

Tworzenie aplikacji web based w Pascalu

Dariusz Mazur

Jubileuszowy Zlot Programistów Delphi 2009
6 - 8 lutego 2009, Kraków, Akademia Górniczo-Hutnicza

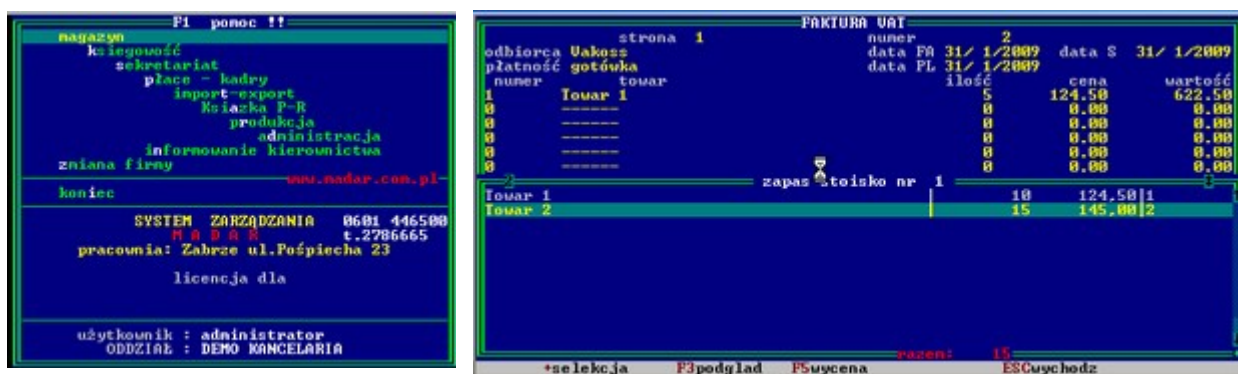
Spis treści

1.Przykład aplikacji wieloplatformowej.....	3
DOS	3
Windows	3
Linux	3
Mac	4
Web based.....	4
2.Cel i założenia projektu.....	4
Ten sam kod źródłowy	5
Architektura rozwiązania.....	5
Zgodność z architekturą i API bibliotek VCL:.....	6
Korzystanie z elementów standardowych	6
Styl programowania:.....	6
Rozszerzenia komponentów.....	6
3.Komunikacja z przeglądarką.....	7
Podobieństwa i różnice.....	7
Stosowane technologie.....	7
100% Ajax	7
Cookie.....	8
JavaScript	8
Html.....	8
CSS.....	8
Optymalizacja transferu.....	8
4.Przykładowy program.....	9
Modyfikacja programu desktopowego.	12
Źródło strony html.....	13
Tworzenie nowego komponentu.....	15
5.Optymalizacja.....	17
Optymalizacja transferu	17
Optymalizacja aplikacji serwerowej.....	17
Uzyskane wyniki	18
6.Do zrobienia:.....	18
7.Porównanie technologii.....	19
8.Wnioski:.....	20

1. Przykład aplikacji wieloplatformowej

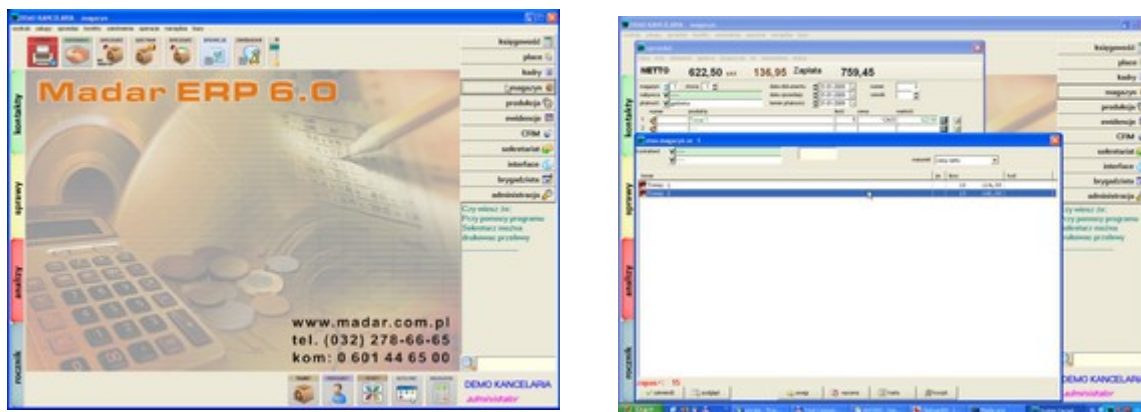
DOS

Najstarsza, o dużych ograniczeniach a mimo to jeszcze popularna platforma. Część użytkowników nie chce się przenieść na nowsze, ładniejsze, bardziej funkcjonalne. Zalety: obsługa w 100% klawiaturą, duże litery, prosty interfejs.



Windows

Najpopularniejszy system operacyjny na desktopach, wadą jest możliwość instalacji wielu programów firmowych, prywatnych, aktualizacji i wirusów, których skomplikowane interakcje powodują znaczące spowolnienie, niewłaściwe działanie lub całkowity brak możliwości użytkownika komputera.

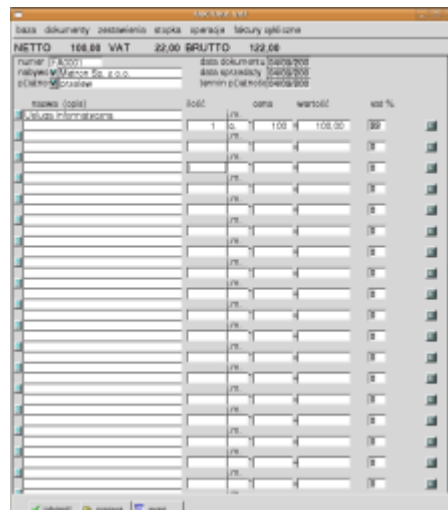


Mac

Popularny w USA, w Europie równie śladowy co Linux.

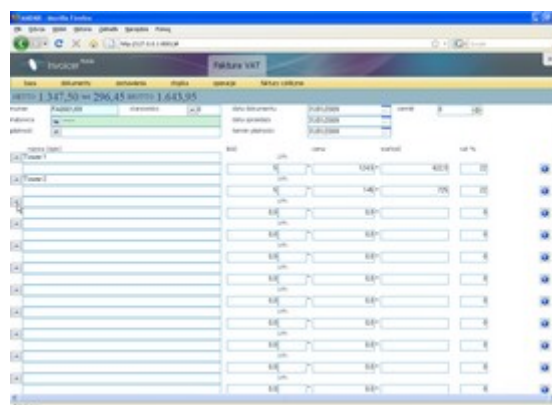
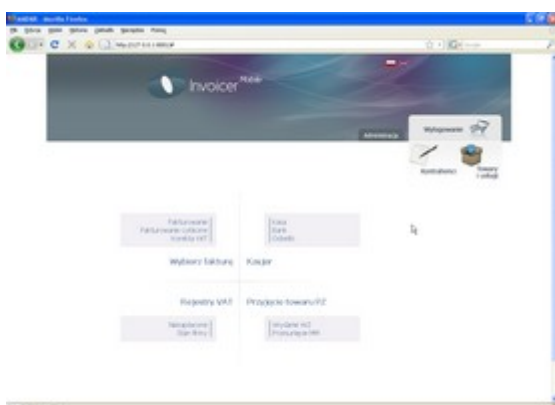
Linux

Duży potencjał, popularny system serwerowy, niszowy na desktopach, szczególnie firmowych. Duża zmienność pomiędzy dystrybucjami co znacząco utrudnia tworzenie oprogramowania.

A screenshot of a financial application window titled 'dokument VAT'. The window displays a summary of VAT data: NETTO 168.00, VAT 22.00, BRUTTO 190.00. Below this is a table with columns: 'nazwa (opis)', 'ilość', 'cena', 'wartość', and 'vat %'. The table contains one row of data: 'usługa informatyczna', '1', '5', '100', '100.00', and '20%'. The application interface includes a menu bar and a toolbar at the bottom.

Web based

Nowa platforma, bardzo popularna w niektórych zastosowaniach (mapy, wyszukiwarki, handel internetowy, portale społecznościowe, gry MMO, obsługa bankowości), brak ograniczeń na odległość, zredukowana konieczność instalacji, brak zależności pomiędzy aplikacjami. Powstają urządzenia predestynowane do obsługi internetu (UMPC, MID). Wady: duża nieufność ze względu na fakt, że przeznaczenie tej platformy nie było zastąpienie klasycznych aplikacji desktopowych, skomplikowane tworzenie niektórych komponentów wizualnych.



1. Cel i założenia projektu

Głównym celem tego projektu było stworzenie biblioteki umożliwiającej przeniesienie istniejących aplikacji do sfery web-based, czyli umożliwienie tworzenia aplikacji internetowych. Dodatkowymi wymaganiami były: maksymalne wykorzystanie istniejącego kodu, oparcie o standardy (w3c.org), minimalizacja różnic w wyglądzie i obsłudze w stosunku do tradycyjnych aplikacji. Nie do pogardzenia były by możliwość wykorzystania systemu operacyjnego Linux.

Ten sam kod źródłowy

Tworzenie aplikacji internetowych w dużej mierze oparte jest o języki skryptowe. Gwałtowny ich rozwój ukazał bogactwo form i możliwości, w dłuższym okresie czasu ujawniły się słabości tychże rozwiązań. O tyle są one istotne, że występują na poziomie podstawowym, fundamentalnym co powoduje, że ominięcie czy zignorowanie występujących problemów jest trudne bądź niemożliwe. Co więcej, problemy te narastają wraz ze wzrostem złożoności tworzonych programów, co nie jest dobrym prognostykiem. Podstawowe problemy to szybkość, słabość języka, przeplatanie struktur programu z kodem HTML, niestabilny rozwój (brak kompatybilności wstecz). Dla wielu związanych z rynkiem aplikacji desktopowych ideałem byłoby jedno narzędzie umożliwiające równoległe tworzenie aplikacji na wszystkie platformy, tzn. operujące na tym samym kodzie źródłowym.

Delphi nie jest świadomym, obiektywnym i niezależnym wyborem narzędzia. Jest wynikiem istniejących uwarunkowań, zastanej sytuacji. Decyzja została podjęta na zasadzie braku możliwości udowodnienia tezy o całkowitej nieprzydatności tego narzędzia do tworzenia aplikacji internetowych. Drogę do celu wyznaczyło szereg doświadczeń i przykładów, które łącznie z dokonanymi pracami implementacyjnymi pozwoliły na osiągnięcie sukcesu w realizacji ww celu.

I tak jako wzór aplikacji wieloplatformowych można przedstawić FPC. Po drugie implementacja bibliotek wizualnych zrealizowana w projekcie Lazarus jest przykładem jak można całkowicie różne interfejsy graficzne opakować w jedno spójne API. Co więcej, jest to dowód że takim najlepszym API jest to zawarte w VCL Delphi. Oczywiście nic by się nie udało gdyby nie istniały biblioteki komunikacji internetowej Indy i Synapse. Wszystko rozpoczęło się od przykładu serwera Elize z Indy, choć ta biblioteka nie jest już używana. Kolejnym krokiem była analiza bibliotek komponentów utworzonych w Javascript i wykorzystujących AJAX , takich jak Ext Js, Dojo itd. Okazało się, że wystarczy że aplikacja serwerowa wygeneruje tylko stosunkowo niewielkiej długości łańcuchy, a całą resztę dokona przeglądarka. Pod pewnymi względami stało się to prostsze niż rysowanie po ekranie w Windows. Ostatecznym dowodem była biblioteka Wt (do tworzenia aplikacji webowych w C++). Ona unaoczniała, że przedstawiony zamiar jest możliwy do realizacji.

Architektura rozwiązania

- serwer HTTP wbudowany w aplikację (biblioteka Synapse),
- obsługa sesji (każda sesja to jakby jedna instancja aplikacji dla jednego użytkownika),
- AJAX: każde działanie przekazywane do serwera, który każdorazowo generuje i przesyła zmiany do przeglądarki,
- praca wielowątkowa aplikacji:
 - wątek główny,
 - wątek przyjmowania zapytań i generowania odpowiedzi: dla każdego zapytania tworzony odrębny wątek,
 - wątek aplikacji: dla każdego użytkownika odrębny,
 - komunikacja pomiędzy wątkami zapytań a aplikacyjnym odbywa się przy pomocy kolejki FIFO (implementacja waitfree),
 - wątki pomocnicze: wątek aplikacyjny niektóre obliczenia dokonuje w trybie wielowątkowym i ta możliwość została zachowana,

Zgodność z architekturą i API bibliotek VCL:

- kolejki komunikatów analogicznej to tej z Windows (wndProc),
- lista okien (Forms),
- kontrolki (comctrls, stdctrls, menu, grid)
- okna modalnych (okna dialogowe) w 100% zgodnych z Delphi (blokada przyjmowania zdarzeń z innych okien, zatrzymanie programu) (Dialogs).

Zawartość plików

- ajax1.js – podstawowa obsługa wywołań XMLHttpRequest, komunikacja serwer-przeglądarka
- xlhttp – implementacja serwera HTTP, z obsługą sesji
- hthttp – implementacja serwera aplikacyjnego, komunikacja socket – instancja aplikacji
- xlapp – korzeń obiektów HTML
- xlcontrol, xlgraphics, xlmenu, xlform

Tworzenie aplikacji web based

Korzystanie z elementów standardowych

1. Tcontrol
2. tEdit
3. Tbutton
4. tFont
5. tBrush
6. tForm

Styl programowania:

1. Formatki zapisywane są w postaci kodu (pure pascal), notacja wywodzi się z biblioteki Turbo Power.
2. Unikanie „pisanie w” HTML .
3. Kompilacja warunkowa.
4. Zmienne globalne – nowe spojrzenie.

Rozszerzenia komponentów

- tFieldBox, tFieldDBL,
- okno dialogowe,
- wizualizacja progresji,
- TpageControl,
- tMenu,
- Grid: Komunikacja, buforowanie.

2. Komunikacja z przeglądarką

Podobieństwa i różnice

Przeglądarka pod pewnymi względami jest podobna w swej strukturze do środowiska windows. Występują w niej takie same kontrolki jak przykładowo input (tEdit), button, checkbox. Bardzo podobna jest również obsługa, w tym możliwość oprogramowania takich samych zdarzeń (onClick, onChange, onMouseMove). Natomiast istnieje szereg różnic i to na różnych poziomach, których niwelacja jest konieczna aby umożliwić efektywne tworzenie aplikacji internetowych w sposób analogiczny to rozwiązań windowsowych. Należy tu wymienić:

- automatyczny rendering wraz z brakiem możliwości rysowania piksel po pikselu, co znacznie utrudnia tworzenie zarówno bardziej skomplikowanych kontrolki jak również wprowadza utrudnienia przy rozmieszczeniu komponentów na ekranie
- płaskie pojedyncze okno, w odróżnieniu od „okienkowego” windowsa,
- duża rozpiętość rozdzielczości ekranu (palmtopy),
- praca z wieloma użytkownikami jednocześnie, bezpołączeniowy protokół (w Windows przy jednym komputerze jest zazwyczaj jeden użytkownik),
- możliwość wielokrotnego otrzymania tego samego zapytania,
- utrudniona komunikacja zwrotna (brak odczytu która kontrolka „ma focus”),
- większa zawodność łącza (niektóre polecenia mogą zostać zignorowane, bo nie „doszły” do przeglądarki.

W rozwiązaniu z pomocą przychodzą narzędzia, które umożliwiają dokonania wszelkich koniecznych „obejść” :

- XMLHttpRequest: najsłynniejsze, podstawa wszelkich aplikacji RIA,
- DOM: Document Object Model,
- wielowarstwowy rendering.

Stosowane technologie

100% Ajax

Całość odbywa się wyłącznie poprzez wywołania asynchroniczne, brak wywołań powodujących przeładowanie strony, dzięki czemu aplikacja staje się odporna na standardowe operacje przeglądarkowe typu reload czy back, które to mogą powodować znaczne niedogodności w użytkowaniu aplikacji.

Cookie

Do zapamiętania i identyfikowania sesji – zrezygnowano całkowicie z jakiegokolwiek zapamiętywania stanów pośrednich w przeglądarce. Wszystkie działania użytkownika (kliknięcie, naciśnięcie klawisza, ruch myszką) powodujące taką zmianę transmitowane są do serwera, który

dokonuje odpowiednich operacji i generuje zmiany na stronie. Jedynym wyjątkiem są naturalne zachowania niektórych komponentów np. menu.

JavaScript

Zapewnienie komunikacji pomiędzy serwerem a użytkownikiem: realizuje funkcje wymiany strony, wymiany pojedynczych elementów strony, wyświetlenie okien dialogowych, obsługa menu i wyboru daty.

Html

- implementacja formularzy przy pomocy tabelki: skalowanie, wymiarowanie, duża zgodność pomiędzy przeglądarkami,
- pominięcie elementu href (linki) ze względu na zatrzymywanie się tabulatora na tych elementach,
- obsługa onClick, onChange, onKeyUp,
- adres strony = adres obiektu w pamięci serwera: unikalność, jednoznaczność, automatyczne mapowanie.

CSS

- odpowiada za wizualizację: wymiary, kolory, czcionki, bitmapy,
- odrębny CSS na palmtopy, GSM, komputery stacjonarne.

Optymalizacja transferu

- prawidłowe ustawienia cache przeglądarki,
- statyczne obrazki, skrypty i css,
- zmienione kontrolki,
- zmienione wartości kontrolek,
- reload całości okna,
- możliwość pakowania zip.

Uzyskane wyniki:

czas generowania odpowiedzi	0.2-15.0 ms
czas podawany w Firebug dla sieci lokalnej	2..30 ms
wielkość ramki	2..50 kB

3. Przykładowy program

```
{ $I start.inc }  
{ $DEFINE XHTML }
```

Dariusz Mazur: Tworzenie aplikacji web based w Pascalu Jubileuszowy Zlot Programistów Delphi 2009

```
unit htunittest;  
interface  
uses  
  sysutils,  
  wpstring,  
  bxmlkom,  
  xlcmd,  
  xlfield,  
  xlapp,  
  xlmenu,  
  xlekran,  
  xlstdctrls,  
  htapp,  
  xlcontrols,  
  xlcomctrls,  
  xlgraphics,  
  classes;
```

Unity o nazwa rozpoczynających się od XL to są webowe odpowiedniki bibliotek VCL.

```
type  
  tTestForm= class(tEntry)  
    public  
      fName,  
      fLogin,  
      fPass2,  
      fEmail,  
      fPass : string;  
      fHaslo : integer;
```

*tEntry jest podstawową klasą dziedziczącą po tForm i obsługującą formularze.
Zdefiniowano w niej szereg metod upraszczających dynamiczne definiowanie wyglądu*

```
    constructor create(aOwner : TObjectXML;aParam:integer=0);override;  
    procedure initEkran;//tworzenie okna  
    procedure plprocess(aKomenda : word);override;/obsługa podstawowych zdarzeń  
    procedure zapiszHaslo;  
    class function nazwaStrony:string;override;  
  end;
```

```
implementation
```

```
{ $R demotest.res }
```

```
constructor tTestForm.create;  
begin  
  inherited create(aOwner,aParam);  
  InitCustom(27, 6, 97, 15, 0, 0);
```

*współrzędne i wielkość okna podawane są w jednostkach odpowiadających wielkości znaku.
W zależności od przeznaczenia biblioteka dokonuje odpowiednich przeliczeń na piksele.
W trybie webowym jeden znak ma szerokość 1% (przyjmuje się 100 znaków w oknie,
wielkość ustalana proporcjonalnie).*

```
  fHaslo:=0;
```

Dariusz Mazur: Tworzenie aplikacji web based w Pascalu Jubileuszowy Zlot Programistów Delphi 2009

```
initEkran;  
  
end;  
procedure tTestForm.initEkran;  
var  
    te : tFieldString;  
    ssText : ansiString;  
begin  
    caption:='test';  
    te:=AddSimpleStringField('login',3,8,'X',3,22,14,10,1,fLog  
in);
```

Typowa funkcja dodająca do okna pole tekstowe wraz z etykietą. Parametry oznaczają współrzędne etykiety i pola, oraz jego szerokość. Treść pola jest ściśle związana ze zmienną fLogin, tzn. Jest przepisywana z i do tej zmiennej każdorazowo przed i po każdej zmianie pola.

```
te.FLabel.Font.Style:=[fsBold];
```

Ustalenie dodatkowych atrybutów etykiety.

```
te:=AddSimpleStringField('nazwa',4,8,'X',4,22,14,10,1,fName);  
te.FLabel.Font.Style:=[fsBold];  
te:=AddSimpleStringField('haslo',6,8,'X',6,22,14,10,1,fPass);  
te.FLabel.Font.Style:=[fsBold];  
te.passwordChar:='*';  
te:=AddSimpleStringField('powtorz',7,8,'X',7,22,14,10,1,fPass2);  
te.FLabel.Font.Style:=[fsBold];  
te.passwordChar:='*';  
esFieldOptionsOn(efExitAtEdges);  
panelPrawy(250);
```

Funkcja umożliwiająca podział okna na kolumny i zdefiniowanie w drugiej szpalcie nowych kontrolek. Funkcja umożliwi uzyskanie efektów analogicznych to powszechnie stosowanych na stronach internetowych.

```
ssText:='Jeżeli podczas <span class="mdQwyr">pierwszego logowania</span> do  
Invoicer Mobile zostało założone konto administratora lub w programie <span  
class="mdQwyr">Invoicer&nbsp;Madar</span>;  
ssText:=ssText+' został założony użytkownik to wystarczy podać nazwy odpowiednio w  
polach: <span class="mdQpole">użytkownik</span> i <span class="mdQpole">hasło</span>.';  
ssText:=ssText+'<br><br>UWAGA! <br><br>Użytkownik założony w programie <span  
class="mdQwyr">Invoicer Madar</span> musi mieć zaznaczone pole <span class="mdQpole">  
dostęp zdalny</span>.<br>';
```

```
addFrame('pierwsze uruchomienie',ssText);
```

Wyświetlenie ramki z tekstem html w ramce z zaokrąglonymi rogami.

```
addKlawisz(ccDone, '#check zatwierdz','');  
addKlawisz(ccQuit, '#erase porzuc','');
```

dodanie u dołu okna dwu klawiszy, które po naciśnięciu wygenerują zdarzenia o wskazanym identyfikatorze, obsługiwane przez metodę plprocess.

```
end;
```

Dariusz Mazur: Tworzenie aplikacji web based w Pascalu Jubileuszowy Zlot Programistów Delphi 2009

```
procedure tTestForm.plProcess;
begin
  case aKomenda of
    ccDone : zapiszHaslo;

    else inherited;
  end;
end;
procedure tTestForm.zapiszHaslo;
begin
```

Procedura wywoływana po naciśnięciu klawisza ZATWIERDZ. Sprawdza czy ustawiono zmienną fPass i czy jest ona równa zmiennej fPass2.

W przypadku niepowodzenia wyświetlany jest komunikat.

```
  if (fPass='') then begin
    komunikat('brak hasła');
    exit;
  end;
  if (fPass<>fPass2) then begin
    komunikat('błąd w hasle');
    exit;
  end;
```

```
end;
```

```
class function tTestForm.nazwaStrony;
begin
  result:='testform'+rHtml;
```

zdefiniowanie nazwy webowej okna

```
end;
```

```
begin
  {$IFDEF XHTML}
  tTestForm.RegisterFormat;
```

Rejestracja klasy w zbiorze okien webowych wywoływanych na podstawie nazwy nazwaStrony.

```
  {$ENDIF}
```

```
end.
```

Modyfikacja programu desktopowego.

Powyższy program wymaga kilku drobnych modyfikacji aby można było skompilować i uruchomić go w wersji desktopowej. W zasadzie jedynym wymaganiem jest zmiana listy unitów biorących udział w kompilacji. W tabelce przykład dwóch plików będących unitami w programie. Plik htbkremane.pas, jak sugeruje prefiks HT jest wrapem plików bkremane.pas na rzecz aplikacji webowej. Instrukcje warunkowe umożliwiają manipulowanie listą załączonych unitów. Sama treść modułu jest w obu przypadkach identyczna (dzięki komendzie INCLUDE na końcu

htbkremane.pas). Dzięki różnicy w nazwie można dokonywać kompilacji obu wersji, webowej i desktopowej, w tym samym katalogu.

Plik bkremane.pas	Plik htbkremane.pas
<pre>{ \$IFDEF XHTML} { %main htbkremane} { \$ELSE} { \$I start.inc} unit bkremane; interface uses { \$I usesin.pas} bazaGR, bazadruk, bazaDr2, bazaTXT, bazadrsu, skTowarA, skTowar, sknew, skKontra, skTowarB, skKontrB, skopisy, bazaLS, bkMagaz, fkTrans, skfile, bkMagazA, fktransa, fkkonblo, bkmemo, windows, antrans, setup, ffkoszyk, wphash, fkDane; { \$ENDIF}</pre>	<pre>{ \$I start.inc} { \$DEFINE XHTML} unit htbkremane; interface uses { \$I usxlin.pas} bxmltxt, bxmls, bxmlgr, xlfile, bxmldruk, htbkmagaza, fkDane, htfktransa, htfktrans, htfkkonblo, httowara, htkontra, httowar, htbkmagaz, htbkmemo, windows, htskopisy, htantrans, htsetup, htkoszyk, wphash; { \$i bkremane.pas}</pre>

Źródło strony html

W strukturze wygenerowanego tekstu można wyróżnić następujące elementy:

1. HEAD – zdefiniowanie języka dokumentu, stosowanych stylów oraz dołączonych plików ze skryptami.
2. Nagłówek strony (pasek_tytul), czyli pierwsza belka rysowanego okna. Zawiera on tytuł okna oraz przycisk do jego natychmiastowego zamknięcia.

3. Belka menu. W tym przypadku pusty, natomiast w rzeczywistości ma postać zagnieżdżonej listy.
4. Ponieważ przykład używa komendy PANELPRAWY - podział okna na dwie części.
5. Główna tabelka formatki - tablica do której komórek wpisywane są poszczególne kontrolki.
6. Wykorzystując raster tabelki, tworzone są DIV'y zawierające poszczególne kontrolki. Każdy div ma unikalny identyfikator, dzięki czemu można zmieniać jego zawartość wywołaniami AJAX.
7. Pola tekstowe mają podpięte zdarzenia onchange="return chk('a009CFC24','f12');", jest to funkcja przekazująca informację do serwera o fakcie zmiany pola tekstowego.
8. Przyciski skonstruowane są przy pomocy DIV, których wygląd zdefiniowany jest w CSS oraz podpięte jest zdarzenie onclick
9. Na końcu kodu dołączone są puste divy. Służą one jako haki, do dynamicznego generowania niektórych elementów (dhtmlwindowholder do wyskakujących okien, dropmenudiv : tzw.popup menu) oraz przekazania parametrów do stacji użytkownika (idadres zawiera systemowy adres okna, identyfikujący go na serwerze).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pl" lang="pl">
<head>
<title>MADAR</title>
<script type="text/javascript" src="ajax1.js"></script>
<script type="text/javascript" src="dhtmlwindow.js"></script>
<script type="text/javascript" src="ddmenu.js"></script>
<meta name="Subject" content="Mobilna praca, zdalna praca, system mobilny">
<meta name="robots" content="index,nofollow">
<meta name="Description" content="Mobilna praca, zdalna praca, system mobilny">
<meta name="Keywords" content="system mobilny, system mobilny erp, madar mobile, mobilny
handlowiec, zdalna praca, madar erp przez komórkę, madar erp w palmtopie, komórka,
palmtop, laptop, program dla firm, praca w terenie">

<link rel="stylesheet" href="pc.css" type="text/css" >
<link rel="stylesheet" href="wyglad.css" type="text/css" >
<link rel="stylesheet" href="dhtmlwindow.css" type="text/css"></link>

</head>

<body onkeydown="return wykonajKey('a009CFC24',event);" oncontextmenu="return false;">
<table id="tfa009CFC24" class="pasek_tytul">
<tr>
<td><center>
<table>
<tr>
<td class="mdzakleft">
<div></div></td>
<td class="mdzakmid">
<div id="tytul"><h1>test</h1></div></td>
<td class="mdzakright">
</td>
</tr>
</table>
</td>
</tr>
</table>
</body>
</html>
```

Dariusz Mazur: Tworzenie aplikacji web based w Pascalu
Jubileuszowy Zlot Programistów Delphi 2009

```
        </table>
    </center>
</td>
<td>
    <div id="powrot">
        <a href="#" onClick="wlk('a009CFC24.html?procedura=L;komenda=22;pozycja=0');">
            </a>
        </div>
    </td>
</tr>
</table>
<table class="tablebar">
<tr>
<td class="midlebar">
    <table class="tableform" width="100%" >
        <col width="9%" xleft="8">
        <col width="19%" xleft="22">
        <col width="19%" xleft="36">
        <col width="53%" >
    <tr>
    </tr>
    <tr><td colspan="10">&nbsp;</td></tr>
    <tr>
    <td></td>
    <td><div id="dfa009D0884" class="fieldlabel">
        <label for="fa009D0BE8">login</label></div></td>
    <td><div id="dfa009D0BE8" class="fieldstring" >
        <input id="fa009D0BE8" type="text" maxlength="10" onChange="return
            chk('a009CFC24','fa009D0BE8');" value=""></div></td>
    <td class="spanstop" colspan="1" xcolstop="1"></td>
</tr><tr>
    <td></td>
    <td><div id="dfa009D1428" class="fieldlabel">
        <label for="fa009D1748">nazwa</label></div></td>
    <td><div id="dfa009D1748" class="fieldstring">
        <input id="fa009D1748" type="text" maxlength="10" onChange="return
            chk('a009CFC24','fa009D1748');" value=""></div></td>
    <td class="spanstop" colspan="1" xcolstop="1"></td>
</tr><tr>
    <td colspan="10">&nbsp;</td>
</tr><tr>
    <td></td>
    <td><div id="dfa009D1FC0" class="fieldlabel" >
        <label for="f12">haslo</label></div></td>
    <td><div id="df12" class="fieldstring" >
        <input id="f12" type="password" type="text" maxlength="10" onChange="return
            chk('a009CFC24','f12');" value=""></div></td>
    <td class="spanstop" colspan="1" xcolstop="1"></td>
</tr><tr>
    <td></td>
    <td><div id="dfa009D2B40" class="fieldlabel" >
        <label for="f13">powtorz</label></div></td>
    <td><div id="df13" class="fieldstring" ><input id="f13" type="password" type="text"
        maxlength="10" onChange="return chk('a009CFC24','f13');" value=""></div></td>
    </tr></table>
</td>
<td class="rightbar" style="width:250px;" >
<div id="dfa009D3C68"><div class="fieldframe"><div class="t1"><div class="tr"><div
class="b1"><div class="br">
<h2>pierwsze uruchomienie</h2>
<div class="box2"><div class="t12"><div class="tr2"><div class="b12"><div
```

```
class="br2"><div class="content">
Jeżeli podczas <span class="mdQwyr">pierwszego logowania</span> do Invoicer Mobile
zostało założone konto administratora lub w programie <span
class="mdQwyr">Invoicer&nbsp;Madar</span> został założony użytkownik to wystarczy podać
nazwy odpowiednio w polach: <span class="mdQpole">użytkownik</span> i <span
class="mdQpole">hasło</span>.<br><br>UWAGA! <br><br>Użytkownik założony w programie <span
class="mdQwyr">Invoicer Madar</span> musi mieć zaznaczone pole <span class="mdQpole">
dostęp zdalny</span>.<br>
</div></div></div></div></div></div></div></div></div></div></div></div>
</td>
</tr>
</table>
<table >
<tr>
<td ><div id="dfa009B7BFC" class="klawisz" onMouseOver="this.className='klawisz-
hover'" onMouseOut="this.className='klawisz'" onClick="return
wlk('a009CFC24.html?procedura=L;komenda=311;control=a009B7BFC');" >
<span>zatwierdz</span></div></td>
<td ><div id="dfa009BF00C" class="klawisz" onMouseOver="this.className='klawisz-
hover'" onMouseOut="this.className='klawisz'" onClick="return
wlk('a009CFC24.html?procedura=L;komenda=311;control=a009BF00C');" >
<span>porzuc</span></div></td>
</tr>
</table>

<div id="dropmenudiv" style="visibility:hidden;" onMouseover="clearhidemenu()"
onMouseout="dynamichide(event)" >
</div>
<div id="idadres" style="display:none">a009CFC24</div>
<div id="dhtmlwindowholder"><span style="display:none">.</span></div>
</body>
</html>
```

Tworzenie nowego komponentu

Implementacja nowego komponentu polega na zdefiniowaniu kilku metod dla nowej klasy dziedziczącej po `TField`, czyli abstrakcyjnym komponentcie. Najważniejszą metodą jest `InnerLink`, jest ona odpowiedzialna za wygenerowanie odpowiedniego kodu HTML, który zostanie wstawiony do właściwego DIV. DIV zagwarantuje przedzielenie odpowiedniego miejsca w oknie, w odpowiednim miejscu i o odpowiednich wymiarach. W tym przypadku chcemy utworzyć pole `INPUT` służące do wprowadzania liczb, dlatego też jako jeden z atrybutów wstawiana jest maska. Kolejnym elementem jest dołączenie funkcji zdarzeń. Jest to realizowane poprzez funkcję `AjaxChange`, która zwaca kod dla przeglądarki, podpinający odpowiednią funkcję. W tym przypadku jest to `onChange`.

Druga metoda to `ValueHTML`. Jej zadaniem jest rozróżnienie stanu zmian komponentu, a tym samym wysyłanie jedynie koniecznych informacji. Metoda sprawdza czy od ostatniego wysłania nastąpiła zmiana wartości komórki i w tym przypadku przesyłany jest jedynie ciąg pola `VALUE`, w przeciwnym komponent jest pomijany (komponent jest poprawny więc przesyłanie informacji o jego stanie jest niepotrzebne). Jeżeli funkcja `ValueHTML` nie jest zdefiniowana lub nastąpiły istotne zmiany komponentu (np. kolor, status włączenia) framework automatycznie wywołuje `innerLink`.

Ostatnia funkcja `postField` jest wywoływana na rzecz zapytania z przeglądarki, wygenerowanego

po zmianie danego pola. Jako parametr otrzymana jest nowa zawartość komponentu. Dokonywana jest modyfikacja i wywoływana jest procedura ogólna formy esPostEdit.

```
tFieldDoubleXML = class(tField)
public
  oldVal : double;
  function innerlink:ansiString;override;
  procedure postField(aPart:integer;const aWartosc : ansistring);override;
  function valueHTML(var astr : ansiString):boolean;override;
end;

function tFieldDoubleXML.innerlink;
var
  dd : double;
  sd,
  sn : string;
begin
  dd:=double(efVarPtr^);
  if dd=0 then sn:=''
  else
    sn:=formsg(dd);
  if readOnly then
    sd:=' disabled="disabled" '
  else
    sd:='';
  oldVal:=dd;
  result:='<input id="'+nazwaField+'" type="text" '
    +sd+mask(fmNumeric)
    +ajaxChange+' value="'+sn+'">';
end;

function tFieldDoubleXML.valueHTML;
begin
  if (double(efVarPtr^) <> oldVal) then begin
    oldVal:=double(efVarPtr^);
    if oldVal=0 then astr:=''
    else astr:=formsg(oldVal);
    result:=true;
  end else result:=false;
end;

procedure tFieldDoubleXML.postField;
var
  sn : string;
  dd : double;
begin
  dd:=str2dbl(aWartosc,',');
  if double(efVarPtr^)<>dd then modified:=true;
  double(efVarPtr^):=dd;
  efPostEdit;
end;
```

4. Optymalizacja

Optymalizacja transferu

Stacja użytkownika komunikuje się z serwerem po prawie każdej akcji użytkownika. Szczególności po naciśnięciu przycisku lub zmianie pola. Przeglądarka wysyła odpowiednią informację do serwera, ten odpowiada ciągiem znaków, który może być:

- tekstem źródłowym okna, wtedy następuje podmiana innerHTML dla elementu BODY
- oknem wyskakującym (komunikat, popupmenu), wtedy wywoływana jest odpowiednia procedura
- modyfikacją istniejącego okna: ciąg zawiera identyfikatory DIV, które mają zostać zmienione oraz ich treści
- zmiany wartości zapisanych w polach: różnica w stosunku do stanu powyżej polega na tym, że przeglądarka nie niszczy starego komponentu, tylko zmienia jego wartość. Dla bardziej skomplikowanych kontrolek zmiana stanu polega na zmianie znacznie większej liczby parametrów jak tylko wartość, natomiast dla pól tekstowych czy checkboxów takie rozróżnienie skutkuje zwiększeniem szybkości przetwarzania zmian oraz skrócenie przesyłanego kodu (przesyłamy tylko wartość a nie całą definicję komórki)

Zadaniem serwera jest każdorazowe obliczenie nowej postaci okna i wysłanie tylko tych fragmentów i w takiej postaci, aby wszelkie zmiany były odzwierciedlone przy zachowaniu optymalnej efektywności pasma.

Optymalizacja aplikacji serwerowej

- szybka obsługa zapytań i generowanie odpowiedzi. Wymagane jest to tego efektywne narzędzie do manipulowania łańcuchami tekstowymi, co w przypadku pascala jest dobrze zrealizowane. W zależności od potrzeb można używać krótkie (szybkie, nie wymagające alokacji) lub długie (brak ograniczeń na długość) łańcuchy.
- integracja z częścią „okienkową” - czyli frameworkiem obsługującym część użytkową,
- rozdzielenie poszczególnych sesji (w przyszłości konieczny pool-threading),
- zapewnienie bezpieczeństwa aplikacji: zawieszenie się jednej sesji nie powinno skutkować uszkodzeniem innych,
- poinformowanie przeglądarki o możliwości buforowania niektórych plików. W szczególności dotyczy to plików graficznych, skryptów czy stylów.

Uzyskane wyniki

Działanie aplikacji webowej opartej na omawianym frameworku Xweb są bardzo obiecujące. Czas opóźnienia mierzony od strony serwerowej liczony jest w milisekundach (zazwyczaj mniej jak 12ms, w większości ~2ms). Można więc przyjąć że bez większego problemu pojedynczy serwer

przyjmie obciążenie kilkudziesięciu, a nawet kilkuset równoległych instancji aplikacji. Większym ograniczeniem będzie wielkość pamięci konieczna do przydzielenia, gdyż zaawansowane aplikacje bazodanowe zużywają wiele MB RAM.

Od strony użytkownika wszystko zależy od łącza oraz szybkości komputera. Ponieważ przesyłane pliki nie są zbyt duże, istotnym parametrem jest opóźnienie do serwera, które w skrajnym przypadku, gdy droga do serwera jest długa, wielostopniowa może wynosić 200ms. Jest to niewielki czas, gdy oczekujemy od aplikacji szybkiego działania (czas reakcji <1s). Natomiast dla osób przyzwyczajonych do aplikacji reagujących natychmiastowo (czas reakcji <0.3 s) jest to zauważalne opóźnienie. Do tego należy doliczyć czas rysowania strony przez przeglądarkę. Nowszy sprzęt (procesor dwurdzeniowy) posiada taką wydajność, że większość zmian treści strony dokonuje się natychmiast. Dodatkowym utrudnieniem jest przyjęte założenie, że każda akcja użytkownika transmitowana jest do serwera. Jest to bardzo ważne założenie, gdyż znakomicie zwiększa wygodę aplikacji oraz upraszcza jest tworzenie. Natomiast skutkuje zwiększonym obciążeniem.

Wprowadzenie obsługi klawiatury do aplikacji webowej znacząco zmniejszyło dopuszczalną granicę opóźnienia reakcji dla programu. Sterowanie klawiaturą jest znacząco szybsze, więc aplikacja powinna odpowiednio szybko reagować. Dlatego też znacząco rozbudowano moduł optymalizujący transfer, dzięki czemu aplikacja uzyskała zadowalającą efektywność.

Dla porównania ta sama aplikacja, lecz obsługiwana po sieci lokalnej (serwer i użytkownik w tej samej sieci) reagowała natychmiast przy każdym zdarzeniu. Przyjmując ciągły rozwój wydajności zarówno komputerów jak i sieci, w niedalekiej przyszłości w większości przypadków realizacja oczekiwania natychmiastowej reakcji będzie znacznie łatwiejsza do spełnienia.

5. Do zrobienia

- obsługa klawiatury,
- komponent virtualtreeview,
- wykorzystanie SVG (bardziej wyrafinowane kontrolki)
- implementacja jako FastCGI,
- przekazanie biblioteki do środowiska OpenSource.

Pominięto takie zagadnienia:

- raportowanie : wykorzystanie przeglądarki PDF
- drukarki i kasy fiskalne: (potencjalnie ActiveX)
- komunikacja z pakietem biurowym

6. Porównanie technologii

	Delphi	java	PHP	C#
Szybkość	Szybki (kompilowalny)	Średni (JIT)	Wolny (interpreter)	Średni (JIT)

Dariusz Mazur: Tworzenie aplikacji web based w Pascalu
Jubileuszowy Zlot Programistów Delphi 2009

Przenośność	Platformy linux , windows, Mac OS. Niewielkie zależności. Łatwa instalacja.	Wiele platform. Duża zależność od wersji środowiska. Konieczność stosowania przeglądarki i dodatkowych pluginów.	Wiele platform. Łatwa instalacja. Konieczność pokazania kodu. Silna zależność od wersji interpretera. Przeglądarka dowolna.	Tylko DotNet. Duża zależność od wersji środowiska. Ograniczona możliwość używania innych niż IE przeglądarek.
Tworzenie złożonych aplikacji	Wieloletnie doświadczenie użytkownika w złożonych projektach.	Wiele projektów komercyjnych, szczególnie korporacyjnych. Mniej popularne na rynku SOHO.	Bardzo duża popularność w małych projektach. Złe doświadczenia z dużymi. Brak bibliotek wspierających tworzenie złożonych aplikacji.	Porównywalny do Javy, lecz mniejsze doświadczenie komercyjne.
Serwer	Wbudowany serwer HTTP,	Java EE	Apache	ASP.NET
Bezpieczeństwo	Średnie: brak kontroli wykonywania kodu, błędy przepełnienia itp.	Średnie JIT potencjalnie umożliwia wprowadzenie i wykonanie nowego kodu.	Słabe.	Analogicznie jak Java.
Interfejs użytkownika	Dojrzały, sprawdzony i skuteczny framework VCL. Niskie doświadczenie na polu web-based. Stworzony from scratch wymaga jedynie przeglądarki. Kod HTML tylko w definicja komponentów.	Szereg frameworków do tworzenia aplikacji web-based. Często uzależniony od dodatkowych pluginów.	Szereg bibliotek udostępniających pełne możliwości GUI, kod html Przeplatany z kodem programu.	Szereg frameworków do tworzenia aplikacji web-based. Często uzależniony od dodatkowych pluginów.
Uruchamianie	Delphi, Lazarus, MSEIDE	eclipse	utrudnione	VisualStudio

7. Wnioski:

Jeden kod na wiele platform (vide FPC): desktop Windows – web based. Serwer obecnie Linux i Windows, potencjalnie inne, jako przeglądarka IE, Firefox, Safari, Opera również w wersjach na palmtopy.

0 kodu HTML

0 różnic wersja mobilna – desktopowa

0 zależności od środowiska i dystrybucji

Całość programu zawarta w pojedynczym EXE o stosunkowo niewielkiej objętości i niewielkiego zapotrzebowania na pozostałe zasoby, co wskazuje na potencjalne zastosowanie w systemach wbudowanych.

Streszczenie

Tworzenie aplikacji desktopowych w Delphi to oczywistość. Natomiast web based: większość powie „nie da się”. Czy ekonomiczne jest jednak zarzucenie dużej aplikacji i rozpoczęcie pisania jej od nowa w bardziej modnym języku? Grozi to olbrzymimi kosztami, aż do wypadnięcia z rynku włącznie. Autor dokonał już z powodzeniem przekładki aplikacji z Dos na Windows i z Windows na Linux. Te doświadczenia wykorzystał do zmierzenia się z kolejnym wyzwaniem: umożliwienie wejścia na najbardziej modną platformę: Internet i to przy zachowaniu w większości obecnego i przyszłego kodu. Zostanie przedstawiona biblioteka, jej architektura, możliwości i sposób wykorzystania. Omówione też zostaną te aspekty aplikacji web based, które są szczególnie istotne czy też trudne dla twórców aplikacji desktopowych.

Agenda

1. Aplikacja wieloplatformowe
2. Cel
3. Architektura rozwiązania
4. Komunikacja z przeglądarką
 1. podobieństwa i różnice
 2. stosowane technologie
5. API biblioteki
 1. TobjectXML = class(tComponent)
 2. tApplication
 3. Kontrolki webowe: tControl, tEdit, tForm; zgodność na poziomie nazewnictwa
 4. Jak kompilować ten sam program na dwie platformy
6. Tworzenie aplikacji web based
 1. Program „hello world”
 2. Korzystanie z elementów standardowych
 3. Styl programowania
 4. Rozszerzenia kontrolki
 5. Grid
 6. Okno dialogowe
 7. Kontrolka progresji
7. Pozostało do zrobienia i wnioski

Notka biograficzna

Informatyka to jego pasja. Zajmuję się programowaniem od dawna, dość powiedzieć że zawodowo ponad 20 lat. Od pierwszych prób do dzisiaj prawie wyłącznie w pascalu. Twierdzi, że nie ma języka efektywniejszego i o szerszym spektrum zastosowań niż pascal.

Jest głównym architektem aplikacji Madar ERP oraz Invoicer. Prywatnie mąż i ojciec dwójki dzieci. Hobbystycznie uwielbia konie oraz górskie i rowerowe wycieczki.

Kontakt

Więcej informacji na stronie: <http://www.emadar.com/fpc/xweb.htm>

Kontakt z autorem: