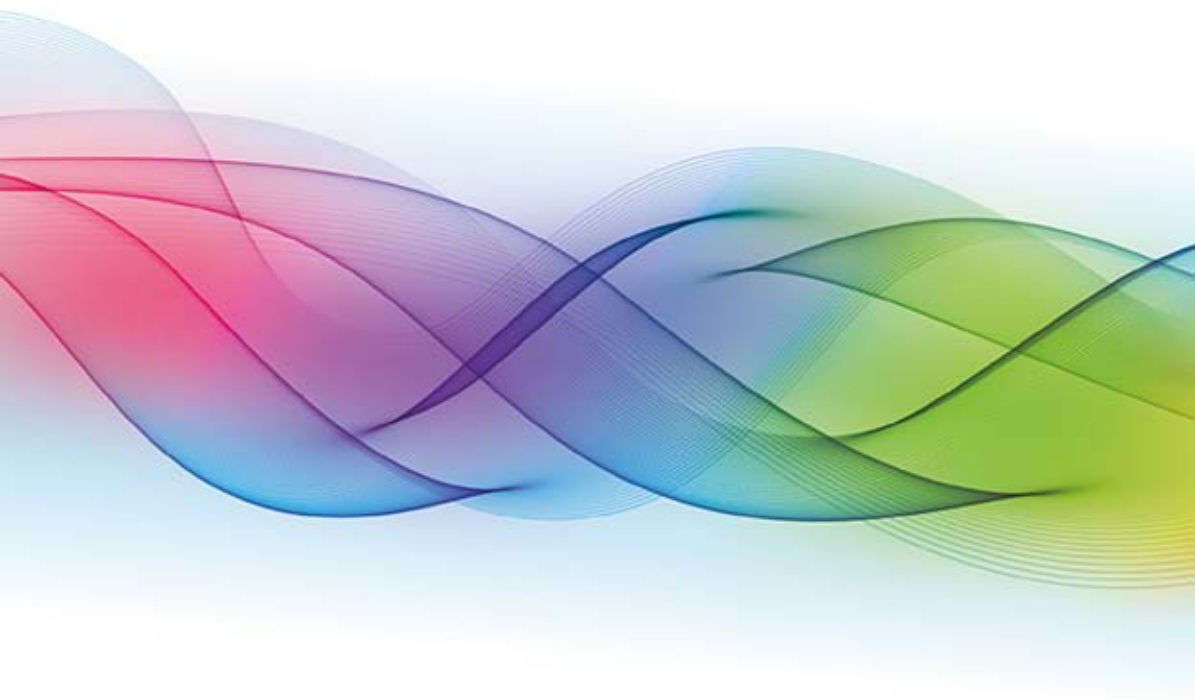
Jacek Matulewski, Maciej Grabek, Maciej Pakulski, Dawid Borycki

ASP.NET Web Forms

Kompletny przewodnik dla programistów interaktywnych aplikacji internetowych w Visual Studio



ASP.NET Web Forms — idealny wybór dla małych aplikacji!

Język C# w projektach ASP.NET Web Forms
Bazy danych aplikacji i korzystanie z LINQ to SQL
Poprawa funkcjonalności stron WWW z wykorzystaniem kontrolki Web Forms i CSS

Redaktor prowadzący: Ewelina Burska
Projekt okładki: Studio Gravite/Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki

Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: http://helion.pl (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/aspnwe
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-246-8284-3

Printed in Poland.

Spis treści

Przedmowa **9**

Wstęp **11**

Rozdział 1. Prosta aplikacja ASP.NET Web Forms, czyli o wszystkim po trochu **15**
Technologia ASP.NET 15
Trivia projektowania aplikacji ASP.NET 16
Tworzenie pustego projektu w Visual Studio 2010 i 2012 16
Tworzenie pustego projektu w Visual Studio 2013 17
Dodawanie strony .aspx 18
Projektowanie interfejsu strony 19
Dodawanie wpisu na stronie 21
Czas życia aplikacji 23
Dane aplikacji 24
Zdarzenia aplikacji — plik Global.asax 25
Przechowywanie stanu aplikacji na dysku serwera 26
Przechowywanie danych po stronie klienta (ciasteczka) 30
Walidacja po stronie klienta 31
Wymagane pole formularza 31
Błąd w Visual Studio 2012 i 2013 32
Ograniczanie zawartości wpisów 34
Podsumowanie walidacji 34
Zadania 35

Rozdział 2. Język C# 5.0 ..... **37**
Platforma .NET 38
Środowisko uruchomieniowe 38
Kod pośredni i podwójna kompilacja 38
Skróty, które warto poznać 39
Podstawowe typy danych 40
Deklaracja i zmiana wartości zmiennej 40
Typy liczbowe oraz znakowy 41
Określanie typu zmiennej przy inicjacji (pseudotyp var) 43
Operatory 43
Konwersje typów podstawowych 45
Operatory is i as 46
Łańcuchy 47
Typ wyliczeniowy 50
Leniwe inicjowanie zmiennych 51

Metody ... 52
Przeciążanie metod ... 53
Domyślne wartości argumentów metod — argumenty opcjonalne (nowość języka C# 4.0) ... 54
Argumenty nazwane (nowość języka C# 4.0) ... 55
Wartości zwracane przez metody ... 55
Zwracanie wartości przez argument metody ... 55
Delegacje i zdarzenia ... 57
Wyrażenia lambda ... 60
Typy wartościowe i referencyjne ... 62
Nullable ... 63
Pudełkowanie ... 64
Typy dynamiczne (nowość języka C# 4.0) ... 65
Sterowanie przepływem ... 68
Instrukcja warunkowa if..else ... 68
Instrukcja wyboru switch ... 68
Pętle ... 69
Wyjątki ... 71
Dyrektywy preprocesora ... 73
Kompilacja warunkowa — ostrzeżenia ... 74
Definiowanie stałych preprocesora ... 74
Bloki ... 75
Atrybuty ... 76
Kolekcje ... 76
„Zwykłe" tablice ... 77
Pętla foreach ... 79
Sortowanie ... 80
Kolekcja List ... 81
Kolekcja SortedList i inne słowniki ... 83
Kolejka i stos ... 84
Tablice jako argumenty metod oraz metody z nieokreśloną liczbą argumentów ... 85
Słowo kluczowe yield ... 86
Nowa forma inicjacji obiektów i tablic ... 88
LINQ ... 88
Operatory LINQ ... 89
Pobieranie danych (filtrowanie i sortowanie) ... 91
Najprostsza prezentacja pobranych danych ... 91
Analiza pobranych danych ... 92
Wybór elementu ... 92
Weryfikowanie danych ... 92
Prezentacja w grupach ... 93
Łączenie zbiorów danych ... 93
Łączenie danych z różnych źródeł w zapytaniu LINQ — operator join ... 94
Możliwość modyfikacji danych źródła ... 95
Elementy programowania współbieżnego (nowość języka C# 4.0) ... 96
Równoległa pętla for ... 96
Przerywanie pętli ... 98
Programowanie asynchroniczne. Modyfikator async i operator await (nowość języka C# 5.0) ... 99
Caller Information (nowość języka C# 5.0) ... 103

Rozdział 3. Programowanie obiektowe w C# 105
Przykład struktury (Ulamek) 106
Przygotowywanie projektu 106
Konstruktor i statyczne obiekty składowe 108
Pierwsze testy 109
Konwersje na łańcuch (metoda ToString) i na typ double 109
Metoda upraszczająca ułamek 110
Właściwości 111
Domyślnie implementowane właściwości (ang. auto-implemented properties) 112
Operatory arytmetyczne 113
Operatory porównania oraz metody Equals i GetHashCode 114
Operatory konwersji 116
Implementacja interfejsu (na przykładzie IComparable) 117
Definiowanie typów parametrycznych 119
Definiowanie typów ogólnych 119
Określanie warunków, jakie mają spełniać parametry 121
Implementacja interfejsów przez typ ogólny 122
Definiowanie aliasów 124
Typy ogólne z wieloma parametrami 124
Rozszerzenia 125
Typy anonimowe 127

Rozdział 4. Wielkie porządki, czyli separacja modelu. Pliki XML 129
Separacja modelu 129
Definicja klasy Wpisy 130
Osadzanie modelu w aplikacji 131
Zapis kolekcji w plikach XML 133
Podstawy języka XML 133
Użycie LINQ to XML 134
Odczyt danych z pliku XML 137
Zapis do pliku XML 138
Rejestrowanie zdarzeń 139

Rozdział 5. Udostępnianie danych przez usługę WCF 141
Tworzenie biblioteki usług WCF 141
Definiowanie kontraktów 143
Definiowanie metod udostępnianych przez usługę 144
Dodawanie odwołania do usługi w aplikacji ASP.NET 146
Użycie usługi WCF w aplikacji ASP.NET 147

Rozdział 6. Baza danych SQL Server w ASP.NET Web Forms. LINQ to SQL 151
Tworzenie bazy danych SQL Server w Visual Studio 2010 152
Tworzenie bazy danych SQL Server w Visual Studio 2012 i 2013 154
Klasa encji 156
Prezentacja danych w kontrolkach 160
Zadania 162

Rozdział 7. Strony wielojęzyczne (lokalizacja) 163

Rozdział 8. Podstawowe wiadomości o kaskadowych arkuszach stylów 169
Formatowanie na podstawie typu znacznika HTML 170
Formatowanie na podstawie identyfikatora 173
Formatowanie na podstawie nazwy klasy 174

Rozdział 9. Kontrolki uwierzytelniania użytkowników w Web Forms 177
Tworzenie wzorca 177
Strona logowania 179
Konfiguracja uwierzytelniania i baza danych 181
Wyświetlanie nazwy użytkownika 183
Strona rejestrowania nowego użytkownika 185
Strona przypominania hasła 187
Strona zmiany hasła i problem dostępu do stron 188
Odczytywanie informacji o koncie z poziomu kodu 191
Zadania 194

Rozdział 10. Przegląd kontrolek biblioteki Web Forms 195
Baza danych 195
Strona z formularzem dodawania zdjęć 197
Strona przeglądania zbiorów zdjęć 204
Menu aplikacji 209
Kontrolki GridView i DetailsView 209
Zadania 213

Rozdział 11. Studium przypadku: gra Reversi 215
Silnik 216
Konstruktor klasy 220
Implementacja zasad gry 221
Pierwsze testy 223
Metody dodatkowe 225
Kontrolka prezentująca planszę 226
Prezentacja stanu gry w kontrolce 230
Wyświetlanie dodatkowych informacji o stanie gry 231
Interakcja z użytkownikiem 234
Wykrywanie szczególnych sytuacji w grze 238
Dziedziczenie 243
Jak znaleźć najlepszy ruch? 245
Podpowiedź komputera 248
Gra z komputerem 250
Zadania 252

Rozdział 12. Technologia AJAX 253
Kontrolka UpdatePanel 254
Dwie kontrolki UpdatePanel 255
Kontrolka Timer 257
Kontrolka UpdateProgress 259
Metody strony (page methods) 262
Metody strony — wykorzystanie jQuery 263

Rozdział 13. AJAX Control Toolkit 265
Co to jest AJAX Control Toolkit? 265
Używanie kontrolek ACT we własnych projektach 268
Instalacja kontrolek ACT w środowisku Visual Studio 268
Użycie rozszerzenia ConfirmButtonExtender 269
Jak to jest zrobione? 271
Suwaki 272
Reklama 274

Rozdział 14. Typowe elementy aplikacji internetowych ASP.NET 277
Pliki konfiguracyjne 277
SiteMap 280
Style, skórki i tematy 283
Temat 283
Skórki 285
Query String 286
Buforowanie (cache) 288

Skorowidz 292

Rozdział 1.

Prosta aplikacja ASP.NET Web Forms, czyli o wszystkim po trochu

Jacek Matulewski

Technologia ASP.NET

Czym jest ASP.NET? W wielkim skrócie jest to technologia uruchamiana na serwerze WWW, który korzystając z platformy .NET, dynamicznie generuje na podstawie przygotowanych przez programistów szablonów (plików *.aspx*) strony HTML ze skryptami JavaScript — są one następnie przekazywane do programu IIS (ang. *Internet Information Services*) udostępniającego ich zawartość w internecie. ASP.NET jest więc technologią pracującą po stronie serwera. Komputer-klient nie musi mieć nawet zainstalowanej platformy .NET. Wystarczy, by dysponował przeglądarką obsługującą skrypty JavaScript. To stawia ASP.NET w jednym szeregu z JSP (ang. *Java Server Pages*) czy aplikacjami internetowymi budowanymi na bazie PHP.

Nazwa ASP.NET wywodzi się od poprzednika tej technologii, czyli ASP (ang. *Active Server Pages*), w której szablonom HTML towarzyszył najczęściej kod VBScript. Aplikacje ASP w odróżnieniu od ASP.NET, a tak samo jak w CGI (ang. *Common Gateway Interface*), nie posiadały jednak stanu, co bardzo utrudniało ich tworzenie.

Trivia projektowania aplikacji ASP.NET

Zbudujmy aplikację, która będzie implementacją prostej księgi gości. Idea aplikacji jest następująca: dowolny internauta po wejściu na stronę bez konieczności logowania będzie mógł umieścić na niej swój wpis. Aby od podstaw przedstawić elementy ASP.NET, zaczniemy od pustego projektu — domyślny projekt aplikacji ASP.NET Web Forms wyposażony jest bowiem we wzorzec, strony zarządzania kontami, kaskadowe arkusze stylów, pliki JavaScript i inne elementy, które poznamy dopiero w dalszej części książki.

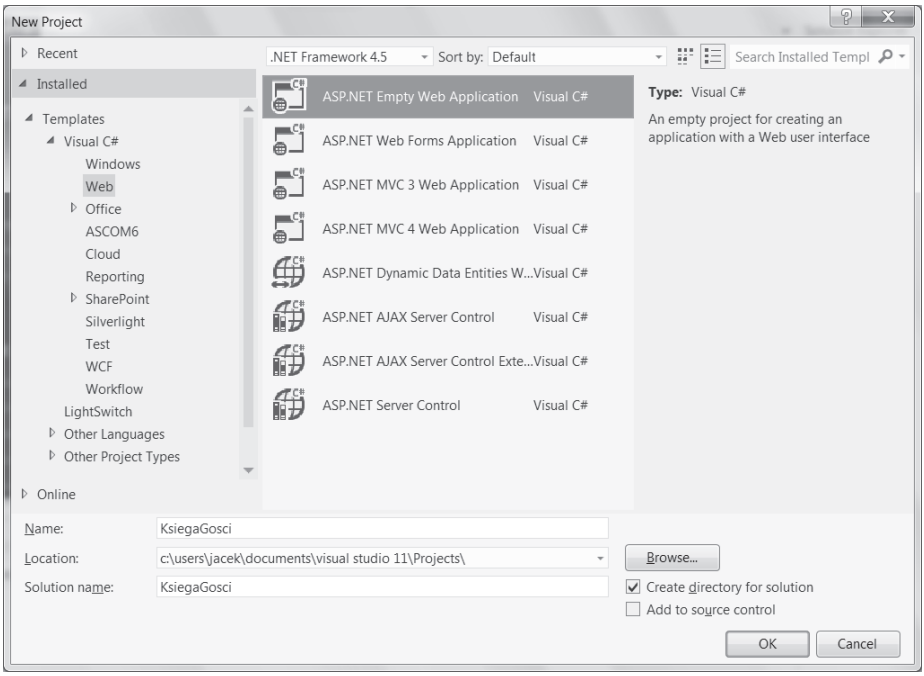
Technologię ASP.NET Web Forms można z powodzeniem zaliczyć obecnie do technologii dojrzałych. Nie następują w niej wobec tego tak duże zmiany jak w przypadku ASP.NET MVC. To powoduje, że opisane w tej książce procedury tworzenia i rozwijania projektów Web Forms mogą być z powodzeniem stosowane w środowisku 2010, 2012 i 2013. Różnice w przypadku poszczególnych środowisk zostały w tekście zaznaczone. Warto zwrócić uwagę na dwie najważniejsze. Po pierwsze, w przypadku Visual Studio 2013 pojawił się zintegrowany kreator umożliwiający tworzenie projektów aplikacji internetowych korzystających z technologii ASP.NET Web Forms i MVC w dowolnych kombinacjach. Tworząc projekty na potrzeby tej książki, wybieramy oczywiście jedynie Web Forms. Drugą zmianą, która może dezorientować, jest sposób edycji tabel w bazach danych (zob. rozdział 6). Od wersji Visual Studio 2012 odbywa się to poprzez generowane specjalnie w tym celu skrypty SQL. W dołączonych do tej książki źródłach znajdują się projekty dla Visual Studio 2010 i 2013 oraz przekonwertowane projekty dla Visual Studio 2013 RC (taka wersja była dostępna w momencie oddawania książki do druku). Aktualizacja projektu z Visual Studio 2012 do 2013 jest automatyczna i pozostawia możliwość otwarcia projektu także w starszej wersji. Ze względu na spójność opisów zrezygnowaliśmy z „polonizacji" Visual Studio, która jest możliwa dopiero od wersji 2012.

Tworzenie pustego projektu w Visual Studio 2010 i 2012

Utwórzmy pusty projekt aplikacji ASP.NET Web Forms:

1. Uruchom Visual Studio.
2. Z menu *File/New* wybierz *Project...* (można również użyć klawiszy skrótu *Ctrl+Shift+N*).
3. Pojawi się okno widoczne na rysunku 1.1, w którym należy z listy zainstalowanych szablonów (*Installed/Templates*) wybrać *Visual C#/Web* i wreszcie *ASP.NET Empty Web Application*.
4. Wpisz `KsiegaGosci` w polu *Name*, sprawdź, czy zaznaczone jest pole opcji *Create directory for solution* (stwórz katalog dla rozwiązania), i kliknij *OK*.

W efekcie powstanie projekt, w którym poza plikiem konfiguracyjnym *Web.config* nie ma żadnych innych plików. Dodajmy wobec tego plik, który będzie szablonem jedynej strony aplikacji (witryny).



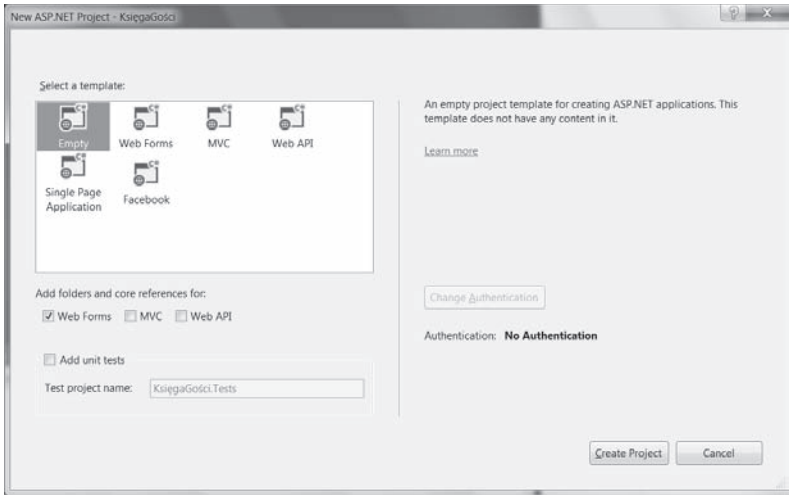
Rysunek 1.1. *Wybór projektu*

Tworzenie pustego projektu w Visual Studio 2013

W najnowszym Visual Studio wygląda to trochę inaczej, choć rozpoczyna się podobnie.

1. Z menu *File/New* należy wybrać pozycję *Project...*.
2. Z listy szablonów w lewym panelu wybieramy *Visual C#/Web*, a następnie jedyną pozycję: *ASP.NET Web Application*.
3. Wskażmy nazwę projektu (pole *Name*), czyli `KsięgaGości`. Upewnijmy się, że pole opcji *Create directory for solution* jest zaznaczone.
4. Po kliknięciu *OK* pojawi się zintegrowany kreator dla wszystkich typów aplikacji internetowych obsługiwanych przez Visual Studio 2013 (rysunek 1.2).
5. W nowym oknie zaznaczmy szablon *Empty*, a poniżej pole opcji *Web Forms* (to ostatnie spowoduje utworzenie folderów typowych dla projektów Web Forms oraz dodanie referencji do bibliotek wykorzystywanych w Web Forms).
6. Klikamy *Create Project*. Utworzenie projektu będzie trwało dłuższą chwilę.

W końcu powstanie projekt, w którym oprócz pliku *Web.config* obecne będą także pliki *Global.asax* i *Global.asax.cs*, a do tego puste katalogi *App_Data* i *Models*.



Rysunek 1.2. *Zintegrowany kreator aplikacji internetowych w Visual Studio 2013*

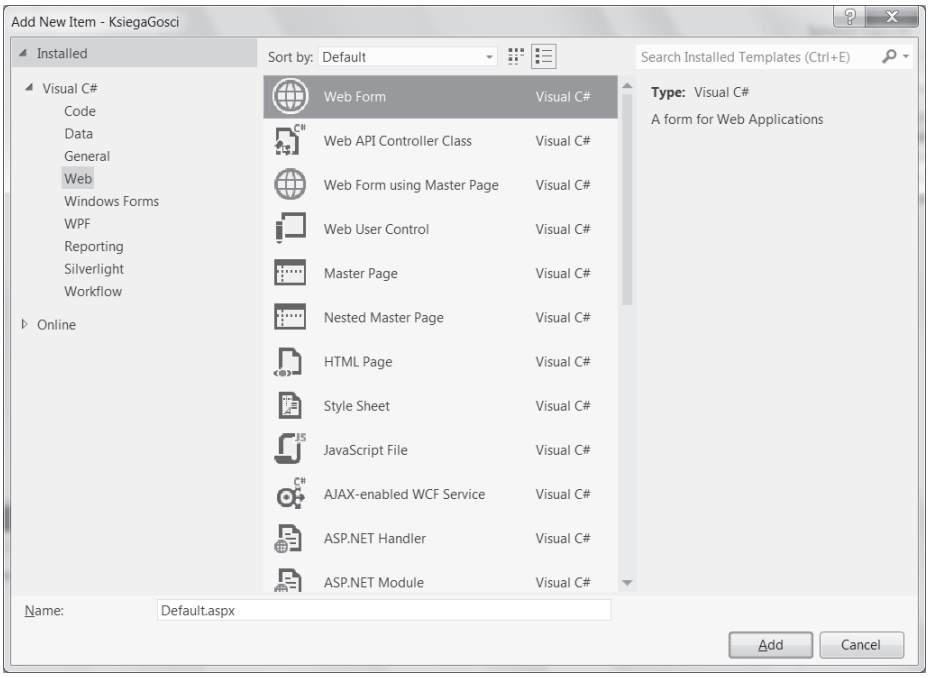
Dodawanie strony .aspx

Dodajmy do projektu stronę, która będzie stanowiła szablon głównej i jedynej strony naszej witryny. Ta czynność, jak zresztą większość czynności związanych z projektowaniem aplikacji Web Forms, jest niemal identyczna w Visual Studio 2010, 2012 i 2013.

1. Z menu *Project* wybierz *Add New Item…* (klawisze skrótu *Ctrl+Shift+A*).
2. Pojawi się okno z rysunku 1.3, w którym z zakładki *Web* wybierz *Web Form*. Zmień nazwę nowego pliku na *Default.aspx*.
3. Kliknij *Add*.

W podoknie *Solution Explorer* pojawił się nowy element — dodany przed chwilą do projektu plik *Default.aspx*. Zauważmy jednak, że reprezentującą go gałąź można rozwinąć. Wówczas zobaczymy dwa towarzyszące mu dodatkowe pliki z kodem C#: *Default.aspx.cs* oraz *Default.aspx.designer.cs*. Do pierwszego będziemy dodawać własny kod C# kontrolujący działanie strony, natomiast w drugim Visual Studio umieszczać będzie deklaracje zmiennych odpowiadających kontrolkom umieszczonym na stronie w oknie projektowania. Osoby poznające dopiero środowisko Visual Studio i język C# nie powinny w tym pliku niczego zmieniać.

Dlaczego wybraliśmy taką nazwę nowego pliku? Serwer IIS traktuje pliki o tej nazwie jako domyślne w katalogach (analogicznie do *index.html* i *Default.htm*) — będzie ona zatem wyświetlona, jeżeli w przeglądarce wpisany zostanie adres zakończony na katalogu aplikacji.



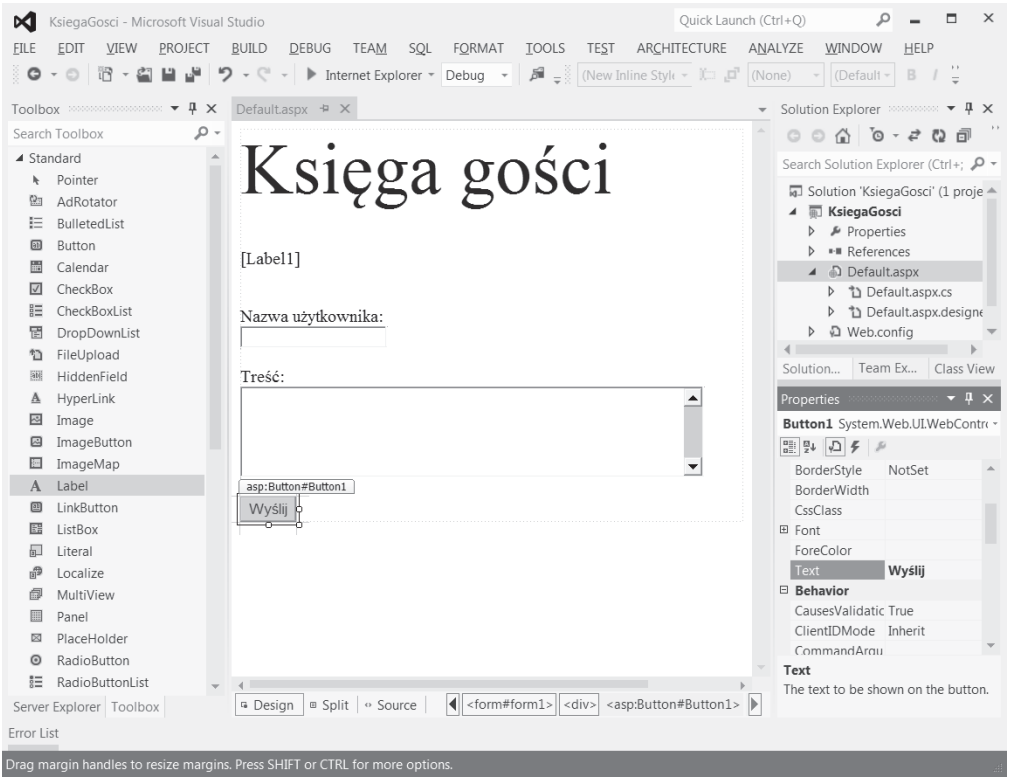
Rysunek 1.3. *Dodawanie pliku do projektu*

Projektowanie interfejsu strony

Po utworzeniu pliku możemy jego zawartość zobaczyć w podoknie widocznym w centrum Visual Studio. Zwróćmy uwagę na obecne w jego dolnej części przyciski *Design*, *Split* i *Source*. Domyślnie edytor ustawiony jest na zakładce *Source*. W wyniku tego oglądamy kod XML będący szablonem kodu HTML wysyłanego przez działającą aplikację ASP.NET do przeglądarki. W tej chwili najważniejszym dla nas znacznikiem jest `div` (znacznik grupujący, który określa blok na stronie). To w nim umieścimy cały dodawany przez nas kod. Zmieńmy widok na *Design*, aby móc projektować wygląd strony za pomocą myszki. Na stronie umieśćmy dwa pola edycyjne (`TextBox`), przycisk (`Button`) i kontrolkę `Label`.

W tym celu:

1. Z lewej strony okna Visual Studio powinno być widoczne podokno *Toolbox*. Jeżeli nie jest — można je włączyć w menu *View*. Przypnijmy je tak, żeby cały czas była widoczna jego zawartość.
2. Z podokna *Toolbox* przeciągnij dwa razy kontrolkę `TextBox`. Należy umieścić je wewnątrz pustego elementu `div` widocznego na podglądzie strony.
3. Możemy również dodać opis pól edycyjnych, wpisując je wprost z klawiatury w kodzie strony (jak w edytorze tekstu). Przykład widoczny jest na rysunku 1.4.



Rysunek 1.4. *Edycja zawartości strony*

Zawartość strony można edytować podobnie jak w edytorach tekstu — strona tworzy dokument, w którym „działają” znaki końca linii. Możemy wobec tego ustawić kursor między dwoma dodanymi polami edycyjnymi `TextBox` i rozdzielić je spacją lub znakiem końca linii (klawisz *Enter*). Nie ma natomiast prostej możliwości ustalania dowolnego położenia kontrolek (ich pozycji na stronie).

4. Tekst można formatować, korzystając z ikon dostępnych po prawej stronie paska narzędzi (niewidoczne na rysunku 1.4). Odpowiadają one typowym ikonom znanym z edytorów tekstu. Znajdziemy tam ikonę pozwalającą na pogrubienie czcionki, użycie kursywy i podkreślenia, zmianę koloru czcionki i tła oraz wyrównania tekstu (do lewej, do prawej, do środka i justowanie). Korzystając z tych ikon, utwórzmy nagłówek strony (rysunek 1.4).

   a) Na górze podglądu strony wpiszmy jej tytuł: „Księga gości” i zaznaczmy go.

   b) Korzystając z paska narzędzi, zmieńmy rozmiar tytułu. W tym celu w rozwijanej liście, w której po rozwinięciu widoczne są względne wielkości (np. *small*, *medium*, *large*), wpiszmy wartość 50.

   c) Zmieńmy kolor czcionki na granatowy. Tło niech pozostanie niezmienione.

d) Przejdźmy do kodu pliku *Default.aspx* (zakładka *Source* u dołu ekranu). Zwróćmy uwagę, że formatowanie zostało zapisane w klasie stylu CSS o nazwie `.style1`[1] (więcej informacji na temat stylów CSS w rozdziale 8.), natomiast tekst tytułu został otoczony znacznikiem `span` z referencją do klasy `style1`.

5. Wróćmy do podglądu strony i zaznaczmy drugie pole edycyjne, a następnie za pomocą podokna *Properties* (jeżeli go nie ma, znajdziemy je w menu *View*) zmieńmy jego właściwość `TextMode` na `MultiLine`. Następnie w podglądzie okna za pomocą myszy zwiększmy jego wysokość.
6. Teraz przeciągnijmy kontrolkę przycisku (`Button`) i zmieńmy jego etykietę (właściwość `Text`) na *Wyślij*.
7. I wreszcie między polami edycyjnymi i nagłówkiem umieśćmy kontrolkę `Label`. W odróżnieniu do zwykłego tekstu wpisywanego w edytorze jej zawartość może być zmieniona z poziomu kodu w trakcie działania aplikacji. Posłuży nam do wyświetlenia zapisów w księdze gości. Na razie wyczyśćmy jej zawartość, usuwając za pomocą podokna *Properties* tekst przypisany do właściwości `Text`.

To tyle, jeżeli chodzi o projektowanie aplikacji ASP.NET za pomocą myszki. Stworzyliśmy interfejs, którego działanie możemy przetestować, uruchamiając aplikację (klawisz *F5*). W polach edycyjnych można wpisywać tekst, kliknięcie przycisku powoduje przeładowanie strony, ale wpisana w formularzu wiadomość nie pojawia się na stronie. To musimy dopiero zaprogramować.

Dodawanie wpisu na stronie

Stwórzmy metodę zdarzeniową przycisku:

1. W widoku projektowania strony (zakładka *Design* u dołu okna) dwukrotnie kliknij przycisk *Wyślij* (na podglądzie strony).
2. Otworzy się edytor kodu z widoczną zawartością pliku *Default.aspx.cs*. Kursor ustawiony jest w nowo utworzonej metodzie `Button1_Click`.
3. W tej metodzie umieść kod widoczny na listingu 1.1.

Korzystamy z języka C# (alternatywą jest Visual Basic). Informacje o tym języku, które są niezbędne do zrozumienia kodów z tej książki, czytelnik znajdzie w dwóch kolejnych rozdziałach.

Listing 1.1. *Reakcja na kliknięcie przycisku Wyślij*

```
protected void Button1_Click(object sender, EventArgs e)
{
    //walidacja po stronie serwera
    if (TextBox1.Text == "" || TextBox2.Text == "") return;
```

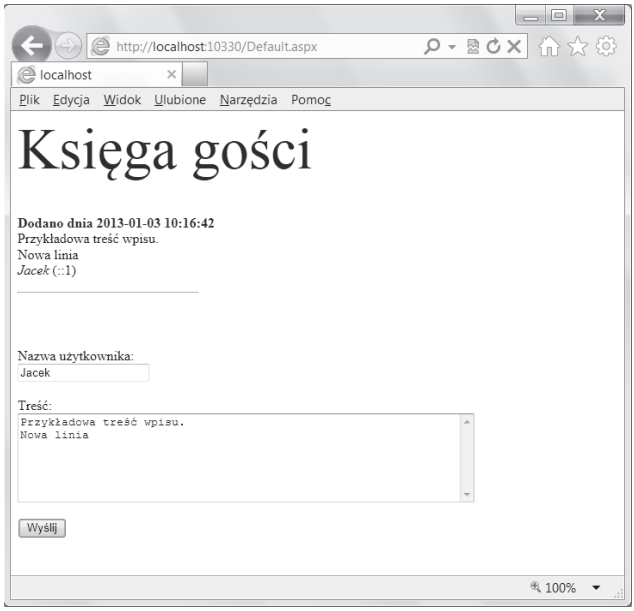
[1] W Visual Studio 2013 styl ten jest nazwany `.auto-style1`.

```
        //ustalanie treści wpisu
        string kolorNagłówka = "navy";
        string nagłówek = "<font color='" + kolorNagłówka +
                          ↪"'><b> Dodano dnia " + DateTime.Now.ToString() + "</b></font>";
        string treść = TextBox2.Text.Trim().Replace("\n", "<br />");
        string podpis = "<i>" + TextBox1.Text + "</i> (" +
                        ↪this.Request.UserHostAddress + ")";

        //dodawanie wpisu
        string nowyWpis = nagłówek + "<br />" + treść + "<br />" + podpis + "<P>";
        nowyWpis += "<hr width='30%' align='left'>";
        Label1.Text += nowyWpis;

        //czyszczenie pola edycyjnego z treścią wpisu
        TextBox2.Text = "";
    }
```

Możemy ponownie uruchomić aplikację (*F5*) i spróbować dodać wpis. Zwróćmy uwagę, że kopiując z pól edycyjnych treść wpisu i pseudonim podany przez użytkownika, uwzględniliśmy formatowanie w postaci znaczników HTML. W szczególności znaki końca linii (`\n`), które mogą znaleźć się w łańcuchu wpisanym do pola edycyjnego `TextBox2`, zamieniliśmy na znacznik `<br />`. Efekt widoczny jest na rysunku 1.5.



Rysunek 1.5. *Wpis w księdze gości. Serwer WWW to IIS 7 uruchomiony na Windows 7*

W nawiasie za podpisem powinien znaleźć się numer IP komputera, z którego dokonano wpisu. W IIS 6 znaleźlibyśmy tam 127.0.0.1, ale w IIS 7 widać tylko numer portu. Po przeniesieniu programu na właściwy serwer numer będzie jednak wyświetlany poprawnie.

Powyższe rozwiązanie ma zasadniczą wadę. Wpisy przechowywane są tylko we własności `Text` kontrolki `Label1`, więc po zamknięciu aplikacji znikną. To jednak nie wszystko — co gorsza, nawet w trakcie działania aplikacji inna osoba, która wejdzie

na tę stronę, nie zobaczy wpisów zostawionych przez innych użytkowników. Można się o tym łatwo przekonać, uruchamiając dodatkową sesję przeglądarki i kopiując adres strony z okna uruchomionego przez Visual Studio (zapewne coś w stylu *http://localhost:* ➥*5262/Default.aspx*, tylko z innym numerem portu). Aby ten błąd naprawić, konieczne jest współdzielenie wpisów na poziomie całej aplikacji, a ponadto trwałe ich magazynowanie na dysku serwera — w bazie danych, pliku XML lub choćby zwykłym pliku tekstowym.

Rozwój tej aplikacji będę opisywał nie tylko w tym, ale również w kilku następnych rozdziałach. Nie będzie to rozwój dążący wprost do „jedynie słusznego" rozwiązania — będę meandrował, opisując różne rozwiązania i podejścia. Proszę wobec tego traktować go jedynie jako pretekst do prezentacji ASP.NET Web Forms, a nie jako wzór projektowania aplikacji internetowych.

Znaczniki tradycyjnego HTML mogą być pisane dużymi lub małymi literami. Nową linię można zatem zapisać równoważnie jako `<BR />` lub `<br />`. Warto jednak zwrócić uwagę, że specyfikacja XHTML, a za nią HTML 5 wymusza zapisywanie znaczników małymi literami. Będę starał się do tego zalecenia stosować. Pozwolę sobie jednak na innego typu anachronizmy: m.in. będę używał wycofywanego znacznika FONT, a także nie zawsze będę dbał o domykanie pojedynczych znaczników (np. `<p>` zamiast `<p />`).

Czas życia aplikacji

Po pierwszym wywołaniu witryny na serwerze WWW uruchamiana jest odpowiedzialna za nią aplikacja ASP.NET, która od tego momentu najczęściej nie przestaje działać aż do zakończenia pracy serwera. Jednocześnie „w obrębie aplikacji" powstaje sesja, która związana jest z tym pierwszym żądaniem. Kolejne żądania od innych przeglądarek-klientów powodują tworzenie kolejnych sesji. Po opuszczeniu przez internautę witryny sesje kończą pracę, ale aplikacja wciąż jest na posterunku. To zasadnicza zmiana w porównaniu do starszych technologii rozszerzeń serwerów WWW, w których aplikacja uruchamiana była tylko po to, aby przetworzyć otrzymane od przeglądarki żądanie i wygenerować nowy kod HTML. W ASP.NET mamy do czynienia ze stale pracującą aplikacją, która przechowuje swój stan. Ciągłość pracy aplikacji powoduje w szczególności, że możemy zapamiętać jakieś dane na jednej stronie witryny, a wykorzystać na innej. Do tego służy zbiór danych (zmiennych) aplikacji. Nie ma także problemu, aby utworzyć zmienną przechowywaną tylko w obrębie jednej sesji. Wystarczy umieścić ją w zbiorze danych sesji.

Skorowidz

A

ACT, AJAX Control Toolkit, 265
AJAX Control Toolkit, 265
AJAX, Asynchronous JavaScript and XML, 253
algorytm bąbelkowy, 260
aliasy, 124
analiza danych, 92
aplikacja Kolory, 272, 274
aplikacje
  ASP.NET Web Forms, 15
  ASP.NET, 146
  desktopowe, 58
  internetowe, 58, 277
argumenty
  nazwane, 55
  opcjonalne, 54
ASP, Active Server Pages, 15
ASP.NET, 15
  Cache, 288
  Web Forms, 277, 291
atrybuty, 76

B

baza danych, 195
  SQL Server, 151, 154
biblioteka
  jQuery, 263
  Web Forms, 195
  .NET, 38
bloki, 75
błędy, 32, 188
buforowanie, caching, 288

C

C# 5.0, 37–104
cache, 288
Caller Information, 103
CAS, Code Access Security, 39
CGI, Common Gateway Interface, 15
ciasteczka, 30
CLR, Common Language Runtime, 38, 39
CLS, Common Language Specification, 39
CSS, Cascade Style Sheet, 169
CTS, Common Type System, 39
czas życia aplikacji, 23

D

dane aplikacji, 24
delegacja, 127
delegacje, 57
DLR, Dynamic Language Runtime, 38, 39, 68
dodawanie
  do siatki, 212
  elementu Site Map, 281
  odwołania do usługi, 146
  strony .aspx, 18
  wpisu na stronie, 21
  zdjęć, 197
domyślna wartość argumentu, 54
domyślnie implementowane właściwości, 112
dostęp do stron, 188
dyrektywy preprocesora, 73
działanie bufora, 290
działanie polityki, 290
dziedziczenie, 243

E

elementy
  aplikacji internetowych, 277
  tablicy, 78
  typu T, 121
encje, 156

F

FIFO, 84
FILO, 84
filtrowanie, 91, 210
formatowanie
  identyfikator, 173
  nazwa klasy, 174
  typ znacznika HTML, 170

G

generowanie ostrzeżenia, 74
gra Reversi, 215
  dziedziczenie, 243
  informacje o stanie, 231
  interakcja z użytkownikiem, 234
  interfejs, 216, 233
  metody dodatkowe, 225
  najlepszy ruch, 245
  plansza, 226, 229
  podpowiedź komputera, 248
  prezentacja stanu, 230
  rozpoczynanie od nowa, 243
  ruch komputera, 251
  silnik, 216
  testy, 223
  wykrywanie sytuacji, 238
  zasady, 221
gra z komputerem, 250

H

hasło, 187, 188

I

IIS, Internet Information Services, 15, 288
IL, Intermediate Language, 38
implementacja
  interfejsów, 117, 122
  zasad gry, 221
informacja
  o błędach, 188
  o koncie, 191
  o użytkowniku, 193
inicjacja
  obiektów, 88
  tablic, 88
instalowanie kontrolek ACT, 268
instancja klasy, 105
instrukcja
  warunkowa if..else, 68
  wyboru switch, 68
interfejs
  IComparable, 118
  strony, 19
  użytkownika, 254, 258

J

język C# 5.0, 37–104
język XML, 133
JIT, Just-In-Time, 39, 40
jQuery, 263
JSON, 253
JSP, Java Server Pages, 15

K

kaskadowe arkusze stylów, 169, 283
klasa
encji, 156
  Lazy, 51
  Random, 86
  ReversiSilnik, 234
  ReversiSilnikAI, 246
  String, 47
  StringBuilder, 49
  Wpisy, 130, 158
klasy, 105
klauzula OutputCache, 291
kolejki, queue, 84
kolekcja
  List, 81
  SortedList, 83
kolekcje, 76
kolor panelu, 272, 273
kompilacja warunkowa, 74
komponent
  ConfirmButton, 266
  ValidationSummary, 266
konfiguracja
  czcionki, 173
  filtra, 210
  uwierzytelniania, 181
konfigurator witryny, 187
konstrukcja
  try..catch, 71
  yield return, 87
konstruktor, 108, 220
konto użytkownika, 191
kontrakt danych, 143
kontrakty, 143
kontrolka
  AlwaysVisibleControl ↪Extender1, 275
  DetailsView, 209, 211
  GridView, 209
  Label, 255
  LoginView, 184
  Repeater, 161, 207, 211
  ScriptManager, 256
  Timer, 257
  TreeView, 282
  UpdatePanel, 254, 255
  UpdateProgress, 259
kontrolki, 160, 173
  ACT, 266, 268
  logowania, 180
  uwierzytelniania, 177
  Web Forms, 195
konwersja
  na łańcuch, 109
  na typ double, 109
  typów podstawowych, 45

L

leniwe inicjowanie zmiennych, 51
liczby pseudolosowe, 86
LINQ, 88, 89
  to Objects, 136
  to SQL, 151
  to XML, 85, 134, 137
logowanie, 179
lokalizacja, 163, 164, 166, 167, 168

Ł

łańcuchy, 47, 48
łączenie
  danych, 94
  zbiorów danych, 93

M

mapa witryny, SiteMap, 280
mapowanie obiektowo-relacyjne, 157
mechanizm reflection, 67
menu aplikacji, 209
metoda CompareTo, 118
metoda Equals, 114
metoda GetHashCode, 114
metoda GetValueOrDefault, 64
metoda ToDouble, 109
metoda ToString, 109
metody, 52, 55, 105
metody klasy String, 47, 48
metody rozszerzające, 90
metody strony, 263
metody strony, page methods, 262
metody udostępniane przez usługę, 144
metody z nieokreśloną liczbą argumentów, 85
metody zdarzeniowe, 58
miniatura zdjęcia, 201
modyfikacja konstruktora kontrolki, 235
modyfikacje danych, 95
modyfikator async, 99
modyfikatory dostępu, 107

N

nazwa użytkownika, 183

O

obiekty, 105
obiekty stałe, 108
obsługa
  wyjątków, 71, 73
  zdarzeń, 256
odczyt z pliku, 137
odświeżanie danych, 162
określanie typu zmiennej, 43
operator as, 46

await, 99
is, 46
join, 94
new, 77
operatory, 43, 45
arytmetyczne, 113
konwersji, 116
LINQ, 89
porównania, 114
osadzanie modelu, 131
ostrzeżenie, 74

P

parametr Query String, 287
parametr T, 121
parametry, 121
pętla
foreach, 79
równoległa for, 96
pętle, 69
platforma .NET, 15, 37
środowisko uruchomieniowe, 38
plik
AssemblyInfo.cs, 76
Default.aspx, 184, 191
Global.asax.cs, 27
Web.config, 17, 184, 190, 277
pliki
.css, 169
.Master, 177
konfiguracyjne, 277, 279
XML, 133
pobieranie danych, 91
pola, 105
polityka sliding expiration, 290
prezentacja danych, 91, 160
programowanie
asynchroniczne, 99
obiektowe, 105
współbieżne, 96
projekt ASP.NET Web Forms, 16
projektowanie interfejsu
strony, 19
przechowywanie danych, 30
przechowywanie stanu
aplikacji, 26
przeciążanie metod, 53
przerywanie pętli, 98
przestrzeń
System.Collections.Generic, 84
przesyłanie
plików, 204
zdjęcia, 202
pseudotyp var, 43
pudełkowanie, 64
pusty projekt, 17

Q

Query String, 286

R

refleksja, 68
rejestrowanie użytkownika, 185
rejestrowanie zdarzeń, 139
reklama, 274
rozszerzanie klasy, 126
rozszerzenia, 125
AlwaysVisibleControlExtender, 274
ConfirmButtonExtender, 269, 270
równoległa pętla for, 96

S

separacja modelu, 129
siatka, 212
SiteMap, 280
skórki, 285
słowniki, 83
słowo kluczowe
default, 64
delegate, 57
event, 58
using, 148
yield, 86
sortowanie, 80, 91, 123
bąbelkowe, 260
tablicy, 118
SQL Server, 151
stałe liczbowe, 42
stałe preprocesora, 74
stan
aplikacji, 26
gry, 230
statyczne obiekty składowe, 108
sterowanie przepływem, 68
stos, stack, 84
strona
.aspx, 18
logowania, 179
przeglądania zbiorów zdjęć, 204
przypominania hasła, 187
rejestrowania użytkownika, 185
tworzenia użytkownika, 182
z formularzem, 197
zmiany hasła, 188
strony wielojęzyczne, 163
struktura
typu FIFO, 84
typu FILO, 84
Ulamek, 106
Wpis, 156
struktury, 106
styl kontrolki, 174
suwaki, 272
szablony, 166
Style Sheet, 283
WCF Data Service, 141

T

tabela Zdjecia, 206
tablice, 77
dynamiczne, 77
jako argumenty, 85
technologia
AJAX, 253
ASP.NET, 15
ASP.NET Web Forms, 16
COM, 68
LINQ, 90
LINQ to SQL, 151
LINQ to XML, 85
tematy, 283
testowanie struktury, 109
testy działania silnika, 224
tłumaczenia, 165
tworzenie
bazy danych, 152, 154
biblioteki usług WCF, 141, 142
konta, 186
projektu, 106
pustego projektu, 16, 17
tablic, 77
wiązania, 198
wzorca, 177
typ
dynamic, 66
Nullable<int>, 64
object, 67
T, 121
Variant, 65
wyliczeniowy, 50
typy
anonimowe, 127
danych, 40, 42
dynamiczne, 65

typy
ogólne z wieloma parametrami, 124
ogólne, generic types, 119
parametryczne, 119
pętli, 70
referencyjne, 62
wartościowe, 62

U

udostępnianie danych, 141
upraszczanie ułamków, 110
usługa WCF, 141, 147, 149
uwierzytelnianie użytkowników, 177, 181

W

walidacja, 31, 34, 267
po stronie serwera, 162
walidator
RegularExpressionValidator, 34
wartość null, 63
WCF, 141, 149
Data Service, 141
Service Library, 141
Web Forms, 195
weryfikowanie danych, 92
wiązanie
kontrolki, 199
rozszerzeń ACT, 269
rozwijanej listy, 200
właściwości, 111
domyślnie implementowane, 112
wpis na stronie, 21
wybór
elementu, 92
języka, 168
wygląd strony, 199
wyjątki, exception, 71, 72
wyrażenia lambda, 60
wyświetlanie nazwy użytkownika, 183
wyzwalacze, triggers, 254
wzorzec, 178

Z

zapis
do pliku, 138
kolekcji, 133
stanu, 29
zapytania LINQ, 89
zdarzenia, 57, 139, 256
aplikacji, 25
zdjęcia, 195
miniatura, 201
przesyłanie, 202
uploadowanie, 197
zbiory, 204
zmiana hasła, 188
zmienne, 40
znaczniki HTML, 170
znak hash, 126
zwracanie wartości
przez argument, 55

ASP.NET Web Forms

Kompletny przewodnik dla programistów interaktywnych aplikacji internetowych w Visual Studio

Jeszcze kilkanaście lat temu projektowanie aplikacji (nie tylko internetowych) wymagało nie lada wysiłku. Sytuacja zmieniła się dość radykalnie wraz z rozwojem technologii, która pozwala programiście budować aplikację z gotowych kontrolek. Programista ma w ten sposób do dyspozycji zestawy gotowych funkcjonalności, które można łatwo dodać do tworzonej aplikacji. Jedną z najczęściej używanych bibliotek jest ASP.NET Web Forms, która swoją popularność zawdzięcza podobieństwu do Windows Forms i równie dużej jak w przypadku pierwowzoru łatwości stosowania. To właśnie tę bibliotekę wybierają programiści, gdy w krótkim czasie chcą stworzyć niewielkie, niezbyt skomplikowane aplikacje, jak również przy tworzeniu prototypów w większych projektach.

Ta książka zawiera wszystkie najważniejsze informacje pozwalające odkryć i wykorzystać zalety biblioteki Web Forms. Znajdziesz tu opis języka C# i programowania obiektowego w tym języku, a także dowiesz się, do czego przydaje się właściwa separacja modelu aplikacji. Odkryjesz, jak implementować usługę sieciową z użyciem technologii WCF, jak korzystać z baz danych i kaskadowych arkuszy stylów oraz do czego służy technologia AJAX. Ponadto poznasz kontrolki Web Forms, a potem zobaczysz, jak zastosować te informacje w praktyce — na przykładzie gry planszowej. Jeśli chcesz szybko i swobodnie projektować niewielkie aplikacje, musisz przeczytać tę książkę!

- Prosta aplikacja ASP.NET Web Forms, czyli o wszystkim po trochu
- Język C# 5.0
- Programowanie obiektowe w C#
- Wielkie porządki, czyli separacja modelu. Pliki XML
- Udostępnianie danych przez usługę WCF
- Baza danych SQL Server w ASP.NET Web Forms. LINQ to SQL
- Podstawowe wiadomości o kaskadowych arkuszach stylu
- Kontrolki uwierzytelniania użytkowników w Web Forms
- Przegląd kontrolek biblioteki Web Forms
- Studium przypadku: gra Reversi
- Technologia AJAX i AJAX Control Toolkit
- Zgodność z Visual Studio w wersjach 2010, 2012 i 2013

Wypróbuj możliwości Web Forms

— tam, gdzie się sprawdzą najlepiej!

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

ISBN 978-83-246-8284-3

9 788324 682843

Cena: 59,00 zł

Informatyka w najlepszym wydaniu