Paweł Janowski, Adam Pelikant
Wyższa Szkoła Informatyki, Katedra Inżynierskich
Zastosowań Informatyki, 93-008 Łódź, ul Rzgowska 17a
email: p.janowski@list.pl,
apelikan@wsinf.edu.pl

SYSTEM OBIEGU WNIOSKÓW JAKO PRZYKŁAD ZASTOSOWANIA ASP.NET 2.0 ORAZ MS SQL 2005 EXPRESS

Streszczenie – Artykuł ma na celu przedstawienie nowego podejścia do pisania aplikacji, które zapewnia wysoki poziom użyteczności oraz bezpieczeństwa przy niskich nakładach związanych z programowaniem i wdrożeniem. W chwili obecnej połączenie ASP.NET 2.0 oraz SQL Server 2005 jest bez wątpienia najsilniejszym i najwydajniejszym środowiskiem programistycznym dostępnym na rynku. Prezentację najważniejszych możliwości oferowanych przez wspomniane narzędzia oparto o system zarządzający obiegiem wniosków. Pomimo, że aplikacja jest w pełni użyteczna i dostosowana do zastosowań komercyjnych stanowi tylko ilustrację prezentowanych metod.

1 Wstęp

Początki elektronicznego obiegu oraz przetwarzania informacji wymagały od twórców tych systemów szerokiej wiedzy z zakresu elektroniki, fizyki oraz matematyki. Razem z rozwojem techniki zwiększało się także zapotrzebowanie na usługi informatyczne. Każda z firm w tym okresie miała swoje własne standardy przechowywania danych. Stosowano jednostki centralne, które były samowystarczalne. Wraz ze wzrostem informatyzacji rosła także liczba technologii firmowych (propieratary technologies), które nie były zgodne z innymi systemami informatycznymi w instytucjach współpracujących ze sobą. To rodziło wiele niedogodności oraz wymuszało znaczne nakłady inwestycyjne na integrację systemów. Te problemy zmusiły twórców oprogramowania do podjęcia prób standaryzacji protokołów wymiany informacji oraz sposobu ich przechowywania. Pierwotne standardy były niestety silnie powiązane z określoną technologią lub systemem. Taka zależność powoli ulegała zanikowi i aktualnie trendy w informatyce dążą do jak njefektywniejszego wykorzystywania architektury zorientowanej na usługi (SOA) [1]. Pozwala ona na wykorzystanie już gotowych wzorców oraz komunikacji pomiędzy systemami wykorzystując ujednolicone protokoły czy struktury.

Dzisiejsze technologie informatyczne znacznie zmieniły sposób realizacji naszych potrzeb. Podczas pisania programów za pomocą nowoczesnych narzędzi programistycznych, możemy poświęcić czas wyłącznie na urzeczywistnianie naszych pomysłów. Także proces wdrożenia aplikacji nie wymaga już szerokiej wiedzy na temat systemów – wystarczy tylko skopiować aplikację i uruchomić (XCOPY Deploy). Pisanie aplikacji dwuwarstwowych jako tzw. gruby klient (rich client) stało się nieopłacalne. Koszty związane z pielęgnowaniem instalacji na maszynach klienckich są często wyższe niż samo pozyskanie programu. Aplikacje oparte o interfejs wykorzystujący powszechnie znany standard HTTP (HyperText Transfer Protocol) oraz stosowanie rozwiązań z tzw. cienkim klientem (thin client) są przyszłością. Także wymiana danych za pomocą warstwy tego protokołu z innymi systemami nie sprawia już kłopotu. Komunikacja odbywa się poprzez rozproszone usługi WWW: WebServices, a dane przekazywane są za pomocą XML [2] (eXtensible Markup Language). Należy jednak podkreślić, że na rynku znajduje się wiele rodzajów przeglądarek WWW, które inaczej interpretują kod strony HTML czy CSS, co ma znaczący wpływ na proces opracowywania interfejsu i dostosowania do konkretnych przeglądarek. Technologia ASP.NET umożliwia tworzenie jednolitego kodu, który jest automatycznie dostosowywany do typu przeglądarki WWW. Ta cecha jest jednym z ważniejszych udogodnień dla programisty i pozwala na znaczne skrócenie czasu opracowywania aplikacji. W ASP.NET 2.0 wiele powtarzalnych zadań programistycznych zastąpiono rozwiązaniami wbudowanymi –uaktualniono i dodano około ponad 50 nowych kontrolek [3]. Kontrolki te wspierają między innymi dostęp do danych, zarządzanie użytkownikami , nawigację po serwisie WWW, wizulizację w postaci drzewa (TreeView) oraz wspomagające personalizację. Microsoft promuje tę wersję jako środowisko, w którym można zmniejszyć ilość kodu potrzebnego do wygenerowania takiej samej aplikacji w porównaniu do wersji wcześniejszej o około 70% ! Także inne podejście do samego rozplanowania serwisu WWW przyczynia się do skrócenia pisania aplikacji – przykładem może tu być zastosowanie w projektowaniu tzw. strony nadrzędnej (master page) [4], która stanowi jednocześnie szablon i kontener dla stron właściwych – podrzędnych. Pomimo tak bogatego zakresu środków do automatyzacji procesu tworzenia aplikacji podkreślić należy, że opracowanie dobrze funkcjonującej aplikacji wymaga zawsze od programisty dogłębnej wiedzy o działaniu stosowanych mechanizmów, a nie jedynie ograniczenia się do mechanicznego korzystania z kreatorów lub innych ułatwień formalnych.

2 Metody budowy aplikacji z zastosowaniem ASP.NET 2.0

Dostęp do danych w ASP.NET 2.0, w porównaniu z wersją poprzednią został całkowicie zmieniony [5], do tego stopnia że możemy nie pisać żadnego kodu. W zamian za to korzystamy z kontrolek typu (Data-Source), które będą reprezentować dane. Następnie związane z nimi dane zostaną przypisane do kontrolek warstwy prezentacyjnej (Data-Bound) np: tabela, widok szczegółowy czy formularz. Data Source - ASP.NET 2.0 wprowadza deklarowane kontrolki dostępu do danych, które mogą operować na danych pochodzących z różnych typów źródeł np.: pliki XML, baza danych SQL czy warstwa pośrednia logiki biznesowej (middle-tier business object). Kontrolki te wspierają takie operacje na danych jak sortowanie, stronicowanie, filtrowanie, aktualizacje, tworzenie nowych rekordów lub ich usuwanie. Pozwalają również, co istotne dla tempa budowy alikacji, na pominięcie czasochłonnego procesu opracowywania kodu na rzecz ustawienia ich właściwości.

```
<asp:ObjectDataSource ID="ODS_Tabelka" Runat="Server"
        TypeName="SunRise.Wniosek" SelectMethod="przetrzymywane_wnioski" >
        <SelectParameters>
                <asp:ControlParameter Name="id_proces" Type="int32"
                        ControlID="id_proces" PropertyName="Value" />
        </SelectParameters>
</asp:ObjectDataSource>
```

Tabela. 1. Tabela 1. Metody prezentacji danych za pomocą kontrolek ASP.net 2.0

Nazwa kontrolki	Opis
GridView	Wyświetla dane w formacie tabeli – jest następcą kontrolki o nazwie DataGrid.
DetailsView	Wyświetla pojedynczy rekord z zestawu – wykazuje duże podobieństwo funkcjonalne z formularzem z MS Access.
FormView	Wyświetla pojedynczy rekord z zestawu oraz umożliwia sformatowanie go według odpowiedniego szablonu np.: tryb do-odczytu, nowy rekord, aktualizacja itp. Wykazuje duże podobieństwo funkcjonalne z formularzem z MS Access.
TreeView	Wyświetla hierarchiczne zestawy danych z możliwością tymczasowego ukrywania podrzędnych węzłów struktury.
Menu	Wyświetla dynamiczne menu kontekstowe strony – także wielokrotnie rozwijalne.

```
<asp:GridView Runat="Server" ID="GV_Tabelka"
        DataSourceID="ODS_Tabelka" DataKeyNames="id_proces"
        AllowPaging=True AllowSorting=True AutoGenerateColumns=True
        PagerSettings-Mode="NextPrevious" PageSize=5 />
```

Nowe kontrolki pozwalające na prezentacje danych [5] (Data Bound) – takie jak tabela (GridView), widok szczegółowy (DetailsView), formularz (FormView), drzewo (TreeView) oraz menu mogą być w szerokim zakresie sparametryzowane. Pozwala to na wyświetlenie danych na wiele różnych sposobów za pomocą tego samego obiektu. Kontrolki te ponadto wyposażone są w mechanizm automatycznego pobierania i odświeżania danych źródłowych z warstwy danych (Data Source). Zapewnia to dyrektywa DataSourceID w opcjach kontrolek prezentacyjnych. Oczywiście wspierają również podstawowe operacje na danych takie jak sortowanie, stronicowanie

Strony nadrzędne (Master Pages) [6] udostępniają wspólną strukturę i elementy interfejsu użytkownika, na serwisie WWW, takie jak nagłówek strony, stopka strony lub pasek menu nawigacyjne w jednym i spójnym obiekcie. Master Pages udostępniają w/w elementy innym stronom podrzędnym. Taka architektura aplikacji znacznie redukuje redundancje kodu oraz co za tym idzie – jest bardziej przejrzysta oraz szybsza w implementacji. Definiowanie struktury szablonu serwisu jest podobne do tworzenia normalnej strony. Strona nadrzędna może posiadać kontrolki, kod lub dowolną kombinację tych elementów oraz dodatkową, specjalną kontrolkę nazywaną ContentPlaceHolder.

Kontrolka ta definiuje region, w którym strony podrzędne będą umieszczały treść lub wyświetlały inne kontrolki, które są przypisane do strony podrzędnej. Kontrolka ContentPlaceHolder może posiadać także domyślną zawartość. W samej stronie nadrzędnej można umieszczać także kod, który będzie dostępny z poziomu stron podrzędnych. Przykładem może być tutaj odwołanie się do strony nadrzędnej, aby ta zwróciła stronie pytającej informację o zalogowanym użytkowniku. Także strona podrzędna może zmienić właściwość stronie nadrzędnej – np. w przypadku, gdy chcemy zmienić tytuł nagłówka strony nadrzędnej z poziomu strony podrzędnej. Wracając jednak do opisu wykorzystania strony nadrzędnej w pozostałych stronach chciałbym przytoczyć przykład wykorzystania tej nowej cechy. W naszym przykładzie jest ona pusta i nie zawiera żadnych treści, ani modułów funkcjonalnych. Na jej przykładzie będzie można pokazać jak wygląda szablon strony właściwej, która jest osadzona w stronie nadrzędnej (SunRise.Master).

Warto dodać, że ASP.NET 2.0 w sposób rewolucyjny podchodzi do personalizacji [6]. Dodano tutaj chyba najwięcej ulepszeń i nowości. Przykładem może być obsługa tematów (Themes), czy lokalizacji językowych (pliki *.resx), dzięki którym możemy w prosty sposób zróżnicować końcowy wygląd strony dla dowolnego użytkownika. Dopełnieniem tej funkcjonalności jest kontrolka WebParts [6], dzięki której użytkownik może sam, za pomocą technologii przeciągnij-i-upuść personalizować serwis WWW. WebPartManager decyduje w jaki sposób Web Parts mogą być przemieszczane na stronie. Zarządzanie użytkownikami zostało wbudowane w sam ASP.NET 2.0, dzięki kontrolkom odpowiadającym za zakładanie konta, logowanie, przypominanie hasła czy zarządzanie rolami. Dzięki nim możemy, nie pisząc kodu, w bardzo szybki sposób wykonać menu do poruszania się w serwisie. W pliku web.sitemap (domyślny) określamy strukturę logiczną serwisu oraz nadajemy prawa do określonych węzłów czy stron. Bardzo istotnym elementem jest również uproszczenie procesu wdrożenia. Ogranicza się ono do skopiowania plików do wcześniej przygotowanego katalogu domowego na serwerze IIS. Wystarczają do tego standardowe narzędzia do zarządzania plikami takie jak xcopy, czy eksplorator Windows. Proces kopiowania jest równoznaczny z procesem zainstalowania aplikacji na serwerze docelowym. Aplikacja przy pierwszym uruchomieniu (przy pierwszym wywołaniu strony przez użytkownika) sama się kompiluje (kod SunRise.vb w katalogu /app_code) oraz sama podłącza bazę danych do źródła danych Servera SQL 2005 Express [7]. Jeżeli odpowiednie podłączenie do bazy danych już istnieje, to nie jest wykonywana żadna akcja. Ścieżka do pliku jest automatycznie pobierana w przypadku, gdy zawrzemy w ciągu połączeniowym (connectionstring) klauzulę |DataDirectory|, która odpowiada ścieżce relatywnej do wskazanego pliku *.mdf. Warto wspomnieć o tym, że dla każdego przypadku dynamicznego podłączenia bazy danych w SQL 2005 podłączane są także pliki logów (*.log). Jeżeli taki plik nie istnieje, jest automatycznie tworzony. W przypadku wygaśnięcia wszystkich połączeń do bazy danych, jest ona automatycznie odłączana (AutoClose).

3 Rozwiązania zastosowane przy budowie aplikacji

Baza danych w aplikacji System Obiegu Wniosków jest jej najważniejszą częścią. To w niej przetrzymywane są specyfikacje wniosków i ich zawartość. Wszelkie modyfikacje w rdzeniu danych są dokonywane za pośrednictwem procedur składowanych oraz warstwy pośredniej [5]. Niektóre operacje na tabelach wniosków bazują na wbudowanej w .NET obsłudze bezpołączeniowej (DataSet), która umożliwia zarządzanie danymi i zabezpiecza przed bezpośrednią ingerencją w dane. Taka architektura zabezpiecza aplikację przed możliwością eskalacji praw, zapewnia poufność danych oraz zabezpiecza przed ich zniszczeniem.

W Systemie Obiegu Wniosków również połączenie do bazy danych odbywa się przez procedury składowane, które są wyposażane w interfejs na pośredniej warstwie aplikacji. Strony ASP.NET 2.0 korzystają z tych interfejsów przy wykonywaniu operacji na danych. Baza danych składa się z niezmiennego rdzenia oraz dynamicznie tworzonej struktury. Struktura niezmiennego rdzenia jest podzielona na kilka części logicznych. Część obiektów jest przypisana do przechowywania i operacji na danych związanych m.in. z obsługą organizacji (pracownicy, departamenty, profile pracowników) czy definiowania wniosków wraz z ich przepływem. Natomiast zmienna część struktury bazy danych jest generowana dynamicznie, na podstawie definicji wniosku. Tworzona jest wtedy grupa obiektów, które przechowują dane zawarte w konkretnych wnioskach. Nazwa dynamicznej tabeli wniosków tworzona jest zakończona numerem identyfikującym proces, czyli prefiksem będzie zawsze *wniosek_tbl_proces_* a sufiksem jego *id* w bazie danych definicji procesów i wniosków. Tabela ta powstaje w momencie publikacji, gdy w warstwie pośredniej wywołuje się metodę odpowiadającą za jej utworzenie w bazie danych. Podczas tej akcji generowana jest nie tylko struktura tabeli wniosków ale także tabela dodatkowa, która ma za zadanie przechowywanie historii operacji na danym wniosku *wniosek_tbl_proces_X_trace* oraz wyzwalacz (trigger), który będzie zapisywał historię wniosku. Poniżej przedstawię zarys procesu generowania tabeli.

Na początku deklarujemy nazwę tabeli oraz stałe arbitralnie ustalone pole, które będzie kluczem głównym tabeli.

```
sql = " CREATE TABLE [dbo].[wniosek_tbl_proces_" & CStr(id_proces) _
          & "] (" & vbCrLf & _
          " [id_wniosek] [int] IDENTITY (1, 1) NOT NULL ," & vbCrLf
```

Następnie dodajemy dynamiczne kolumny z bazy danych. Poniższy kod pokazuje w jaki sposób tworzona jest tabela przechowująca wniosek. Widać, że tworzony kod SQL bazuje na definicji pól przyszłej tabeli, które są odczytywane z funkcji obieg.pokaz_pola_w_procesie:

```
Dim pokaz_pola As SqlDataReader = Obieg.pokaz_pola_w_procesie(id_proces)
While pokaz_pola.read()
Sql = Sql & "    [" & pokaz_pola("nazwa_pola_sql").ToString & _ "] " &
pokaz_pola("sql_typ").ToString & " NULL ," & vbCrLf
  If Not IsDBNull(pokaz_pola("sql_default")) Then
          constraints = constraints & "CONSTRAINT [wniosek_tbl_proces_" _
& CStr(id_proces) & "_" & _ pokaz_pola("nazwa_pola_sql").ToString _
&  "] DEFAULT (" & pokaz_pola("sql_default").ToString & _
") FOR [" _
& pokaz_pola("nazwa_pola_sql").ToString & "]," & vbCrLf
  End If
End While
```

W przypadku, gdy w definicji pola znajduje się wartość domyślna jest ona dodawana do struktury tabelki jako ograniczenie (constraints). Następnie dodawane są kolumny domyślne:

```
sql = sql & "    [id_status] [int] NOT NULL, " & vbCrLf
sql = sql & "    [symbol_sprawy] [char] (12) NOT NULL , " & vbCrLf
sql = sql & "    [kto] [int] NOT NULL , " & vbCrLf
```

```
sql = sql & "    [kiedy] [datetime] NOT NULL " & vbCrLf
sql = sql & "  ) ON [PRIMARY] " & vbCrLf & vbCrLf
```

oraz ustanawiany jest klucz główny na polu id_wniosek:

```
        sql = sql & " ALTER TABLE [dbo].[wniosek_tbl_proces_" & _ CStr(id_proces) _
         & "] WITH NOCHECK ADD  CONSTRAINT [PK_wniosek_tbl_proces_" _
         & CStr(id_proces) & "] PRIMARY KEY  CLUSTERED ( [id_wniosek] )  ON
[PRIMARY]  "
```

W kolejnych etapach dodawane są ograniczenia kluczy obcych (foreign key) do tabel odpowiadających za przechowywanie informacji o statusach, pracownikach.

```
        constraints = " ALTER TABLE [dbo].[wniosek_tbl_proces_" & _ CStr(id_proces)
        & "] ADD " & vbCrLf & constraints & _
        " CONSTRAINT [FK_wniosek_tbl_proces_" _
        & CStr(id_proces) & "_obieg_tbl_status] FOREIGN KEY " & _ [id_status])
        REFERENCES " & _
        [dbo].[obieg_tbl_status] ([id_status])" & vbCrLf & vbCrLf

constraints = constraints & " ALTER TABLE [dbo].[wniosek_tbl_proces_"  _
          & CStr(id_proces) & "] ADD " & vbCrLf & " CONSTRAINT " & _
          " [FK_wniosek_tbl_proces_" _
          & CStr(id_proces) & "_organizacja_tbl_pracownicy] " & _
          " FOREIGN KEY ([kto]) REFERENCES" & _
           " [dbo].[organizacja_tbl_pracownicy] ([id_pracownik])"
```

Szerszy opis struktury bazy danych oraz warstwy pośredniej Systemu Obiegu Wniosków wraz z kodami źródłowymi jest dostępny w [8]. Na zakończenie, chciałbym przedstawić proces generowania formularza wniosku w zależności od jego definicji, aktualnego statusu oraz od profilu zalogowanego użytkownika. Jest on tworzony dynamicznie na podstawie danych zawartych w bazie danych, skąd poprzez procedurę składowaną jest zwracana struktura wniosku wraz z parametrami pól (do odczytu - O, edytowalne - E, wymagane - W). Rozpoczynamy od odczytania danych na temat pól dostępnych we wniosku oraz deklarujemy tabelę, w której będą umieszczane wypełniane pola.

```
Dim sqldtr As SqlDataReader
 sqldtr = wniosek.dostepne_pola_dla_pracownika_pokaz_liste(id_proces.Value, _
        wniosek.idstatus_dla_S(id_proces.Value), _
        Master.zalogowany_user)

While sqldtr.Read()
 Dim tr As New TableRow
 Dim tc As New TableCell
 Dim lbl As New Label (…)
```

Następnie sprawdzany jest format pola, które jest zadeklarowane w bazie danych jako wyrażenie regularne i przypisujemy je jako zagnieżdżone kontrolki w tabeli.

```
(…) If sqldtr("typ").ToString.ToUpper = "E"  Then

Select Case (sqldtr("html_typ").ToString.ToLower)
  Case "textbox"
    Dim txbx As New TextBox
       txbx.ID = sqldtr("nazwa_pola_sql").ToString
       txbx.Width = CInt(sqldtr("html_opcje_width"))
    If CInt(sqldtr("html_opcje_maxlength")) > 0 Then
       txbx.MaxLength = sqldtr("html_opcje_maxlength")
    End If
    If Not IsDBNull(sqldtr("html_opcje_regexp")) Then
      Dim RegExpVal As New RegularExpressionValidator
         RegExpVal.ControlToValidate = sqldtr("nazwa_pola_sql").ToString
         RegExpVal.ValidationExpression = _
         sqldtr("html_opcje_regexp").ToString
         RegExpVal.ErrorMessage = "<b>!</b>"
       tc.Controls.Add(RegExpVal)
    End If
(…)
  tc.Controls.Add(txbx)
End Select
(…)
 tr.Controls.Add(tc)
 tbl.Controls.Add(tr)
End Chile
```

Jak widać na powyższym przykładzie, wniosek posiada zestaw pól, które są sparametryzowane wg definicji przechowywanej w bazie danych. Zapis wypełnionego wniosku odbywa się dzięki odczytaniu struktury tabeli oraz stworzeniu nowego wiersza, do którego wpisywane są wartości.

4 Podsumowanie

Całość opracowania poświęconego aplikacji stanowiącej ilustracje omawianych w artykule metod, a które liczy ponad sto stron jest do pobrania z [8]. Zawartych jest tam szereg zrzutów ekranowych oraz diagramów, co pozwala na lepsze zrozumienie budowy Systemu Obiegu Wniosków. Pod tym adresem internetowym znajdują się także kody źródłowe aplikacji. Prezentowane oprogramowanie jest projektem o otwartych źródłach i jest dostępny na portalu SourceForge [9]. Zachęcam do rozwijania aplikacji oraz dzielenia się swoimi uwagami. Mam ogromną nadzieję, że choć w minimalnym stopniu zachęciłem

Państwa do urzeczywistniania własnych pomysłów za pomocą prezentowanych nowoczesnych narzędzi programistycznych.

Literatura

[1] http://en.wikipedia.org/wiki/Service-oriented_architecture

[2] Thangarathinam T.: Professional ASP.NET 2.0 XML, Jon Wiley & Sons, 2006

[3] Evjen B.: ASP.NET 2.0 Beta Preview, Jon Wiley & Sons, 2004.

[4] Evjen B., Hanselman S., Muhammad F., Srinivasa Sivakumar S., Rader D.: Professional ASP.NET 2, Jon Wiley & Sons, 2005.

[5] http://msdn.microsoft.com/asp.net/reference/data/default.aspx

[6] http://weblogs.asp.net/lkempe/archive/2004/07/18/186694.aspx

[7] Otey M.: Microsoft SQL Server 2005. *Nowe możliwości,* Helion, Gliwice 2005

[8] http://www.koluszki.neostrada.pl/janowski/sysobwn.zip

[9] http://sourceforge.net/projects/sysobwn

SYSTEM OF DOCUMENTS MANAGEMENT AS A EXAMPLE OF ASP.NET 2.0 AND MS SQL 2005 EXPRESS APPLICATION

Summary – The main idea of this paper concern new method of application creation, which guarantee high security level with low programming costs consumption. This time combination of ASP.NET 2.0 and SQL Server 2005 seems to be the most powerful and efficient available technology. Presentation of most important possibilities offer by this technologies is based on system of documents management. In spite that this application can be useful in practice and commerce is only the illustration of presented tools and methods.