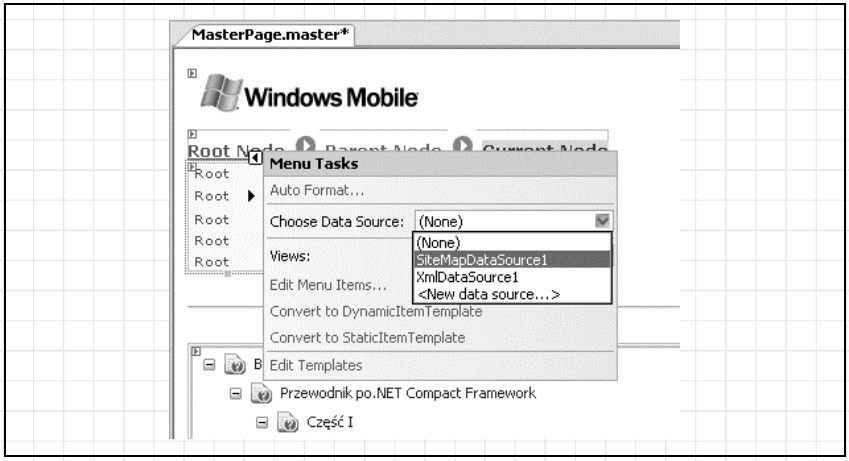
1. Korzystając z projektu utworzonego w ćwiczeniu „Wyświetlanie hierarchicznych danych przy użyciu kontrolki TreeView", dodaj kontrolkę `Menu` (znajdującą się w zakładce *Navigation* paska narzędziowego) do strony wzorcowej. Następnie zastosuj temat *Colorful* (poprzez

odnośnik *Auto Format...* w menu *Menu Tasks*), aby zmienić jego wygląd (rysunek 2.42).



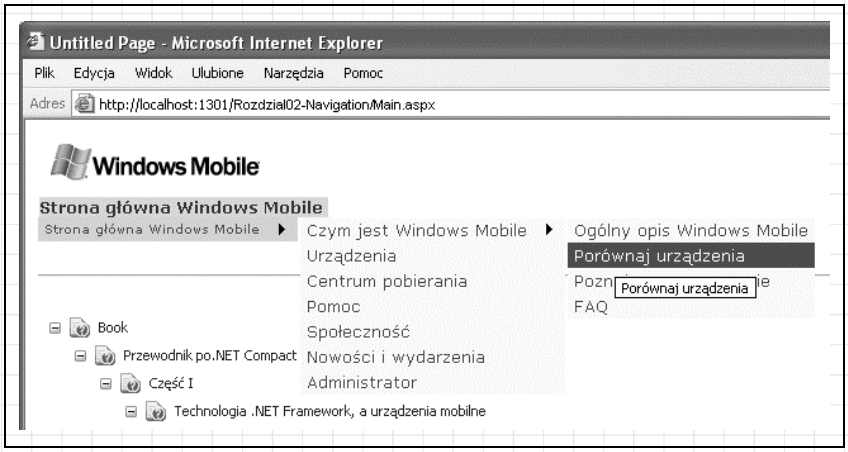
Rysunek 2.42. Dostosowanie do własnych potrzeb kontrolki Menu

2. W *Menu Tasks* kontrolki `Menu` wybierz `SiteMapDataSource1` jako jego dane źródłowe (rysunek 2.43). Kontrolka `SiteMapDataSource1` została dodana podczas wcześniejszej konfiguracji kontrolki `TreeView`.



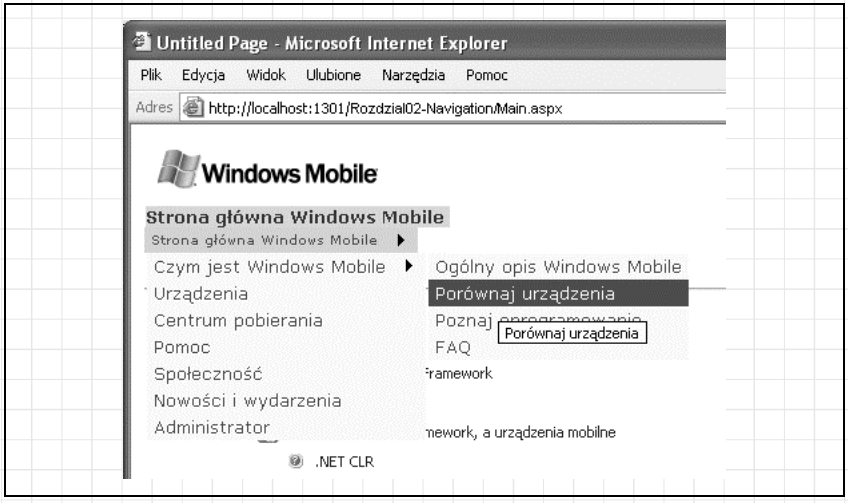
Rysunek 2.43. Dodanie kontrolki Menu do strony wzorcowej

3. Aby przetestować kontrolkę `Menu`, naciśnij klawisz *F5*. Powinieneś być w stanie poruszać się po witrynie za pomocą kontrolki `Menu` (rysunek 2.44).



Rysunek 2.44. Testowanie kontrolki Menu

4. Zauważ, że kontrolka `Menu` obsługuje dwa położenia: `Horizontal` i `Vertical`. Zmień właściwość `Orientation` kontrolki `Menu` z `Vertical` na `Horizontal` i zauważ zmianę (rysunek 2.45).



Rysunek 2.45. Zmiana położenia kontrolki Menu

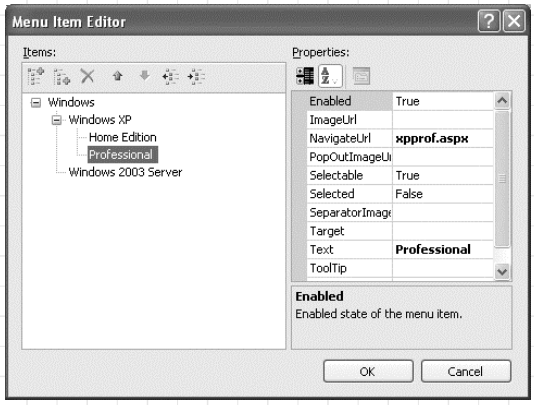
WSKAZÓWKA

Domyślnym położeniem kontrolki `Menu` jest `Vertical`.

A co...

...z deklaratywnym tworzeniem elementów w kontrolce Menu?

Łączenie kontrolki `Menu` z plikiem XML lub mapą witryny nie jest jedynym sposobem dodawania elementów do kontrolki — możesz również ręcznie utworzyć elementy w kontrolce `Menu`. W tym celu możesz albo wykorzystać edytor *Menu Item Editor* (dostępny w menu *Menu Tasks* po zaznaczeniu *Edit Menu Items...*) pokazany na rysunku 2.46, albo dodać elementy deklaratywnie.



Rysunek 2.46. Użycie edytora Menu Item Editor w celu ręcznego dodania elementów do kontrolki Menu

Aby deklaratywnie dodać elementy menu, przeciągnij i upuść kontrolkę menu na formularzu i przełącz się do widoku *Source View*. Listing 2.11 przedstawia, w jaki sposób zaimplementować strukturę menu pokazaną na rysunku 2.46.

Listing 2.11. Deklaratywne dodanie pozycji menu

```
<asp:Menu ID="Menu1" Target="newWindow" runat="server">
   <Items>
      <asp:MenuItem Value="Windows" Text="Windows">
         <asp:MenuItem Value="Windows XP" Text="Windows XP">
            <asp:MenuItem Value="Home Edition" Text="Home Edition"
                  navigateurl="xphome.aspx">
            </asp:MenuItem>
            <asp:MenuItem Value="Professional" Text="Professional"
                  navigateurl="xpprof.aspx">
            </asp:MenuItem>
```

```
        </asp:MenuItem>
        <asp:MenuItem Value="Windows 2003 Server"
              Text="Windows 2003 Server"
              navigateurl="win2003.aspx">
        </asp:MenuItem>
      </asp:MenuItem>
    </Items>
</asp:Menu>
```

WSKAZÓWKA

W celu otworzenia nowego okna w momencie wybrania elementu menu, musisz ustawić dla atrybutu `Target` pewną wartość. Zwróć uwagę, że atrybut `Target` jest obsługiwany zarówno przez element `<Menu>`, jak i `<MenuItem>`. Jeśli określisz cel na poziomie menu, wszystkie elementy menu będą domyślnie wyświetlane w podanym celu, dopóki nie unieważnisz celu indywidualnie dla pozycji menu. Jeśli atrybut `Target` nie jest ustawiony (lub ma ustawiony pusty łańcuch), to docelową dla menu będzie bieżąca strona.

Zauważ, że otwarcie nowego okna często bywa mylące dla osoby przeglądającej stronę. Dzieje się tak, dopóki element menu jasno nie wskazuje zewnętrznej strony lub witryny.

Więcej informacji

Więcej informacji na temat tworzenia menu przy użyciu skryptów po stronie klienta jest dostępnych pod podanymi poniżej adresami:

- *http://dhtml-menu.com/menu-demos/demo117.html*
- *http://wsabstract.com/javatutors/crossmenu.shtml*
- *http://archive.devx.com/dhtml/articles/td041801/td041801-1.asp*