ASP.NET

Marcin Szeliga

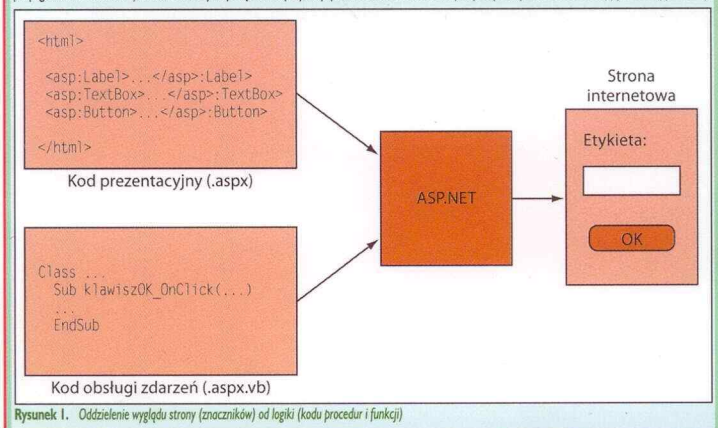
TABLICE INFORMATYCZNE

Helion

Pod adresem *http://msdn.microsoft.com/library/default.asp*, w sekcji *Web Development > Server Technologies* jest ogólnie dostępna pomoc dotycząca ASP.NET — może z niej skorzystać każdy, kto nie dysponuje lokalnie zainstalowaną biblioteką MSDN. Zawiera ona między innymi bardzo dokładny opis poszczególnych kontrolek, klas, ich atrybutów i metod.

WPROWADZENIE

ASP.NET to technologia tworzenia dynamicznych stron internetowych, w pełni wykorzystująca funkcjonalność platformy .NET Framework oraz wspólnego środowiska uruchomieniowego CLR. Głównym celem przyświecającym powstawaniu ASP.NET było uproszczenie procesu tworzenia aplikacji internetowych tak bardzo, jak to tylko możliwe. Osiągnięto to poprzez implementację programowania zdarzeniowego. Programiści dodają do formularza kontrolki i piszą kod, który zostanie wykonany wtedy, gdy wystąpi określone zdarzenie związane z tymi kontrolkami (np. załadowanie lub opuszczenie strony, wpisanie tekstu do pola lub kliknięcie przycisku). Programowanie zdarzeniowe umożliwia także oddzielenie kodu „logicznego" aplikacji od kodu prezentacji danych (HTML). Zabieg rozdzielenia kodu logicznego i kodu prezentacji danych określany jest jako schowanie kodu (ang. *Code behind*). Dzięki temu wiele stron może wykorzystywać ten sam kod logiczny do prezentacji różnych danych, co ułatwia tworzenie i późniejsze zarządzanie stronami (zmian kodu nie trzeba propagować na wiele stron), a kod źródłowy staje się bardziej czytelny, ponieważ kod ASP.NET nie jest przeplatany z kodem formatującym dane (rysunek 1).



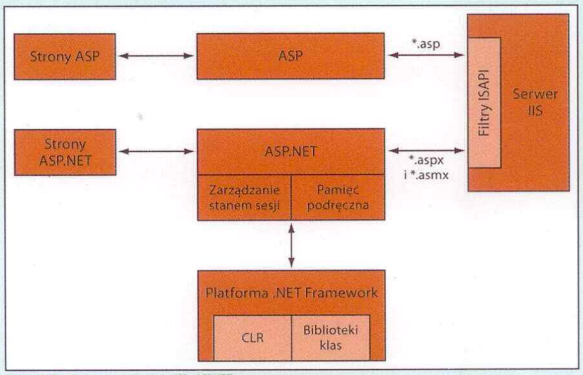
Rysunek 1. *Oddzielenie wyglądu strony (znaczników) od logiki (kodu procedur i funkcji)*

Architektura ASP.NET

ASP.NET służy do tworzenia aplikacji internetowych, które działają na serwerze internetowym Microsoft IIS i wykorzystują protokoły internetowe takie jak HTTP i SOAP. ASP.NET komunikuje się z serwerem IIS w taki sam sposób, w jaki komunikowało się ASP, czyli poprzez interfejs ISAPI. ASP.NET w pełni wykorzystuje środowisko .NET Framework — wspólne środowisko uruchomieniowe CLR kompiluje i zarządza wykonywaniem kodu stron, a biblioteki klas udostępniają klasy narzędziowe (takie jak Web Forms i XML) obsługujące przyjmowanie żądań i generowanie odpowiedzi (rysunek 2).

Do najważniejszych usług zaimplementowanych w samej technologii należą:

1. Ścisła kontrola typów.
2. Kompilacja na żądanie — strony ASP.NET są kompilowane do niezależnego od procesora kodu pośredniego MSIL, który w czasie działania aplikacji jest kompilowany do kodu natywnego.
3. Zarządzanie pamięcią — programista nie musi martwić się przydzielaniem i zwalnianiem pamięci.
4. Strukturalna obsługa wyjątków — błąd może być przechwycony i obsłużony na poziomie kontrolki, strony czy całej aplikacji WWW.
5. Możliwość buforowania w pamięci podręcznej wybranych danych lub całych stron.
6. Zarządzanie stanem sesji użytkownika (ang. *Session State Management*).

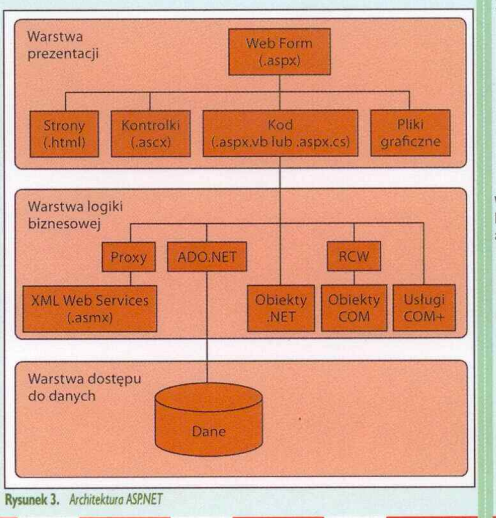


Rysunek 2. *Porównanie technologii ASP i ASP.NET*

ASP.NET umożliwia tworzenie i wdrażanie aplikacji internetowych dwóch typów:

1. Aplikacji Web Forms — dynamicznie generowanych stron WWW. Ich tworzenie jest dużo prostsze dzięki zawierającym wiele kontrolek bibliotekom klas platformy .NET Framework. Wystarczy, że programista w kodzie strony wpisze odpowiedni znacznik (np. `<asp:Buton>`) lub przeciągnie odpowiednią kontrolkę w edytorze graficznym, takim jak Visual Studio.
2. Usług internetowych XML Web Services — udostępniających funkcjonalność innym aplikacjom i umożliwiających wymianę danych pomiędzy aplikacjami.

ASP.NET doskonale nadaje się do tworzenia aplikacji wielowarstwowych — utworzony w ASP.NET interfejs użytkownika może łączyć się z najróżniejszymi obiektami warstwy logiki biznesowej lub bezpośrednio ze źródłami danych (rysunek 3).



Rysunek 3. *Architektura ASP.NET*

Edytory

Programista ASP.NET ma do dyspozycji albo Visual Studio (w tym darmową, dostępną pod adresem *http://www.microsoft.com/poland/msdn/vstudio/express/default.mspx* wersję Express), albo dedykowany do tworzenia aplikacji WWW, łatwiejszy do opanowania, również darmowy, program Visual Web Developer (*http://www.microsoft.com/poland/msdn/vstudio/express/vwd/default.mspx*).

Struktura projektu

Projekt aplikacji WWW może składać się z następujących plików:

1. Formularzy WWW (pliki *.aspx*) — definiują wygląd stron.
2. Plików kodu (*.vb* lub *.cs*) — zawierają kod związany z określoną stroną WWW (plikiem *.aspx*) lub usługą sieciową (plikiem *.asmx*). Podwójne rozszerzenie wskazuje na język, w którym kod został napisany (VB .NET lub C#). **W wersji 2.0 kod podzielony jest pomiędzy kilka plików** (ang. *Partial Classes*), **dzięki czemu kod wygenerowany przez Visual Studio jest ukryty i oddzielony od kodu użytkownika.**
3. Usług sieciowych (pliki *.asmx*) — definiują usługi sieciowe, które są używane przez inne programy, a nie bezpośrednio przez użytkowników.
4. Plików opisujących usługi WWW (pliki *.vsdisco* lub *.disco*) — dokumenty XML zawierające informacje, które pozwalają programom klienckim odczytać dodatkowe informacje o usłudze sieciowej.
5. Pliku klas globalnych (plik *Global.asax*) — w pliku aplikacji ASP.NET znajdują się kody procedur zdarzeń poziomu sesji lub aplikacji, takich jak nawiązanie sesji przez użytkownika czy zatrzymanie aplikacji WWW.
6. Plików zasobów (pliki *.resx*) — przechowują wszystkie, z wyjątkiem wykonywalnych, zasoby aplikacji, takie jak grafiki, pliki pomocy czy wersje językowe opisów kontrolek.
7. Pliku arkuszy stylów (plik *Styles.css*) — zawiera definicję domyślnego arkusza stylów aplikacji ASP.NET.
8. Plików konfiguracyjnych (plików *Web.config*) — dokumentów XML, których poszczególne sekcje odpowiadają opcjom konfiguracyjnym aplikacji WWW.

Dodatkowo podczas kompilacji dodawane są kolejne pliki:

1. Biblioteki DLL (pliki *.dll*) — zawierają skompilowaną do postaci kodu pośredniego logikę aplikacji WWW.
2. Nagłówki (pliki *AssemblyInfo.vb* lub *AssemblyInfo.cs*) — zawierają opis aplikacji, m.in. numer wersji bibliotek.

Pliki .aspx

Każda strona *.aspx* jest pochodną klasy `Page` i dziedziczy wszystkie jej publiczne metody i atrybuty. Dodając do projektu stronę ASP.NET, automatycznie tworzymy nową klasę o nazwie odpowiadającej nazwie tworzonej strony WWW:

Visual Basic .NET
```
Public Class WebForm1 Inherits System.
Web.UI.Page
```
C#
```
public class WebForm1 : System.Web.UI.Page
```

Gdy będziemy tworzyć nową stronę w którymś z polecanych edytorów, kreator automatycznie doda do niej dyrektywę `@ Page` i umieści w niej wymagane przez parser i kompilator atrybuty, między innymi nazwę pliku kodu:

Visual Basic .NET
```
<%@ Page Language="VB"
AutoEventWireup="false"
Codebehind="WebForm1.aspx.vb"
Inherits="WebApplication1.WebForm1"%>
```
C#
```
<%@ Page Language="c#"
AutoEventWireup="false"
Codebehind="WebForm1.aspx.cs"
Inherits="WebApplication1.WebForm1"%>
```

Jak to wszystko razem działa?

Sposób zrealizowania żądania klienta (wysłania strony WWW do przeglądarki) zależy od tego, czy jest to pierwsze żądanie, czy też dany zasób był już wcześniej pobierany. W przypadku pierwszego żądania:

1. Przeglądarka internetowa klienta wysyła do serwera WWW żądanie `GET HTTP`.
2. Parser ASP.NET przygotowuje kod źródłowy strony.
3. Jeżeli kod strony nie był wcześniej skompilowany do postaci biblioteki DLL, wywołany zostanie kompilator (kompilacja na żądanie).
4. Wspólne środowisko uruchomieniowe CLR ładuje i wykonuje odpowiedni kod.
5. Wygenerowany przez stronę ASP.NET kod HTML jest odsyłany do klienta.

W przypadku ponownego odwołania się do tego samego zasobu proces realizacji jest znacznie szybszy:

1. Przeglądarka internetowa klienta wysyła do serwera WWW żądanie `GET HTTP`.
2. Wspólne środowisko uruchomieniowe CLR ładuje i wykonuje wcześniej przygotowany kod.
3. Wygenerowany przez stronę ASP.NET kod HTML jest odsyłany do klienta.

Wygenerowane strony mogą być też buforowane — wtedy identyczne żądania klienckie w ogóle nie muszą być przetwarzane przez aplikację WWW. Zamiast tego zbuforowana kopia strony jest odsyłana do klienta.

INTERFEJS UŻYTKOWNIKA

Przygotowanie graficznego interfejsu użytkownika

Na graficzny interfejs użytkownika składają się wszystkie wizualne elementy strony (kontrolki), takie jak pola tekstowe, przyciski czy tabelki. Istnieją dwa typy kontrolek strony serwera:

1. Odpowiadające typowym elementom standardu HTML **kontrolki HTML** — ich definicje znajdują się w przestrzeni nazw `System.Web.UI.HtmlControls`.
2. Abstrakcyjne, bardzo różnorodne, przypominające kontrolki formularzy Windows i oferujące większą funkcjonalność (na przykład możliwość automatycznego wykrycia i dostosowania do możliwości przeglądarki) **kontrolki Web** — ich definicje znajdują się w przestrzeni nazw `System.Web.UI.WebControls`.

Znaczniki HTML obu typów kontrolek są podobne i przypominają standardowe znaczniki HTML, z tym że wszystkie kontrolki strony serwera muszą zawierać atrybut `runat=server`. Kontrolki Web dodatkowo muszą mieć prefiks *asp*:

```
<asp:Button id="btnOK" runat="server"
Text="Wyślij"></asp:Button>
<asp:button id="Enter" Text="Enter"
OnClick="Enter_Click"
runat="server"> </asp:button>
<input id="OK" type="button" value="OK"
OnServerClick="OK_Click" runat="server">
```

Aby dodać kontrolkę do formularza WWW:

1. W widoku strony przeciągnij odpowiednią kontrolkę z paska narzędzi (ang. `Toolbox`).
2. W widoku HTML zdefiniuj kontrolkę albo (dotyczy wersji 2.0) przeciągnij ją z paska narzędzi.
3. W pliku kodu strony dynamicznie utwórz odpowiednią kontrolkę. W tym celu:
   a. Umieść na formularzu WWW kontrolkę `PlaceHolder`.
   b. Utwórz nową instancję klasy kontrolki w pliku kodu.
   c. Dodaj utworzoną kontrolkę do kolekcji kontrolek `PlaceHolder`:

Visual Basic .NET
```
Dim mLabel as Label = new Label()
PlaceHolder1.Controls.Add(mLabel)
```
C#
```
Label mLabel = new Label();
PlaceHolder1.Controls.Add(mLabel);
```

Wygląd i funkcjonalność kontrolek, szczególnie kontrolek Web, może być zmieniana zarówno w trybie projektowania, jak i w trybie działania aplikacji WWW. Kontrolki możemy konfigurować:

1. Za pomocą okienka właściwości (ang. `Properties`) — po wybraniu kontrolki zobaczymy w nim wszystkie atrybuty kontrolki i ich wartości.
2. W widoku HTML, zmieniając definicje kontrolki.
3. Zmieniając styl całej strony lub kontrolki.
4. Programowo ustawiając atrybuty kontrolki:

Visual Basic .NET
```
Calendar1.DayStyle.BackColor = Color.Green
```
C#
```
Calendar1.DayStyle.BackColor = Color.Green;
```

5. Definiując obiekt typu `Style` i przypisując tak utworzony styl kontrolkom. W takim przypadku należy:
   a. Utworzyć instancję klasy `Style` i nadać wartości odpowiednim atrybutom:

Visual Basic .NET
```
Dim s As Style = New Style()
s.BackColor = Color.Red
```
C#
```
Style s = new Style();
s.BackColor = Color.Red;
```

   b. Przypisać utworzony styl kontrolce. Możemy do tego użyć metody `CopyFrom` — w takim przypadku wszystkie atrybuty stylu zostaną zastosowane, nawet jeżeli nie nadaliśmy im wartości, albo zastosować metodę `MergeWith` — w takim wypadku nieustawione wartości stylu nie zostaną zastosowane:

Visual Basic .NET
```
Calendar1.SelectedDayStyle.CopyFrom(s)
DataGrid1.HeaderStyle.MergeWith(s)
```
C#
```
Calendar1.SelectedDayStyle.CopyFrom(s);
DataGrid1.HeaderStyle.MergeWith(s);
```

Zmiany przeprowadzone w okienku właściwości będą natychmiast widoczne w widoku HTML strony i na odwrót. W wersji 2.0 zmiany nie wpłyną na układ (formatowanie) znaczników HTML.

Obsługa zdarzeń

Większość kontrolek ma zdefiniowane domyślne zdarzenie — odpowiada ono typowej dla tej kontrolki akcji użytkownika lub systemu (na przykład domyślnym zdarzeniem przycisku polecenia jest jego kliknięcie). Zadaniem programisty jest przechwycenie i oprogramowanie zajścia zdarzenia, czyli napisanie procedury obsługi zdarzenia. Możemy to zrobić:

1. **Dla domyślnego zdarzenia** dwukrotnie klikając lewym przyciskiem myszy kontrolkę.
2. **Dla innych zgłaszanych przez kontrolkę zdarzeń**:
   a. W przypadku projektu VB.NET w wersji 1.x otwierając plik kodu i wybierając z widocznych w górnej części okienka edytora kodu pól listy kontrolkę i zdarzenie.
   b. W przypadku projektu C# lub projektów w wersji 2. klikając widoczny w okienku właściwości przycisk *Events* i dwukrotnie klikając wybrane zdarzenie albo wpisując przy nazwie zdarzenia nazwę obsługującej je procedury.

Możemy również scentralizować obsługę zdarzeń — jeżeli w formularzu znajduje się więcej kontrolek tego samego typu lub realizujących te same zadania, możemy napisać jedną procedurę zdarzenia i powiązać ją ze zdarzeniami zgłaszanymi przez wiele kontrolek. **Jest to możliwe tylko wtedy, jeżeli wszystkie obsługiwane zdarzenia mają taki sam nagłówek**:

Visual Basic .NET
```
Sub ProceduraObslugiWieluZdarzen (ByVal
sender as Object, ByVal e as EventArgs) _
Handles Button1.Click, Button2.Click,
Button3.Click
```
C#
```
this.Button1.Click += new
System.EventHandler(this.ProceduraObslugi
WieluZdarzen);
this.Button2.Click += new
System.EventHandler(this.ProceduraObslugi
WieluZdarzen);
this.Button3.Click += new
System.EventHandler(this.ProceduraObslugi
WieluZdarzen);
Private void ProceduraObslugiWieluZdarzen
(object sender, System.EventArgs e)
```