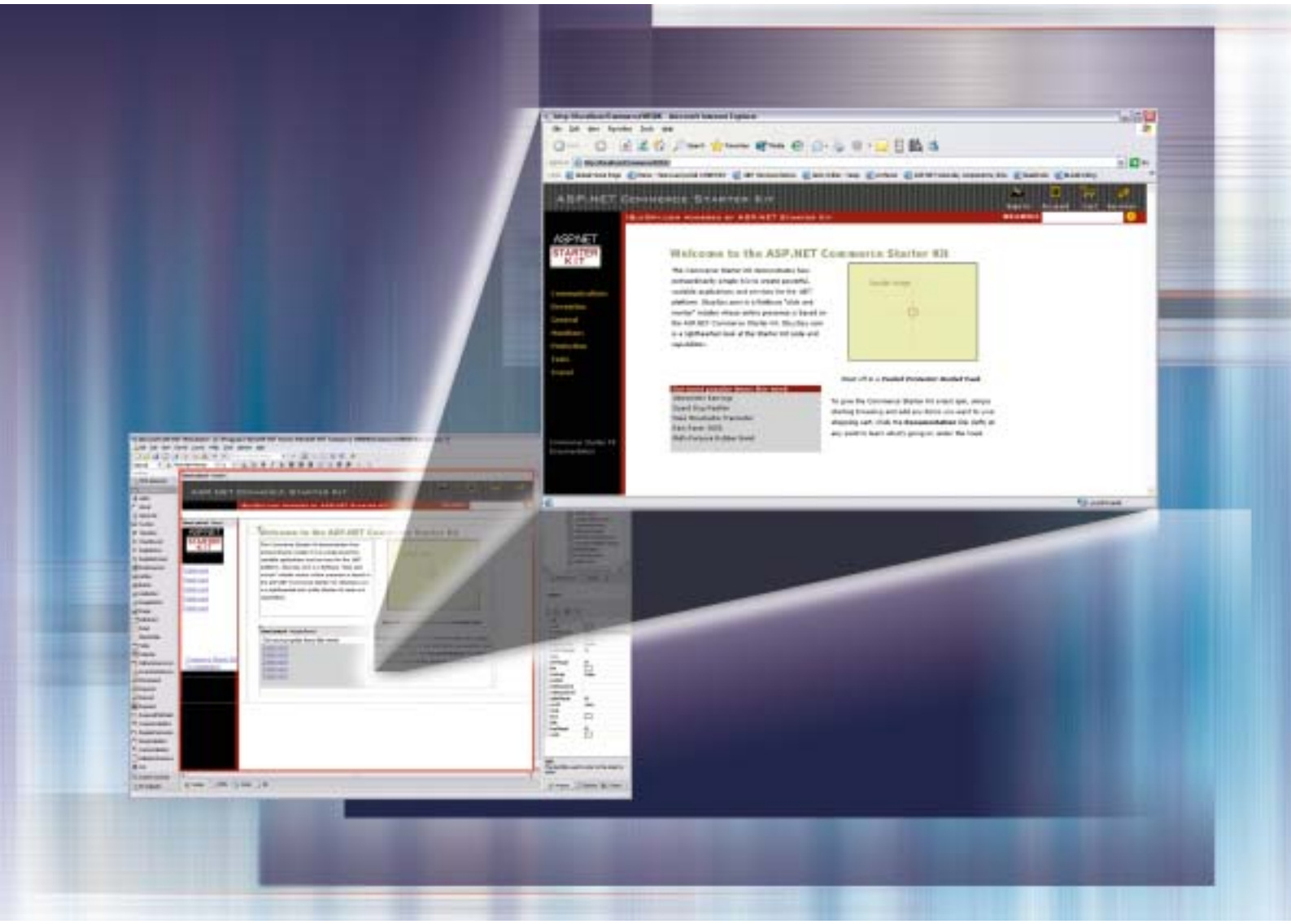
Microsoft .net



Ľuboslav Lacko

ASP.NET pre začiatočníkov

Microsoft®

Obsah

Úvod

Pri surfovaní na Internete sa stretávame s viacerými druhmi webových aplikácií. Pod pojmom webová aplikácia, rozumieme aplikáciu, ktorá je umiestnená na serveri, pričom vo väčšine prípadov na serveri beží aj jadro výkonného kódu aplikácie a spravidla takáto aplikácia využíva pre svoju činnosť aj nejakú databázu. Pokiaľ sa pokúsime webové aplikácie kategorizovať, dospejeme k „hrubému“ rozdeleniu na klientské a serverové aplikácie

Klientské aplikácie
Tieto aplikácie sú na serveri len umiestnené využívajú HTML kód , prípadne skripty, ktoré sa interpretujú na strane klienta. Môžu byť statické (HTML) a dynamické (DHTML). Tieto jednoduché aplikácie, hlavne dynamické dokážu pri vhodnom využití možnosti HTML a skriptových jazykov na strane klienta vytvoriť určitú mieru informačnej funkcionality a pomerne zaujímavé efekty.

Serverové aplikácie
Začínajúci programátor HTML stránok, očarený možnosťami DHTML kódu a skriptov na strane klienta, hlavne ak má k dispozícii kvalitný nástroj, napríklad Microsoft FrontPage si možno myslí, že s pomocou skriptov na strane klienta sa dá zvládnuť prakticky všetko. Bohužiaľ to nie je pravda. Niektoré, na prvý pohľad jednoduché a bežné veci sa bez podpory skriptov na strane servera realizovať nedajú. Nedokážeme na strane klienta naprogramovať dokonca ani jednoduché počítadlo prístupov na našu stránku. Medzi neriešiteľné úlohy by taktiež patrili aj všetky tie ankety, publikačné portály internetové obchody, zoznamky, chaty a iné aplikácie s ktorými sa na webe denne stretávame..
Hlavný problém totiž je, že vo všetkých vymenovaných typoch aplikácií potrebujeme väčšinu stránok generovať dynamicky na základe vybraných údajov, napríklad objednávky a podobne. Pre ukladanie údajov sa využívajú databázy. Vybudovanie internetovej aplikácie typu e-bussiness s automatickým spracovaním objednávok, sledovaním stavu zásob, prípadne s riadením marketingovej stratégie bez výkonného databázového servera nie je možné a ako si ukážeme neskôr budeme potrebovať aj výkonnú serverovú technologickú platformu a dobré vývojové prostredie.
Z hľadiska financovania prevádzky sa webové aplikácie rozdeľujú na komerčné, aplikácie rôznych inštitúcii a aplikácie typu hobby. V tomto článku sa zameriame hlavne na hobby aplikácie. Náklady na webové sídlo môžeme rozdeliť na dve časti, jednak sú to jednorázové náklady na jeho zriadenie a potom pravidelné náklady na prevádzku.
Z technického hľadiska potrebujeme na vývoj a následné fungovanie webovej aplikácie serverový operačný systém, vo väčšine prípadov aj databázový server a aplikačná logika je vytvorená pomocou nejakého skriptového systému. Poďme ale pekne po poriadku a skúsme analyzovať ktoré konkrétne produkty sa v jednotlivých segmentoch používajú najviac.

Komerčné aplikácie
U komerčných aplikácií, napríklad internetbankingu, internetových obchodov a podobne sa predpokladá, že budú svojmu majiteľovi prinášať zisk a tým pádom je jasné z akých prostriedkov sa uhradia počiatočné investície. Keďže každá sekunda výpadku business aplikácie sa dá vyčísliť pomerne vysokou finančnou čiastkou, dôraz sa kladie na spoľahlivosť, pričom počiatočné náklady nehrajú až takú rolu. Takže tu nachádzajú uplatnenie komerčné databázové servery napríklad Oracle 9i, Microsoft SQL Server 2000, IBM DB2... prípadne aj komplexné platformy pre e-business napríklad Microsoft Commerce Server, aplikačný server Oracle 9i, Web Sphere od IBM... všetko záležitosti hodne nad 10 000 USD.

Aplikácie inštitúcií
Asi nás to príliš nepoteší, ale aj aplikácie rôznych štátnych a iných inštitúcií majú svoj zdroj financovania, ktorým sú naše dane. Na rozdiel od predchádzajúcich aplikácií nemusia prinášať zisk a ich prevádzkovateľom ani príliš nezáleží či budú navštevované. Informovanosť občanov nebýva hlavným cieľom snahy štátnych úradníkov, skôr sú radšej keď im na ich často nekalé praktiky verejnosť nedíva.

Aplikácie typu hobby
V tejto skupine vývojárov a prevádzkovateľov webových aplikácií je zrejme najviac tvorivého potenciálu ale bohužiaľ najmenej financií. Reklamné agentúry síce občas „rozpustia“ do webového priestoru nejaké prostriedky za zobrazenie bannerov, ale napríklad na nákup licencie databázového serveru to ani zďaleka nepostačuje. Takže v segmente hobby aplikácií kraľujú voľne šíriteľné produkty pričom azda najčastejšie sa využíva kombinácia operačného systému LINUX, webového servera APACHE, databázového servera MySQL a skriptového systému PHP.

Ak by sme to zrekapitulovali, na „serverovej“ strane webových aplikácií typu hobby má prevahu LINUX ale na klientskej strane, čo sú u hobby aplikácií domáce počítače pripojené na Internet a samozrejme počítače vo firmách (klienti hobby aplikácií totiž určite občas využijú aj počítač vo firme), tam má zase prevahu operačný systém Windows. Tu sa pred rokmi podarilo Microsoftu obsadiť so svojim Internet Explorerom rozhodujúci podiel na trhu webových prehliadačov. O niečo podobné sa teraz snaží Microsoft aj na serverovej strane, veď webovú stránku má dnes

prakticky každá firma a aj veľa jednotlivcov. A tieto kategórie sa samozrejme aj prelínajú, veď úspešný manažér veľkej firmy môže byť aj zanieteným pestovateľom kaktusom, chovateľom exotických živočíchov a podobne.

V prípade že si chceme zriadiť či už osobnú webovú stránku, alebo stránku na ktorej budeme propagovať našu záľubu stačí nájsť nejaké webhostingové sídlo, vyvinúť alebo z komponentov poskladať svoju webovú aplikáciu a umiestniť ju na server. Vo väčšine prípadov u webhostingu natrafíme práve na kombináciu LINUX - MySQL - PHP. Existujú samozrejme aj webhostingové sídla, ktoré využívajú operačný systém Windows, skriptový systém ASP (najnovšie ASP.NET) a databázový server SQL Server 2000.

Čo potrebujeme pre prevádzku jednoduchej webovej aplikácie

Webová aplikácia potrebuje na strane servera tieto služby:

- Operačný systém podporujúci sieťové služby
- Webový server
- Databázový server

Môžu to byť napríklad prostriedky od Microsoftu: Operačný systém Windows NT/2000/XP/2003 Server, Internet Information Server a databázový server Microsoft SQL Server 2000. Niektoré z týchto programov sú alebo boli pomerne drahé, preto sa používajú aj voľne šíriteľné programy pod operačným systémom LINUX, napríklad Apache web server a databázový server MySQL.

Nasledujúca tabuľka prehľadne zobrazuje popísané možnosti.

Operačný systém	Webový server	Databázový server
Linux	Apache	MySQL
Windows 2000/XP/2003	Internet Information Server	SQL Server 2000
Windows 98	Personal Web Server	MSDE

Nie je to nijaké pevné pravidlo, existuje napríklad Windows verzia populárneho webového serveru APACHE aj databázového servera MySQL. Taktiež pod operačným systémom LINUX môžeme použiť (v obmedzenom rozsahu) engine pre ASP stránky.

Náčrt architektúry a možnosti implementácie rôznych aplikácií na serveri

Na tomto mieste je vhodné pre zopakovanie vysvetliť princíp práce väčšiny webových aplikácií. Vysvetlíme základný princíp, ako funguje napríklad jednoduchý zásielkový obchod s knihami.
Klient prostredníctvom prehliadača web stránok zadá adresu požadovanej stránky. Na hlavnej stránke virtuálneho kníhkupectva si vybrerie žáner alebo kategóriu, napríklad detektívku. Na ďalšej stránke si vyberie konkrétny titul a objedná si ho. Klientove požiadavky sú smerované na príslušný webový server. Tento postupne generuje HTML stránky, ktoré budú prostriedkami siete doručené klientovi. Stránky budú pravdepodobne obsahovať texty, tabuľky, obrázky, komponenty a skriptové kódy. Vývojári aplikácie „webového kníhkupectva“ nemohli predpokladať, aké tituly sa budú predávať, ani čo si ktorý klient bude prezerať, alebo objednávať. Preto HTML stránky nie sú uložené na diskoch servera v statickej podobe, ale sa dynamicky generujú podľa požiadaviek klientov, na základe informácií o jednotlivých tituloch, uložených v databáze kníh.

Operačný systém a webový server

Firma Microsoft si uvedomila problém ceny svojich špičkových produktov, preto pre menej náročné aplikácie distribuuje spolu s operačnými systémami Windows 98 a Millenium Edition jednoduchšiu verziu Internet Information Serveru pod názvom Personal Web Server. Obvykle sa webové servery umiestňujú v klimatizovaných serverových sálach s prísnymi bezpečnostnými opatreniami. S použitím Windows 98 (alebo vyššieho), Personal Web Servera (nachádza sa na inštalačnom CD Windows v adresári AddOn) a jadra databázového serveru MSDE si môže každý čitateľ pokusne nakonfigurovať svoj domáci počítač ako webový a databázový server a dokonca je možné ladiť aplikácie na tom istom počítači pomocou prehliadača HTML stránok

Databázový server

Problém s databázovým serverom pre webové aplikácie typu hobby bol na platforme Windows a .NET vyriešený produktom MSDE (Microsoft Data Engine), čo je plne funkčná, a podľa licenčného ujednania v prípade Web Matrixu s v ňom vyvinutou webovou aplikáciou voľne šíriteľná verzia SQL Serveru. Jediným obmedzením je veľkosť databázy,

ktorá nemôže prekročiť 2 GB. Pre ilustráciu: 2 GB pamäti postačujú pre uloženie informácií o všetkých telefónnych hovoroch z jednej dekády ústredne (10 000 účastníkov, spolu 100 000 hovorov denne) po dobu 6 mesiacov. Kompatibilita s SQL Serverom umožňuje projekt kedykoľvek prekonvertovať na SQL Server bez akejkoľvek zmeny v návrhu databázových tabuliek, alebo uložených procedúr. MSDE spája bezpečnosť a spoľahlivosť SQL Servera, na platforme Windows NT. Bližšie informácie o MSDE sú na adrese http://www.microsoft.com/sql/msde. Podrobnejšie informácie o MSDN vrátane inštalácie, administrátorských a klientských aplikácií sú v piatej kapitole o vývoji databázových aplikácií

Kód aplikácie na strane servera
Zatiaľ sme sa nedostali k vlastnému jadru kódu webovej aplikácie. Okrem HTML kódu a prípadne databázového jazyka SQL budeme potrebovať aj nejaký programovací jazyk alebo skriptový systém. V nedávnej histórii sa najčastejšie používali ASP stránky a PHP skripty. S nástupom nových technologických platforiem a komplexných aplikačných prostredí napríklad Microsoft .NET Framework, Oracle 9i AS, IBM WebSphere... vystupujú do popredia programovacie jazyky JAVA (Microsoft ponúka Visual J#), Visual Basic a Visual C#. Závisí to aj od použitej softvérovej platformy. Na serveroch s operačným systémom Windows NT/2000/XP sa používa najčastejšie webový server IIS (Internet Information Server) s ASP, prípadne novšie ASP.NET stránky. Pod operačným systémom UNIX alebo LINUX sa najčastejšie používa web server APACHE a PHP skripty, prípadne Java.

Princíp činnosti skriptových serverových stránok (ASP, PHP...) spočíva v postupnej analýze kódu takejto stránky. Ak niektorý riadok obsahuje skriptový kód, tento sa interpretuje a jeho výstupy sú vložené do HTML stránky, ktorá sa dynamicky generuje pre klienta. Ak riadok neobsahuje skriptový kód, je spravidla vložený do HTML stránky priamo. A aby to nebolo také jednoduché, aj HTML stránky, generované serverom, ktoré sa potom zobrazujú u klienta obsahujú spravidla okrem HTML kódu aj kódy v skriptovom jazyku, ktoré sa vykonávajú pre zmenu na klientskom počítači. Na strane klienta sú najrozšírenejšie skriptové jazyky **JScript** a **VB Script**.

Poznámka
Táto brožurka je šírená dvomi spôsobmi. Buď v tlačenej podobe a vtedy je jej súčasťou aj CD-ROM, kde všetky potrebné nástroje nájdete, prípadne je brožúra k dispozícii v elektronickej podobe konkrétne fo formáte PDF a vtedy si čitateľ musí všetky aplikácie (WebMatrix, MSDE...) stiahnuť z webu

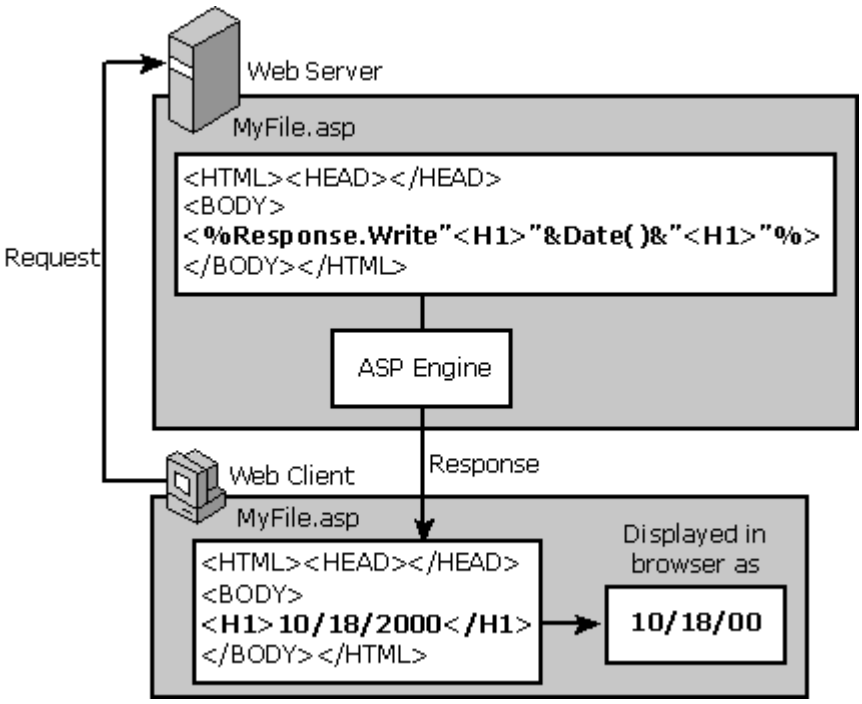
KAPITOLA 1:

Stručný prehľad technológií ASP, PHP a JSP

Našim primárnym cieľom je síce technológia ASP.NET, no aby sme mohli porovnávať predstavíme aspoň v stručnosti jednotlivé, dalo by sa povedať už „historické“ skriptové systémy (veď sa používali už v minulom storočí :-)).

ASP stránky a PHP skripty

ASP stránky pracujú na princípe vkladania kódu oddeleného oddeľovačmi, párovými znakmi **<% ... %>**. Súbory majú spravidla príponu **.ASP**, Skriptový jazyk pre ASP stránky môže byť VBScript, alebo JScript. VBScript je skriptový jazyk odvodený od Visual Basic for Aplications. JScript je implementácia Java Scriptu. Teoreticky je možné na jednej stránke použiť obidva skriptové jazyky, z hľadiska výkonu to nie dobré riešenie, pretože server musí použiť dva skriptové stroje na spracovanie jednej stránky. Dokonca ani vtedy, ak nám nezáleží na rýchlosti, nemôžeme sa spoľahnúť, že skriptové stroje ukončia svoju prácu a uložia výsledky do HTML stránky v takom poradí v akom boli spustené.



Princíp činnosti ASP stránok

Ak riadok na ASP stránke obsahuje skriptový kód, tento sa interpretuje a jeho výstupy sú vložené do HTML stránky pre klienta. Ak riadok neobsahuje skriptový kód, je vložený do HTML stránky priamo. Novšie verzie serverov najskôr skontrolujú celý súbor a ak tento neobsahuje žiadny skriptový kód, priamo ho vložia do HTML stránky. Preto môžeme používať príponu ASP pre všetky stránky bez ohľadu na to, či obsahujú, alebo neobsahujú riadky skriptového kódu.

PHP skripty používajú ako oddeľovače párové znaky **<? ... ?>**. Skriptový jazyk je podobný C++, alebo Jave. Pre súbory so skriptami PHP sa používa prípona php.

Pre ilustráciu syntaxe PHP a ASP uvediem niekoľko príkladov. Prvý bude príklad typu Hello Word, ktorý vypíše na obrazovku jednoduchý oznam. V našom prípade by výpis oznamu zvládol HTML príkaz, preto najskôr uložíme text do premennej a potom vypíšeme do HTML stránky jej obsah.

Hello.asp

```
<HTML>
  <BODY>
  <%
    txt = "Ahoj svet"
    Response.write (txt)
  %>
  </BODY>
</HTML>
```

Hello.php

```
<HTML>
  <BODY>
  <?
$txt = "Ahoj svet";
echo $txt;
  ?>
  </BODY>
</HTML>
```

V obidvoch prípadoch bude klientovi odoslaný HTML kód

```
<HTML>  <BODY>
  Ahoj svet
</BODY> </HTML>
```

Pre ilustráciu ukážeme ešte príklad cyklu a podmienky. Všimnime si kombináciu skriptu a HTML príkazu vo vnútri cyklu.

Cyklus.asp

```
<% for i = 1 to 5 %>
Opakovany text<BR>
<% Next %>
```

Cyklus.php

```
<? for($i=1; $i<=5; $i++): ?>
Opakovany text<BR>
<? endfor ?>
```

Podmienka.asp

```
<%
  cas = Time()
  if cas >= #12:00:00 AM# and cas <= #12:00:00 PM# then
  Response.write "Prijemne dopoludnie"
  else
  Response.write "Prijemne popoludnie"
  end if
%>
```

Podmienka.php

```
<?
    if (date("A")=="AM")
      {echo " Prijemne dopoludnie ";}
    else
   { echo " Prijemne popoludnie ";}
?>
```

Podobné je aj spracovanie údajov, ktoré zadal používateľ pomocou formulára. Riešenie pre ASP sa skladá zo stránky **formular1.html**, obsahuje jednoduchý formular pre zadanie mena.

```
<h1>formular1</h1>
<FORM ACTION="formular.asp" METHOD="post" ><P>
<B>Meno:</B>
<TD><INPUT id=Meno name=Meno> </P><P>
<INPUT TYPE=submit value=Potvrd id=submit1 name=submit1> </P>
</FORM>
```

ASP stranka **formular.asp** preberie údaje pomocou metódy Request.Form od klienta a vypíše ich.

```
html><head> </head>
<body><h1>Vypis udajov z formulara</h1>
<%
     Response.write ("Meno: " &Request.Form("Meno") & "<BR>")
%>
</body></html>
```

Pre ilustráciu spracovania údajov z formulára pomocou skriptov PHP vytvoríme stránku **formular2.html**,

```
<h1>formular1</h1>
<FORM ACTION="formular.pho" METHOD="post" ><P>
<B>Meno:</B>
<TD><INPUT id=Meno name=Meno> </P><P>
<INPUT TYPE=submit value=Potvrd id=submit1 name=submit1> </P>
</FORM>
```

a skript **formular.phtml**.

```
html><head> </head>
<body><h1>Vypis udajov z formulara</h1>
<?
  echo "Meno: ";
  echo $Meno;
?>
</body></html>
```

Pri ASP stránkach musíme rozlišovať medzi metódou **POST**, alebo **GET** pre odovzdanie údajov z formulára. Metóda **POST** zabalí všetky potrebné údaje z formulára a odošle ich na server. ASP stránka na serveri prečíta tieto údaje volaním funkcie **Request.Form**. Metóda **GET** odovzdá parametre ASP stránke ako súčasť URL adresy, pre zistenie parametrov na strane servera použijeme funkciu **Request.QueryString**. Pri použití skriptov PHP vždy zodpovedá meno poľa názvu premennej bez ohľadu na použitú metódu.

Konektivita na databázu

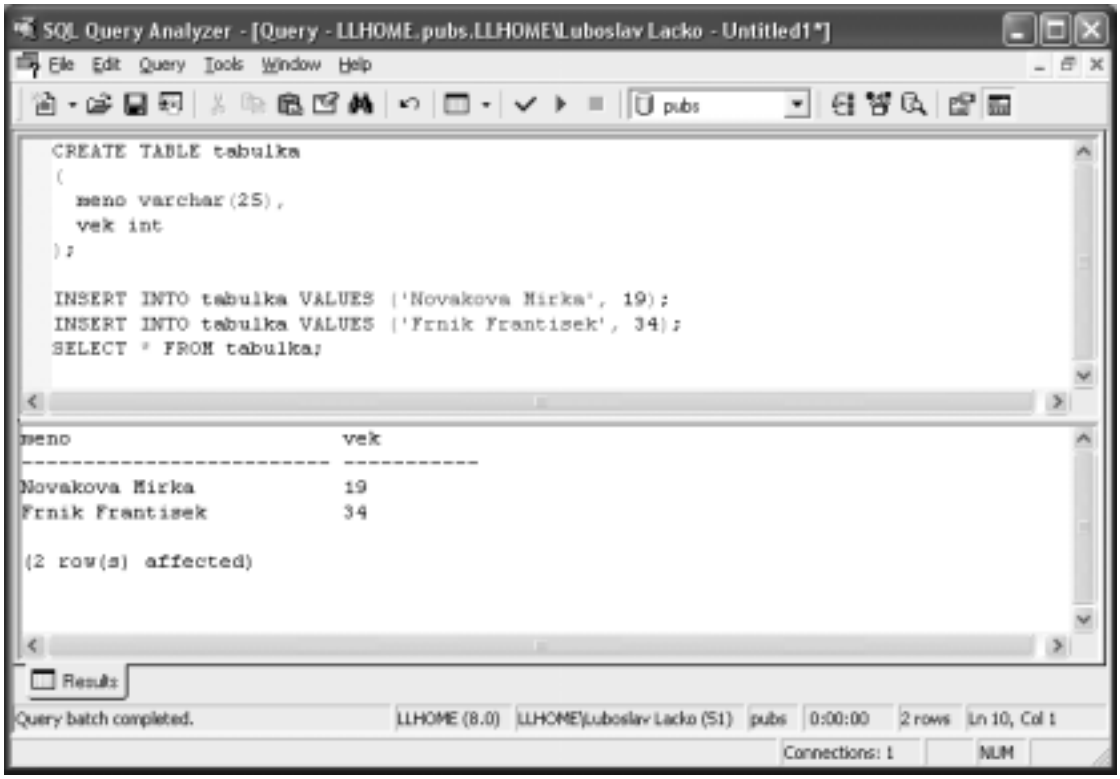
Pri riešení niektorých jednoduchých typov aplikácií napríklad počítadla prístupov a podobne môžeme síce zapisovať údaje do súborov a potom ich odtiaľ načítať. (príklad riešenie počítadla v ASP)

```
<html><head></head><body> <h1>Pocitadlo pristupov</h1>
<%
On Error Resume Next
err.Clear

       Dim ObjSubor  ' Vytvorime instanciu objektu FileSystemObject
       Set ObjSubor = Server.CreateObject("Scripting.FileSystemObject")

       dim TxtSubor  ' Otvorime textovy subor
       Set TxtSubor=ObjSubor.OpenTextFile("c:/pocitadlo.txt")

if err.number<>0 then ' Ak taky subor nejestvuje, tak ho vytvorime
   Response.write ("Prvy pristup na stranku")
   Set TxtSubor=ObjSubor.CreateTextFile("c:/pocitadlo.txt")
   TxtSubor.WriteLine (1)
   TxtSubor.Close
   err.clear
else
          nn = TxtSubor.ReadLine  ' Doterajsi pocet pristupov
          nn = nn+1 ' Inkrementujeme
        Response.write("Pocet pristupov: " & nn) ' Vypiseme
        TxtSubor.Close
        ObjSubor.DeleteFile("c:/pocitadlo.txt") ' Vymazeme stary subor
```

```
        Set TxtSubor=ObjSubor.CreateTextFile("c:/pocitadlo.txt")
        TxtSubor.WriteLine(nn) ' Do noveho ulozime pocet pristupov
        TxtSubor.Close
end if
%>
</BODY></HTML>
```

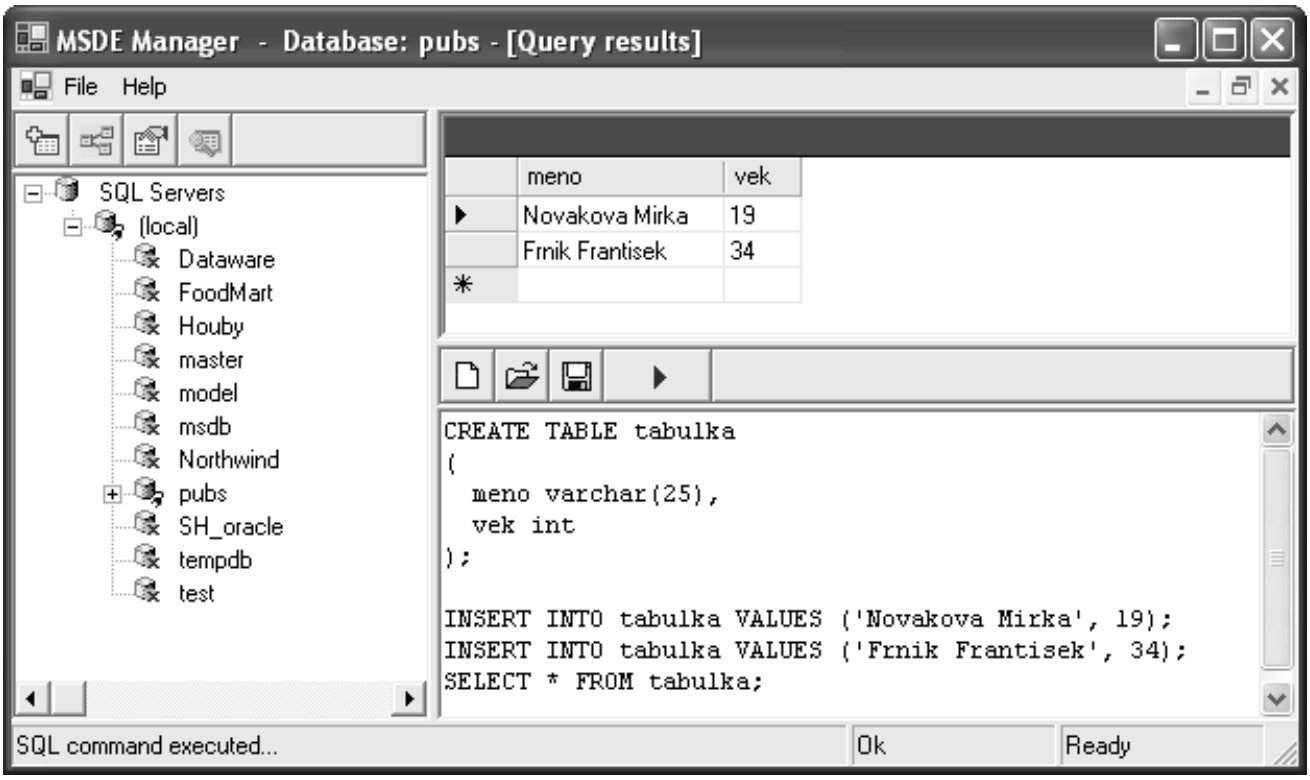
Táto technológia v núdzi postačuje aj pre jednoduché hobby aplikácie, napríklad pre evidenciu adries zberateľov a podobne. Pokiaľ však potrebujeme vyhľadávať informácie v súbore niekoľko tisíc položiek, alebo poskytnúť údaje utriedené podľa rôznych kritérií, zistíme, že súborový systém ukladania údajov je veľmi neefektívny. Každú operáciu musíme sami pracne naprogramovať a rýchlosť výstupu nás málokedy uspokojí. Každý sa snaží získať, zhromaždiť a spracovať čo najviac informácií. Väčšie množstvo informácií ešte samo o sebe neznamená vyššiu kvalitu. Dôležité je, ako bezpečne a spoľahlivo sú informácie uložené a aký rýchly je k nim prístup.

Spôsob pripojenia sa takejto aplikácie na databázu závisí od použitého skriptového, alebo programovacieho jazyka. Obvykle pri vývoji webovej aplikácie odladíme najskôr databázovú časť pomocou klientskej konzolovej aplikácie databázového servera. V prípade SQL Servera sa používa aplikácia Query Analyzer, (nainštaluje sa spolu s SQL Serverom)



Obr. 1 – SQL Server Query Analyser

pre voľne šíriteľný databázový stroj MSDE môžeme použiť napríklad aplikáciu MSDE Manager, ktorú nájdeme na priloženom CD



Obr. 2 – SQL Server Query Analyser

Aby sme ukázali základný princíp konektivity na databázu, prácu s SQL príkazmi a výpis údajov, použijeme ako námet úplne jednoduchú databázovú tabuľku

Meno stĺpca	dátový typ
meno	varchar(25)
vek	int

Z hľadiska SQL príkazov ide o úplne triviálnu záležitosť. Celá aplikačná logika spočíva v týchto SQL príkazoch:

```
CREATE TABLE tabulka
(
  meno varchar(25),
  vek int
);

INSERT INTO tabulka VALUES ('Novakova Mirka', 19);
INSERT INTO tabulka VALUES ('Frnik Frantisek', 34);
SELECT * FROM tabulka;
```

ASP a databázy

Skripty na ASP stránkach pristupujú k údajom v databázach cez rozhranie ODBC (Open Database Connectivity) prostredníctvom komponentov ActiveX Data Object (ADO). Tieto komponenty sú umiestnené medzi aplikáciou a databázovým serverom. Opíšeme len najdôležitejšie z nich, ktoré sú potrebné pre spoluprácu webovej aplikácie a databázy.

Objekt **Connection** umožňuje pripojenie aplikácie k databáze. Vytvára sa metódou **CreateObject**:

```
set db_osoby=server.CreateObject("ADODB.Connection")
```

Pre otvorenie spojenia sa použije metóda Open.

```
db_osoby.Open "DSN=test","klient","heslo"
```

Ak potrebujeme vykonať SQL príkaz, použijeme metódu **Execute**

```
sSQL = "INSERT INTO tabulka VALUES ('Novakova Mirka', 19)"
db_osoby.Execute(sSQL)
```

Ak potrebujeme vypísať niektoré, v našom prípade všetky záznamy použijeme SQL príkaz:

```
SELECT * FROM test_db
```

Tušíme, že databázový server nám na tento dotaz nejako odpovedal (príkaz sme predsa predtým odskúšali pomocou konzoly, kde sa požadované informácie vypísali na obrazovku), ale my potrebujeme údaje umiestniť na HTML stránku. K tomu slúži objekt **Recordset**. Ako vyplýva z názvu, jedná sa o množinu záznamov, ktorá je odpoveďou servera na SQL príkaz.

```
Set rs_Zaznam = Server.CreateObject("ADODB.Recordset")
rs_Zaznam.Open strSQL, db_osoby
```

Našou úlohou je urobiť výpis údajov z recordsetu na HTML stránke. Vlastný výpis údajov sa vykoná v cykle DO - LOOP. Cyklus sa ukončí pri dosiahnutí konca záznamov (EOF) :

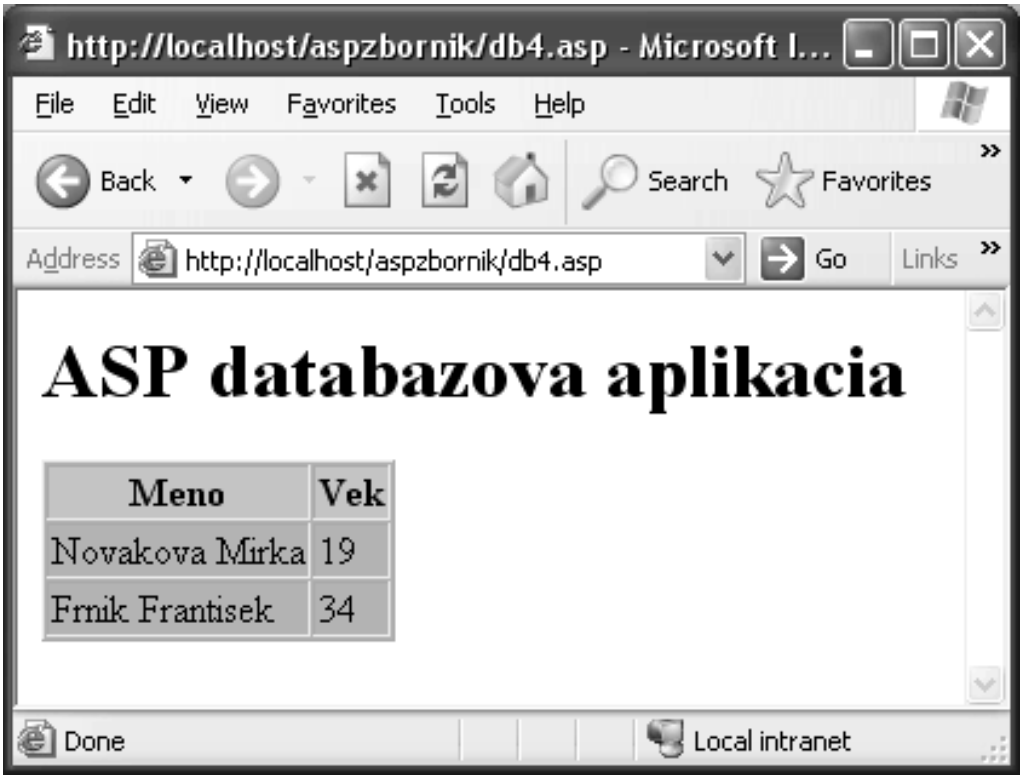
```
Do Until rs_riadok.EOF
  Response.Write "<TR><TD>" & rs_Zaznam.fields("meno")& "</TD>"
  Response.Write "<TD>" & rs_Zaznam.fields("vek")& "</TD>"
  Response.Write "</TR>"
  rs_Riadok.MoveNext
Loop
```

Kompletný kód jednoduchej ASP databázovej aplikácie potom bude:

```
<HTML><HEAD></HEAD><BODY><H1>ASP databazova aplikacia </H1>
<P><TABLE bgColor=aqua border=1 cellPadding=1 cellSpacing=1 >
<TR>
     <TH bgColor=#c0c0c0>Meno</TH>
     <TH bgColor=#c0c0c0>Vek</TH>
</TR>

<%
   set db_osoby=server.CreateObject("ADODB.connection")
   db_osoby.Open "DSN=test","klient","heslo"

    'Vytvorime tabulku
    Dim sSQL
    sSQL = "CREATE TABLE tabulka (meno varchar(25), vek int)"
    db_osoby.Execute(sSQL)

    'Vlozime zaznamy
     sSQL = "INSERT INTO tabulka VALUES ('Novakova Mirka', 19)"
     db_osoby.Execute(sSQL)
     sSQL = "INSERT INTO tabulka VALUES ('Frnik Frantisek', 34)"
     db_osoby.Execute(sSQL)

    Dim strSQL
    strSQL = "SELECT * FROM tabulka"
```

```
    'Vytvorime recordset
    Set rs_Riadok = Server.CreateObject("ADODB.Recordset")
    rs_Riadok.Open strSQL, db_osoby

    Do Until rs_Riadok.EOF
      Response.Write "<TR><TD>" & rs_Riadok.fields("meno")& "</TD>"
      Response.Write "<TD>" & rs_Riadok.fields("vek")& "</TD>"
      Response.Write "</TR>"
      rs_Riadok.MoveNext
    Loop

rs_Riadok.Close
Set rs_Riadok = Nothing

db_osoby.Close
Set db_osoby = Nothing
%>
</BODY></HTML>
```

Výsledkom našej snahy bude vytvorená, naplnená a v okne prehliadača HTML stránok zobrazená tabuľka



Obr. 3 – Jednoduchá ASP databázová aplikácia

Takáto aplikácia je síce funkčná, ale v reálnej praxi nepoužiteľná. Musíme totiž ošetriť potenciálne chyby. Pri práci s databázou sa môžu vyskytnúť rôzne chyby, napríklad sa pokúšame aktualizovať záznam, ktorý bol predtým vymazaný, používateľ sa prihlási nesprávnym heslom a podobne. Chyby sú síce ošetrené na úrovni databázového stroja, ale implicitné chybové hlásenia sú určené skôr pre vývojárov a nie pre klientov.

PHP a databázy

PHP skripty umožňujú okrem ODBC aj natívny prístup k údajom. Pre pripojenie k databáze sa používa funkcia **xxx_connect**, kde XXX je názov databázy, alebo rozhrania ODBC Napríklad

```
int odbc_connect ([adresa_servera] [, pouzivatel [, heslo]])
```

Ako ilustračný príklad ukážeme jednoduchú aplikáciu, ktorá bude využívať pripojenie k databázovému serveru Microsoft SQL Server 2000 cez rozhranie ODBC.

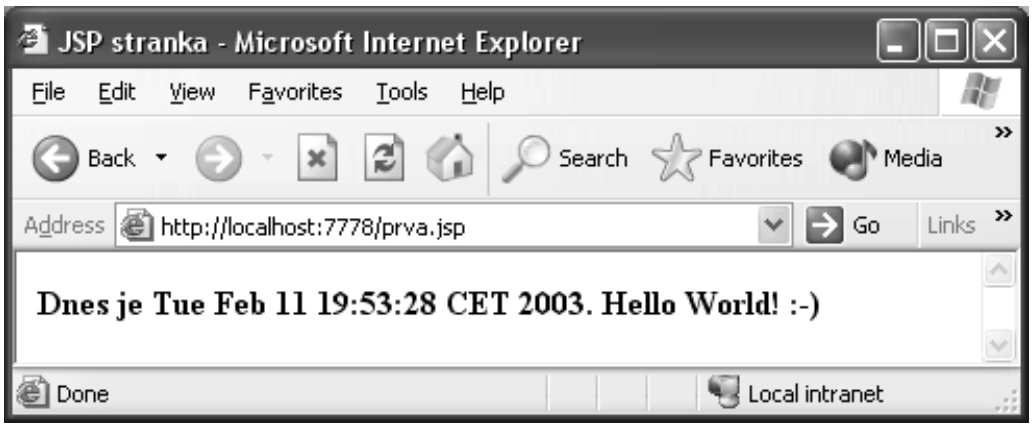
```
<HTML><BODY>
<TABLE BORDER=1 CELLPADDING=2>
<TR>
    <TH>Nazev</TH>
    <TH>Pocet kusu</TH>
</TR>
  <?
     $id_connect = odbc_connect("test","klient","heslo");
     odbc_exec($id_connect, "INSERT INTO tabulka VALUES ('Novakova Mirka', 19)");
     odbc_exec($id_connect, "INSERT INTO tabulka VALUES ('Frnik Frantisek', 34)");

     $id_sql = odbc_exec($id_connect, "SELECT * FROM tabulka");

  echo "<P>";
  while(odbc_fetch_row($id_sql))
  {
     echo "<TR>";
     echo "<TD>".odbc_result($id_sql, "meno")."</TD>";
     echo "<TD>".odbc_result($id_sql, "vek")."</TD>";
     echo "</TR>";
  }
  odbc_close($id_connect);
  ?>
  </BODY>
</HTML>
```

JSP stránky

Skratka JSP znamená Java Services Pages. Môžeme ich použiť na všetkých platformách (LINUX, Windows...). Engine pre JSP stránky môžeme napríklad doinštalovať do webového servera Apache. Pre naše príklady sme použili server Apache dodávaný spolu s databázou Oracle. Najskôr ukážeme jednoduchú stránku typu „Hello World"

```
<HTML><HEAD> <TITLE>JSP stranka </TITLE> </HEAD>
<BODY  BGCOLOR=white>
<P><B> Dnes je <%= new java.util.Date() %>.  Hello World! :-) </B></P>
</BODY></HTML>
```

Vidíme, že bloky príkazov v skriptovom jazyku JSP, sú podobne ako v ASP od zvyšku HTML stránky oddelené párovými znakmi **<% ... %>**. Výsledkom bude stránka



Obr. 4 – JSP stránka

Pre prístup k údajom v databázových tabuľkách môžeme použiť prístup pomocou objektov typu JavaBean, napríklad pomocou objektu DBBean. Pre ilustráciu ukážeme JSP stránku, ktorá pristupuje k databáze

```
<%@ page import="java.sql.*" %>
<jsp:useBean id="dbbean" class="oracle.jsp.dbutil.DBBean" scope="session">
<jsp:setProperty name="dbbean" property="User" value="scott"/>
<jsp:setProperty name="dbbean" property="Password" value="tiger"/>
<jsp:setProperty name="dbbean"
     property="URL" value="jdbc:oracle:thin:@LLHOME:1521:ORCL" />
</jsp:useBean>
<HTML><HEAD></HEAD><BODY>
<H2>JSP databazova aplikacia</H2>
<%
  try {
        String sql_string = "SELECT * FROM tabulka";
      //vytvorenie spojenia
        dbbean.connect();

      //Vykonanie SQL dotazu s vypisom vysledkov do HTML stranky
        out.println(dbbean.getResultAsHTMLTable(sql_string));

      //ukoncenie spojenia
        dbbean.close();
      } catch (SQLException e)
          {
             out.println("<P>" + "Chyba:");
             out.println ("<PRE>" + e + "</PRE> \n <P>");
          }
%>
</BODY></HTML>
```

Nevýhody ASP, PHP, JSP..

Aby sme mohli dostatočne oceniť novú platformu, musíme uviesť aj hlavné nevýhody ASP stránok, PHP skriptov a JSP stránok. Nevýhody uvedieme vo forme heslovitého prehľadu.

- kód vytvorený v spomínaných skriptových systémoch predstavuje lineárny bezstavový programový modul, so všetkými nevýhodami s tým spojenými.
- skriptový kód ASP, PHP a JSP sa v týchto systémoch mieša s HTML kódom. Nevýhoda je zrejmá. Na väčších webových projektoch pracujú obvykle týmy vývojárov, dizajnérov, grafikov a podobne. Pre vývojára môže predstavovať problém kód vytvorený grafikom, hlavne ak je webová stránka rozdelená do viacerých rámcov, prípadne ak sa používajú moderné technológie, ako napríklad Macromedia Flash. Pre grafika webových stránok je zasa záhadou skriptový kód, ktorý nachádzajú vo svojich predtým vytvorených stránkach, a to na tých najneočakávanejších miestach, napríklad vo vnútri HTML kódu pre vytvorenie tabuliek.
- dochádza k miešaniu jednotlivých architektonických vrstiev projektu
- vytvoriť vývojové prostredie pre takúto technológiu nie je práve jednoduché. Vývoj webových aplikácií na spomínaných platformách môže byť pomerne náročný a zdĺhavý a teda aj drahý.
- programátori sú zvyknutí využívať osvedčené moduly v rôznych projektoch. Spomínané technológie však takúto modularitu neumožňujú, jedine kopírovanie zdrojového kódu z jedného zdrojového súboru do nového súboru a jeho následné prispôsobenie.
- Po vygenerovaní HTML kódu na strane servera, bol tento kód odoslaný ku klientovi a následne server na všetko zabudol.

Okrem týchto základných nevýhod na ktoré narazia aj vývojári jednoduchých aplikácií typu hobby, obsahujú technológie ASP, PHP a čiastočne aj JSP ďalšie nevýhody na ktoré narazíme pri vývoji podnikových aplikácií. Napríklad ako používať stav na webových farmách, aby stavová informácia prežila pád aplikácie, a aby stav fungoval bez cookie. Pri aplikáciách s masívnym prístupom potrebujeme zvýšiť výkon použitím cache pre pseudostatické data. Väčšina webových aplikácií využíva databázy a tým pádom vzniká problém autentifikácie používateľa voči databáze. Riešenia týchto a mnohých iných problémov musia v ASP/PHP/JSP riešiť vývojári webových aplikácií. U ASP.NET sú príslušné bloky, ktoré riešia naznačené problémy súčasťou infraštruktúry.

KAPITOLA 2:

Základy architektúry ASP.NET

Napriek tomu, že sa jedná o pomerne novú technológiu, má už sa sebou určitú krátku históriu. Prvýkrát sa technológia ASP.NET, vtedy ešte pod označením **ASP+** objavila pred vyše dvomi rokmi ako súčasť prvej beta verzie vývojového prostredia Visual Studio .NET. V ďalších beta verziách a samozrejme aj vo finálnej verzii Visual Studia .NET sa označenie tejto technológie ustálilo na ASP.NET. Pre svoje výhodné vlastnosti technológia ASP.NET zrejme postupne nahradí klasické ASP stránky. Hlavný rozdiel medzi technológiou ASP a ASP.NET je ten, že kódy na stránkach ASP.NET sú kompilované. Tým sa odstránila nutnosť analýzy a interpretácie jednotlivých riadkov pri každom prístupe klienta. Vznikne kompilovaný kód, ktorý je samozrejme oveľa rýchlejší. Klasické ASP stránky pomocou skriptov na strane servera priamo generujú HTML stránky, ktorá sa zašlú klientovi. Stránky sa interpretujú od začiatku do konca bez možnosti ošetrovať vzniknuté stavy a udalosti. ASP.NET používajú rovnakú technológiu okien, dialógov a formulárov ako bežné Windows aplikácie. K jednotlivým vizuálnym prvkom sa viažu procedúry pre ošetrenie stavov a udalostí.

Komerčný pohľad na ASP.NET

Možno ešte skôr než technologické hľadisko je v tomto prípade potrebné rozobrať hľadisko komerčné. Ak sa totiž rozhodujeme o prípadnej migrácii na novú softvérovú technológiu, zaujíma nás samozrejme aj cena, ktorú budeme musieť za túto migráciu zaplatiť. Technológia ASP.NET je v podstate **zadarmo**, samozrejme ak nepočítame cenu operačného systému Windows. Podobne tomu bolo aj pri ASP stránkach. Technológia ASP.NET sa právom aj neprávom spája s novým vývojovým prostredím Microsoftu s názvom **Visual Studio NET**, ktoré samozrejme zadarmo nie je, ale pre vývoj ASP.NET aplikácií ho nutne nepotrebujeme. Pre najjednoduchšie aplikácie postačí dokonca bežný textový editor, napríklad Notepad. Pre väčšinu projektov hlavne pre sféru hobby je ideálne voľne šíriteľné vývojové prostredie Web Matrix. Samozrejme ak Visual Studio NET máme k dispozícii, vývoj hlavne náročných podnikových aplikácií ASP.NET bude podstatne efektívnejší.

Čo budeme potrebovať pre prevádzku ASP.NET aplikácie

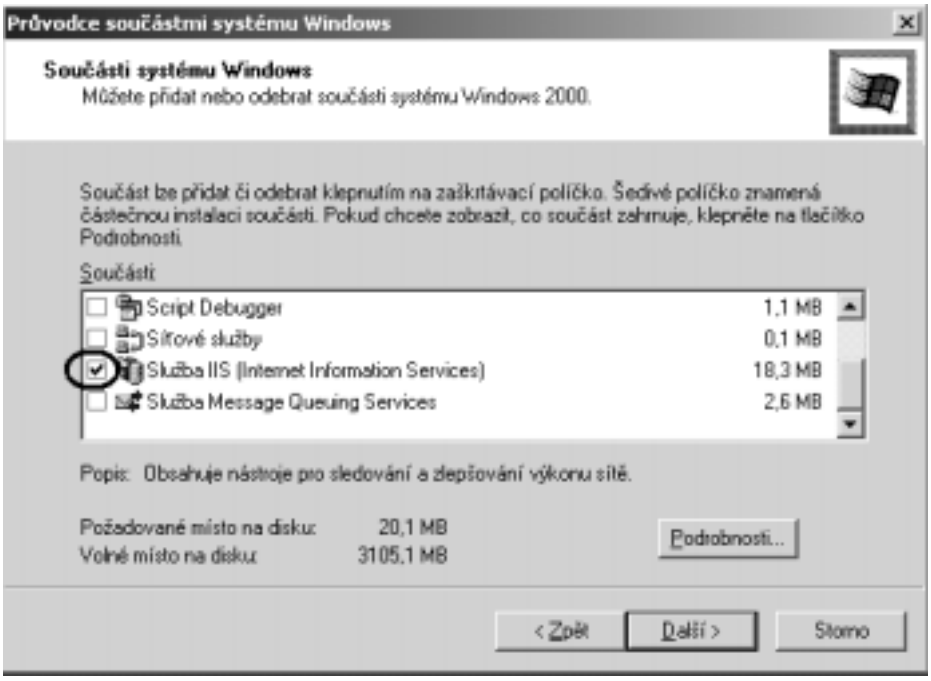
Predpokladajme, že máme zatiaľ nainštalovaný len operačný systém, napríklad Windows 98, Windows 2000 Professional, Windows XP, Windows 2003 Server..... Pre základné fungovanie ASP.NET stránok potrebujeme webový server a technologickú platformu .NET Framework.

Webový Server

Väčšina operačných systémov pri implicitnej inštalácii Internet Information Server nenainštaluje, nevie sa totiž či ho budeme potrebovať a nezabezpečený webový server na lokálnom počítači pripojenom do siete predsa len predstavuje určité bezpečnostné riziko. Ak chceme pracovať s ASP.NET stránkami, prípadne na svojom serveri takéto aplikácie prevádzkovať, musíme IIS nainštalovať.

Inštalácia pre Windows NT/2000/XP

Nasledujúci postup platí pre Windows NT/2000/XP. Pre inštaláciu potrebujeme inštalačné CD operačného systému. V ponuke menu Nastavenia - Ovládacie panely - Pridať alebo odobrať programy aktivujeme tlačidlo „Pridať alebo odobrať súčasti Window“. Inštalácia spočíva v zaškrtnutí voľby Služba IIS (Internet Information Services).



Inštalácia Internet Information Servera

Po nainštalovaní a spustení služby IIS potrebujeme vedieť dve základné veci. Aká je naša URL adresa a do ktorého adresára je potrebné umiestniť súbory ktoré tvoria našu webovú aplikáciu. Obidva parametre môžeme pri inštalácii ovplyvniť, ak však do procesu inštalácie nezasahujeme (odporúčame začiatočníkom), tieto parametre nám prezradí dialógové okno Internet Information Servera, napríklad:

Pracovné adresáre Internet Information Servera

Domovský adresár bol v našom prípade: C:\inetpub\wwwroot. Na lokálnom počítači je implicitná adresa http://localhost Najjednoduchšie je umiestniť APS.NET stránku, teda napríklad súbor s príponou ASPX buď priamo do adresára C:\inetpub\wwwroot, prípadne v ňom vytvoriť ďalší podadresár.

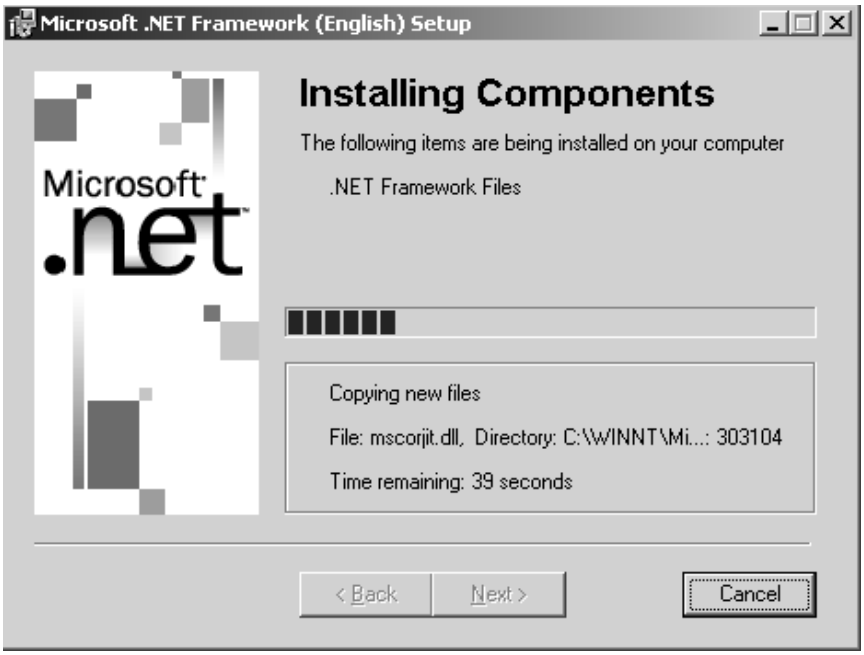
Inštalácia pre Windows 98/Millennium

Pre tieto operačné systémy môžeme nainštalovať zjednodušenú verziu IIS s označením Personal Web Server (nájdeme ju na inštalačnom CD Windows spravidla v adresári addons). Existuje však aj jednoduchšie riešenie. Pre vývoj ASP.NET aplikácií vo vývojovom prostredí Web Matrix IIS nutne nepotrebujeme, Web Matrix totiž má jednoduchý webový server (listener) pre ladiace účely. Na tomto mieste je potrebné znovu zdôrazniť, že spomínaný server, ktorý je súčasťou vývojového prostredia Web Matrix je určený skutočne len pre vývoj a ladenie aplikácie, teda pre prístup jedného, alebo niekoľko málo klientov. Ono vlastne rodina operačných systémov Windows 98 a Millennium nie je pre serverové aplikácie ani určená.

Microsoft .NET Framework

Platforma .NET Framework. je spojená s operačným systémom Windows, takže pri vymenovávaní komponentov jednotlivých konfigurácií budeme predpokladať už nainštalovaný operačný systém Windows (2000 alebo XP) a Internet Information Server (IIS). Táto konfigurácia bola úplne postačujúca pre prevádzkovanie aplikácií založených na ASP stránkach, pretože ASP engine (súbor ASP.DLL) je súčasťou operačného systému Windows.
Pre fungovanie ASP.NET aplikácie nevyhnutne potrebujeme technologickú platformu .NET Framework. Inštalačný súbor (20 MB) môžeme zdarma stiahnuť z webu (http://msdn.microsoft.com/netframework/downloads/redist.aspx), prípadne môžeme .NET Framework nainštalovať z priloženého CD. Pre niektoré staršie operačné systémy bude potrebné najskôr nainštalovať novšiu verziu Internet Explorera. .NET Framework je v princípe sada objektov použiteľných na strane klienta aj servera. Predpokladá sa, že vývojári budú vyvíjať aj komponenty pre všeobecné použitie. Ušetrí sa tým veľa námahy a času pri vývoji ďalších aplikácií. Kostru novej aplikácie môžeme vybudovať na existujúcich komponentoch, ktoré je možné získať, prípadne kúpiť. Ak napríklad navrhujeme aplikáciu typu klient - server ktorá pracuje s údajmi v databázach, potrebujeme zakaždým sformulovať SQL dotaz do databázy pomocou interakcie s klientom prostredníctvom vhodného dialógu. Potom musíme klientovi vhodnou formou zobraziť údaje.

Údaje môžu obsahovať text, tabuľky, obchodnú grafiku alebo obrázky. Takže doterajší kód na ASP stránkach, ktoré zobrazovali výsledky z databáz sa priamo hemžil programovými konštrukciami Response.Write, výpisom tabuliek, stránkovaním databázových výpisov a podobne.
Ako minimálnu HW konfiguráciu odporúča Microsoft počítač s procesorom Pentium II 300 MHz, a pamäťou minimálne 128 MB RAM.
Samotná inštalácia platformy .NET Framework spočíva v spustení inštalačného súboru dotnetredist.exe. Po súhlase s licenčnými podmienkami zadáme meno adresára, do ktorého sa inštalačný súbor rozbalí. U starších operačných systémov sa najskôr automaticky upraví komponenta Microsoft Windows Installer na novšiu verziu. Potom inštalácia .NET Frameworku pokračuje.

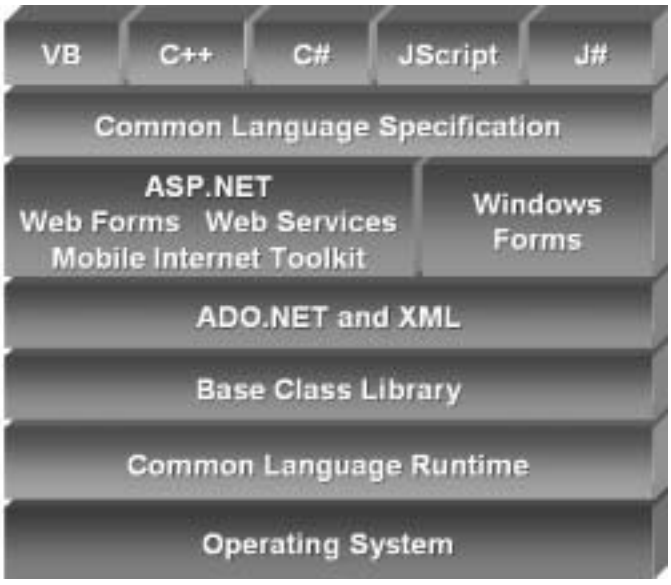


Inštalácia .NET Framework

Po úspešnom nainštalovaní je potrebné reštartovať počítač, aby sa prostredie .NET Framework zaviedlo do operačného systému.

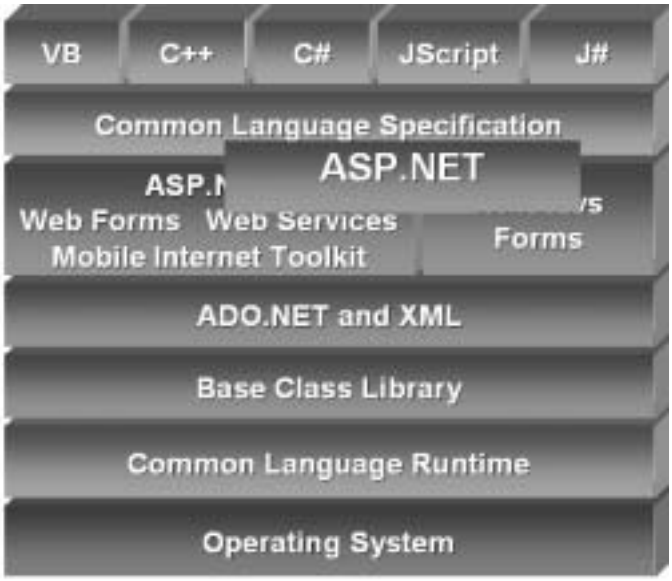
Architektúra ASP.NET

Základný model architektúry ASP.NET vychádza samozrejme z architektúry .NET Frameworku. Táto, ako vidíme z obrázku, sa skladá z viacerých vrstiev. Pri letmom pohľade na najvrchnejšiu vrstvu vidíme niekoľko najpoužívanejších .NET programovacích jazykov, presnejšie jazykov Visual Basic, Visual C++, Visual C#, JScript, Visual J#, ktoré dodáva Microsoft.



Architektúra.NET Framework

Okrem toho pre platformu .NET existuje ďalších 20-30 programovacích jazykov od tretích strán. Napr. ObjectPascal, Fortran, Cobol či Eiffel... Jedná sa o programovacie jazyky, ktoré sa neprekladajú do natívneho kódu konkrétneho procesora (s výnimkou C++), ale do tzv. riadeného alebo v pôvodnej terminológie managed kódu, čiže do akéhosi medzikódu, ktorý sa spracuje na úrovni vrstvy CLR (Common Language Runtime). Pod pojmom spracovanie rozumieme v tomto prípade kompiláciu až v okamihu spustenia, tzv JIT (Just In Time) kompiláciu. Ak by sme mali v tejto architektonickej schéme charakterizovať miesto bloku ASP.NET, vypadalo by to v schéme .NET Frameworku nasledovne:



Začlenenie ASP.NET

Na ďalšej schéme môžeme sledovať cestu požiadavky klienta. Požiadavka klienta postupuje cez jednotlivé vrstvy zdola nahor, pričom handlerov pre vetvenie kódu k jednotlivým blokom je takmer až na samom konci tohoto procesu.

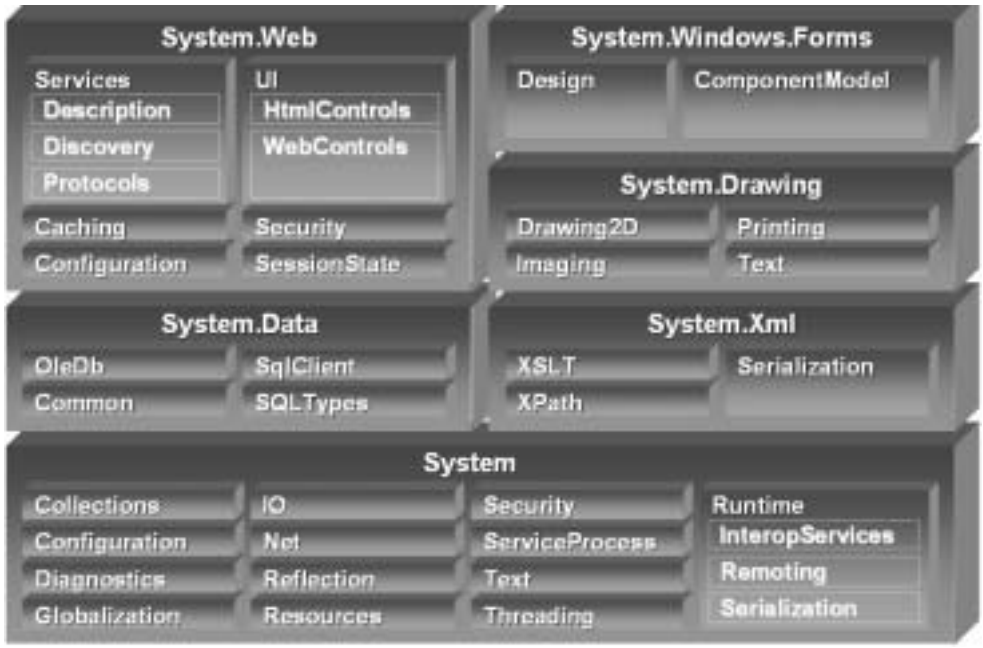


Schéma spracovania požiadavky

Požiadavka postupuje od Internet Information Servera cez aplikačnú vrstvu až k vrstve handlerov, pričom môže dôjsť k viacerým vetveniam podľa nastavenia konfigurácie. Klasické ASP stránky postúpia k modulu ASP ISAPI, čím je vlastne zabezpečená spätná kompatibilita. Súbor GLOBAL.ASAX je obdobou súboru GLOBAL.ASA, ktorý poznáme z ASP stránok. Na úrovni aplikačnej vrstve sa obvykle rieši cachovanie, session, autentifikácia a autorizácia klientov. Na úrovni handlerov sa spracovávajú aplikačné formuláre, webové služby a podobne. Z obidvoch schém vidíme, že sa jedná o otvorený systém, kam môžeme integrovať nové programovacie jazyky, prípadne nové handlery.

Pre ASP.NET stránky je vyčlenená súborová prípona **ASPX** (ASP stránky mali príponu súborov ASP). Tieto stránky jednak zapúzdrujú vizuálne elementy pre komunikáciu s používateľom ktoré generujú príslušné udalosti, napríklad reakciu na kliknutie na tlačidlo a podobne. Tým je dosiahnuté nelineárne, udalosťami riadené spracovávanie kódu.

Vymenujeme základné HTTP handlery, to jest bloky v ktorých sa spracováva obsah kódu zo súborov jednotlivých typov.
Web Forms (*.aspx) - System.Web.UI.PageHandler
Webové služby (*.asmx) - System.Web.Services.Protocols.WebServiceHandlerFactory
Tracing (trace.axd) - System.Web.Handlers.Tracehandler
.NET Remoting (*.rem, *.soap) - System.Runtime.Remoting.Channels.Http.HttpRemotingHandlerFactory
Súbory zo zákazom prístupu (asax, ascx, config, vb, vbproj, cs, csproj, webinfo, asp, licx, resx, resources) - System.Web.HttpForbiddenHandler
System.Web.StaticFileHandler (GET, HEAD) - System.Web.HttpMethodNotAllowedHandler (zvyšok)

Hlavné výhody technológie ASP.NET
- rýchlosť aplikácii, pretože všetok kód je predkompilovaný. ASP.NET aplikácie sú niekoľkokrát rýchlejšie než klasické ASP aplikácie
- striktné objektové programovanie
- využitie klasických programovacích jazykov, napríklad C#, Visual Basic, JScript...
- striktné oddelenie jednotlivých vrstiev aplikácie. Vývoj aplikačného kódu môže byť dôsledne oddelený od dizajnového a grafického návrhu HTML stránok
- vývoj je oveľa efektívnejší, hlavne vďaka tomu, že nie je potrebné neustále programovať tie isté záležitosti ako napríklad autentifikáciu a autorizáciu klientov, konektivitu k databázam, ošetrovanie najčastejšie sa vyskytujúcich chybových stavov, stránkovaný výpis údajov z databáz, .. ale veď to ako vývojári sami z vlastnej praxe poznáte
- pri zmenách aplikácie sa na rozdiel od technológie ISAPI nevyžadujú žiadne reštarty, takže webový projekt môžeme veľmi jednoducho spravovať na diaľku.
- Zmenu aplikácie môžeme realizovať jednoduchým nakopírovaním súborov bez zastavovania webového servera.
- Konfiguračné údaje sú uložené v XML súboroch

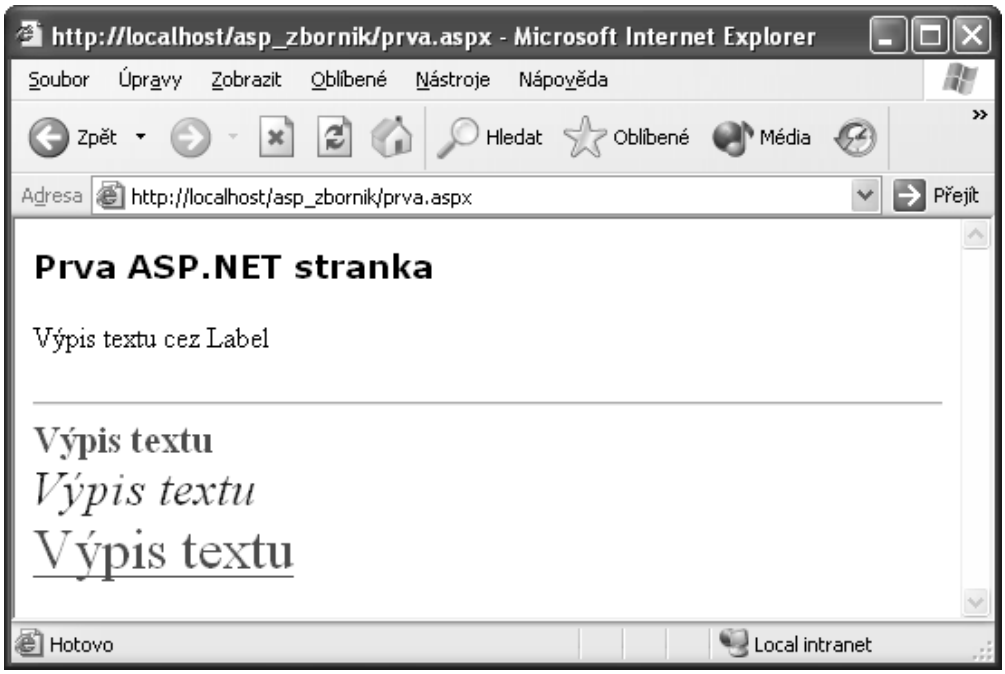
Nevýhody technológie ASP.NET
Podobne ako tomu bolo pri technológii ASP kedy sme jej hlavné nevýhody vymenovali na konci predchádzajúcej kapitoly, aj ASP.NET má určite nejaké nevýhody a môžete sa spoľahnúť, že budú vymenované pri propagovaní novej nástupníckej technológie.

Prvá jednoduchá ASP.NET aplikácia
Aby sme neporušili známu tradíciu príkladov pre programovacie jazyky, bude prvá ASP.NET aplikácia typu Hello World, teda aplikácia ktorá vypíše na HTML stránku jednoduchý text. Zatiaľ sa nebudeme zamýšľať nad syntaxou jednotlivých príkazov a programových konštrukcií. Ide len o to, umiestniť prvý skript do správneho adresára a spustiť ho.

```
<%@ Page Language="C#" %>
<html>
    <body>
       <h3><font face="Verdana">Prva ASP.NET stranka</font></h3>
       Výpis textu cez Label
       <p>
       <hr>
       <asp:label id="Message1" font-size="16" font-bold="true" forecolor="red"
                  runat=server>Výpis textu</asp:label>
       <br>
```

```
        <asp:label id="Message2" font-size="20" font-italic="true" forecolor="blue"
                   runat=server>Výpis textu</asp:label>
        <br>
        <asp:label id="Message3" font-size="24" font-underline="true" forecolor="green"
                   runat=server>Výpis textu</asp:label>
    </body>
</html>
```

Tento kód umiestnime do súboru **prva.asp**. Súbor je v pracovnom adresári webového servera, v našom prípade **C:\Inetpub\wwwroot** a aby to bolo prehľadné v tomto adresári sme vytvorili podadresár **ASP_zbornik**. Kompletný názov súboru aj s cestou bude **C:\Inetpub\wwwroot\ASP_zbornik\prva.asp**
Skript bol trochu neprehľadný, ale výsledok, ktorý sa zobrazí v okne klientovho prehliadača webových stránok sa po zadaní URL adresy http://localhost/asp_zbornik/prva.aspx je celkom zrozumiteľný:



ASP.NET aplikácie typu Hello World v klientskom okne

Všimnime si hlavne deklaráciu `runat=server`, pomocou ktorej špecifikujeme, že príslušná časť kódu sa vykoná na serveri. Určite namietnete, že rovnaký efekt by sme dosiahli aj pomocou kódu na HTML stránke, interpretovanej u klienta bez ASP.NET kódu a dokonca bez použitia akéhokoľvek kódu, len s využitím HTML. Áno je to pravda, dokonca si ten kód môžete nechať zobraziť v kontextovom menu prehliadača *„Zobraziť zdrojový kód“*.

```
<html>
    <body>
        <h3><font face="Verdana">Prva ASP.NET stranka</font></h3>
        Výpis textu cez Label
        <p>
        <hr>
        <span id="Message1" style="color:Red;font-size:16pt;font-weight:bold;">Výpis textu</span>
        <br>
        <span id="Message2" style="color:Blue;font-size:20pt;font-style:italic;">Výpis
textu</span>
        <br>
        <span id="Message3" style="color:Green;font-size:24pt;text-decoration:underline;">Výpis
textu</span>
    </body>
</html>
```

Základy syntaxe ASP.NET

Znalcov iných serverových skriptových systémov pravdepodobne kód prvej jednoduchej ASP.NET aplikácie mierne zaskočí. Budú hľadať oddeľovače kódu použitého programovacieho jazyka od HTML kódu. Úvodné riadky kódu ich celkom uspokoja

```
<%@ Page Language="C#" %>
<html>
...
```

Príkazy alebo riadky programového kódu môžu byť podobne ako u ASP stránok od zvyšku HTML stránky oddelené párovými znakmi <% ... %>. Tento spôsob však nie je z hľadiska behu aplikácie ideálny, nakoľko kód uzavretý v týchto párových znakoch sa nekompiluje. Preto kód radšej uzatvárame do samostatných blokov, ktoré sa kompilujú. Pre úplne prvé pokusy s technológiou ASP.NET, kedy napíšeme jednoduchý kód napríklad pomocou textového editora môže byť tento spôsob výhodný z hľadiska používateľa, nakoľko nemusíme spúšťať riadkový kompilátor pre kompiláciu blokov kódu.

Skriptový jazyk pre ASP.NET stránky sa nastavuje pomocou príkazu: **<%@ Page Language=jazyk %>** V našom príklade sú aj iné časti kódu, komponenta LABEL. Táto časť právom pripomína syntax XML súboru.

```
<asp:label
    id="Message1" font-size="16" font-bold="true" forecolor="red" runat=server>Výpis textu
</asp:label>
```

Objekty

Technologická platforma .NET Framework je prísne objektovo orientovaná. Preto sa budeme v programovom kóde stretávať s rôznymi triedami a priestormi mien (namespace), využívať ich metódy a podobne. Na svoje si teda prídu aj priaznivci objektovo orientovaného programovania, a prísne štrukturovane využívať triedy, objekty, dedičnosť, polymorfizmus... Ale ani sviatoční programátori, programátori, ktorí sa viac sústredia na aplikačnú logiku než na prísnu objektovú štruktúru nemusia zúfať. Stačí sa len naučiť využívať základné triedy, teda volať ich metódy a nastavovať parametre alebo inak povedané vlastnosti objektov.

Programovacie jazyky pre ASP.NET serverové stránky

Paleta programovacích jazykov pre ASP.NET bola rozšírená kvantitatívne aj kvalitatívne. Kvantitatívne rozšírenie spočíva v použití „plných" programovacích jazykov **Visual Basic.NET**, **JScript.NET** a **C#**. Kvalitatívna zmena spočíva v tom, že kód napísaný v spomínaných programovacích jazykoch je kompilovaný najskôr do IL kódu a následne do natívneho kódu procesora webového servera.

Programovací jazyk je deklarovaný v záhlaví stránky. Visual Basic (úplná verzia nahradila VBScript) a JScript už poznáme z ASP stránok. Programovací jazyk C# (hudobníci vedia, že symbol C# znamená notu C zvýšenú o poltón) vychádza koncepčne z jazyka C++. Je to jazyk jednoduchý, typovo bezpečný a samozrejme objektovo orientovaný. Pre prvé zoznámenie so spomínanými troma programovacími jazykmi ukážeme jednoduché príklady, ktoré budú obsahovať komentáre, deklarácie premenných, priradenie hodnoty a výpis obsahu premenných do kontextu HTML stránky

Príklad v jazyku Visual Basic *(súbor prvaVB.aspx)*

```
<%@ Page Language="VisualBasic" %>

<%
  'jednoriadkovy komentar

  'deklaracia premennych
   Dim nX As Integer
   Dim sNapis As String

  'priradenie hodnoty premennych
   sNapis = "Hello World"
   nX = 1

  'Vypis
   Response.Write(sNapis & " - " & nX)
%>
```

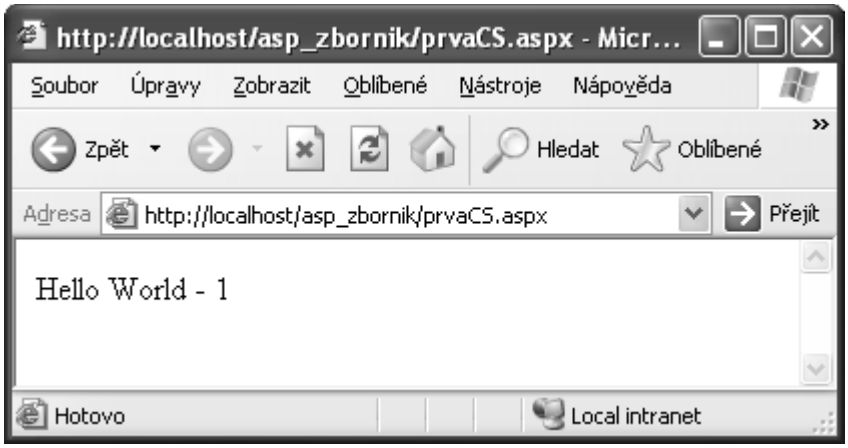
Príklad v jazyku JScript *(súbor prvaJS.aspx)*

```
<%@ Page Language="JScript" %>

<%
 //jednoriadkovy komentar
 /*viacriadkovy
  komentar */

//deklaracia premennych
  var nX : int;
  var sNapis : String;

//priradenie hodnoty premennych
  sNapis = "Hello World";
  nX = 1;

//Vypis
  Response.Write(sNapis + " - " + nX);
%>
```

Príklad v jazyku C# *(súbor prvaCS.aspx)*

```
<%@ Page Language="C#" %>
<%
 //jednoriadkovy komentar
 /*viacriadkovy
  komentar */

//deklaracia premennych
  int nX;
  String sNapis;

//priradenie hodnoty premennych
  sNapis = "Hello World";
  nX = 1;

//Vypis
  Response.Write(sNapis + " - " + nX);
%>
```

Klientovi sa zobrazí HTML stránka v tvare:



Jednoduchá ASP.NET aplikácia

pričom kód HTML stránky, ktorý sa vygeneruje na základe ASP.NET kódu a pošle klientovi bude proste

```
Hello World - 1
```

Vidíme, že ak sa klient začne zaujímať o zdrojový kód HTML stránky, síce ho zistí, ale nič viac. Nemá ani len tušenia, akým spôsobom textový reťazec "`Hello World - 1`" vlastne vznikol.
To, že pri pohľade na takýto kód by sa priaznivcom „čistoty" kódu obzvlášť z radov znalcov XML robilo nevoľno je samozrejme naša chyba. Prehliadaču to nevadí, no správne by ten HTML kód mal obsahovať aspoň počiatočné a koncové HTML tagy. My sme ich vynechali pre názornosť

```
<html>
     <body>
        Hello World - 1
     </body>
</html>
```

Potom by náš kód (už bez zbytočných komentárov) v programovacom jazyku C# bol v tvare

Príklad v jazyku C# *(súbor prvaCS1.aspx)*

```
<%@ Page Language="C#" %>
<html>
     <body>
     <%
         //deklaracia premennych
           int nX;
           String sNapis;

         //priradenie hodnoty premennych
           sNapis = "Hello World";
           nX = 1;

        //Vypis
          Response.Write(sNapis + " - " + nX);
     %>
     </body>
</html>
```

Z týchto ukážok je jasná syntax jednoriadkového a viacriadkového komentára deklarácia obsahu premenných a výpisy. Nebudeme to komentovať, všetci vidia aká náročná by asi bola migrácia na iný programovací jazyk. Pri zložitejších programových konštrukciách rozdiely medzi kódom v jednotlivých programovacích jazykoch budú samozrejme väčšie.

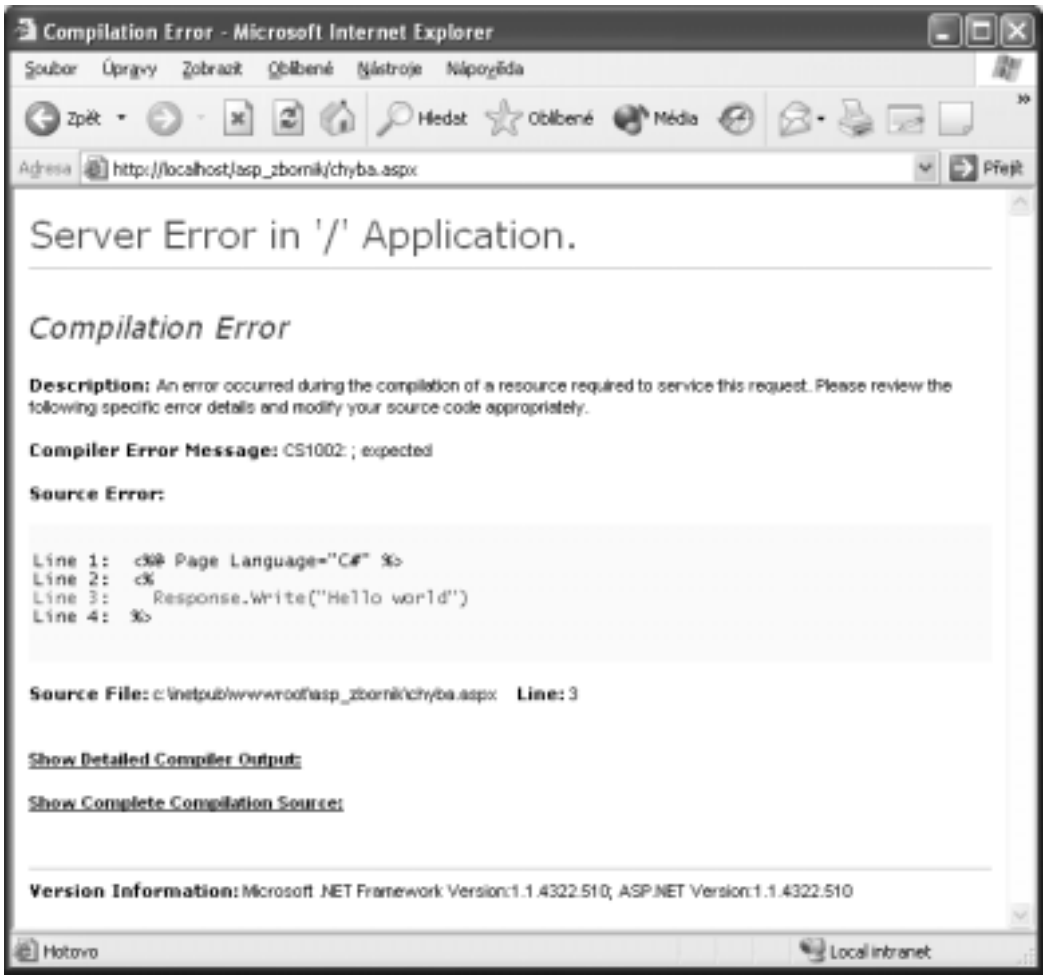
Výpis textu a prvé kroky pri ladení programu
Príklad typu Hello World sa neuvádza takmer vo všetkých učebniciach programovania ako prvý úplne náhodne. Vedieť vypísať v danom programovacom jazyku na obrazovku vhodný text je pomerne silný nástroj. Veď čo by sme mali z toho, keby sme pomocou zložitého programu dokázali realizovať náročný výpočet, keby jeho výsledok zostal navždy (z technického hľadiska do vypnutia prúdu) ležať v hlbinách procesora alebo pamäti RAM. Výpisy sa uplatnia aj pri ladení, hlavne v prípadoch ak programovací jazyk nedisponuje dobrým vývojovým a ladiacim prostredím. Pre výpis textu sa používa príkaz `Response.Write`.

Aby sme ukázali aj domovskú stránku začínajúcich, prípadne nepozorných ASP.NET programátorov, ukážeme reakciu .NET servera na chybný kód. Napríklad zabudneme stredník (bodkočiarku) za príkazom v jazyku C#.

Príklad v jazyku C# *(súbor chyba.aspx)*

```
<%@ Page Language="C#" %>
<%
  Response.Write("Hello world")
%>
```

Zobrazí sa stránka s popisom chybového stavu



Stránka s popisom chybového stavu

Chybové hlásenie je veľmi poučné a naznačuje nám niekoľko faktov. Už popis chyby

Description: An error occurred during the compilation of a resource required to service this request. Please review the following specific error details and modify your source code appropriately.

Compiler Error Message: CS1002: ; expected
Line 1: <%@ Page Language="C#" %>
Line 2: <%
Line 3: Response.Write("Hello world")
Line 4: %>

svedčí o tom, že ku chybe došlo pri kompilácii, teda je to nepriamy dôkaz nášho tvrdenia, že kód na ASP stránkach je kompilovaný. Vynikajúce bude aj ladenie. V chybovom hlásení nám kompilátor jasne odkazuje, že nám v kóde chýba bodkočiarka a dokonca nám červenou farbou naznačí aj v ktorom riadku. Po kliknutí na odkaz: **Show Detailed Compiler Output**:

sa zobrazí na HTML stránke kompletný výpis okna kompilátora. (zhruba tak ako by sme ho videli v ladiacom okne Visual Studia .NET). Po zhliadnutí tohoto ladiaceho výpisu sa pravdepodobne už vývoj ASP.NET v minimálnej konfigurácii pomocou textového editora nebude zdať až taký nereálny.

C:\WINDOWS\system32> "c:\windows\microsoft.net\framework\v1.1.4322\csc.exe" /t:library /utf8output
/R:"c:\windows\assembly\gac\system.web\1.0.5000.0__b03f5f7f11d50a3a\system.web.dll"
/R:"c:\windows\assembly\gac\system\1.0.5000.0__b77a5c561934e089\system.dll"
/R:"c:\windows\assembly\gac\system.web.services\1.0.5000.0__b03f5f7f11d50a3a\system.web.services.dll"
/R:"c:\windows\assembly\gac\system.xml\1.0.5000.0__b77a5c561934e089\system.xml.dll"

```
/R:"c:\windows\assembly\gac\system.web.mobile\1.0.5000.0__b03f5f7f11d50a3a\system.web.mobile.dll"
/R:"c:\windows\assembly\gac\system.drawing\1.0.5000.0__b03f5f7f11d50a3a\system.drawing.dll"
/R:"c:\windows\assembly\gac\system.data\1.0.5000.0__b77a5c561934e089\system.data.dll"
/R:"c:\windows\assembly\gac\system.enterpriseservices\1.0.5000.0__b03f5f7f11d50a3a\system.enterpriseservic
es.dll" /R:"c:\windows\microsoft.net\framework\v1.1.4322\mscorlib.dll"
/out:"C:\WINDOWS\Microsoft.NET\Framework\v1.1.4322\Temporary ASP.NET
Files\root\050a1654\638a5594\d0phtzim.dll" /debug- /optimize+ /warnaserror /w:1
"C:\WINDOWS\Microsoft.NET\Framework\v1.1.4322\Temporary ASP.NET
Files\root\050a1654\638a5594\d0phtzim.0.cs"


Microsoft (R) Visual C# .NET Compiler version 7.10.2292.4
for Microsoft (R) .NET Framework version 1.1.4322
Copyright (C) Microsoft Corporation 2001-2002. All rights reserved.

c:\inetpub\wwwroot\asp_zbornik\chyba.aspx(3,32): error CS1002: ; expected
```

Ak klikneme na odkaz: **Show Complete Compilation Output**: získame kompletný výpis zdrojového kódu, ktorý vstupoval do kompilátora.

```
Line 1:     //------------------------------------------------------------------------------
Line 2:     // <autogenerated>
Line 3:     //      This code was generated by a tool.
Line 4:     //      Runtime Version: 1.1.4322.510
Line 5:     //
Line 6:     //      Changes to this file may cause incorrect behavior and will be lost if
Line 7:     //      the code is regenerated.
Line 8:     // </autogenerated>
Line 9:     //------------------------------------------------------------------------------
Line 10:
Line 11:    namespace ASP {
Line 12:        using System;
Line 13:        using System.Collections;
Line 14:        using System.Collections.Specialized;
Line 15:        using System.Configuration;
Line 16:        using System.Text;
Line 17:        using System.Text.RegularExpressions;
Line 18:        using System.Web;
Line 19:        using System.Web.Caching;
Line 20:        using System.Web.SessionState;
Line 21:        using System.Web.Security;
Line 22:        using System.Web.UI;
Line 23:        using System.Web.UI.WebControls;
Line 24:        using System.Web.UI.HtmlControls;
Line 25:
Line 26:
Line 27:        public class chyba_aspx : System.Web.UI.Page,
System.Web.SessionState.IRequiresSessionState {
Line 28:
Line 29:            private static bool __initialized = false;
Line 30:
Line 31:            private static System.Collections.ArrayList __fileDependencies;
Line 32:
Line 33:            public chyba_aspx() {
Line 34:                System.Collections.ArrayList dependencies;
Line 35:                if ((ASP.chyba_aspx.__initialized == false)) {
Line 36:                    dependencies = new System.Collections.ArrayList();
Line 37:                    dependencies.Add("c:\\inetpub\\wwwroot\\asp_zbornik\\chyba.aspx");
Line 38:                    ASP.chyba_aspx.__fileDependencies = dependencies;
```

```
Line 39:                     ASP.chyba_aspx.__initialized = true;
Line 40:                 }
Line 41:             }
Line 42:
Line 43:             protected override bool SupportAutoEvents {
Line 44:                 get {
Line 45:                     return false;
Line 46:                 }
Line 47:             }
Line 48:
Line 49:             protected System.Web.HttpApplication ApplicationInstance {
Line 50:                 get {
Line 51:                     return
((System.Web.HttpApplication)(this.Context.ApplicationInstance));
Line 52:                 }
Line 53:             }
Line 54:
Line 55:             public override string TemplateSourceDirectory {
Line 56:                 get {
Line 57:                     return "/asp_zbornik";
Line 58:                 }
Line 59:             }
Line 60:
Line 61:             private void __BuildControlTree(System.Web.UI.Control __ctrl) {
Line 62:                 __ctrl.SetRenderMethodDelegate(new
System.Web.UI.RenderMethod(this.__Render__control1));
Line 63:             }
Line 64:
Line 65:             private void __Render__control1(System.Web.UI.HtmlTextWriter __output,
System.Web.UI.Control parameterContainer) {
Line 66:
Line 67:                 #line 2 "c:\inetpub\wwwroot\asp_zbornik\chyba.aspx"
Line 68:
Line 69:     Response.Write("Hello world")
Line 70:
Line 71:
Line 72:                 #line default
Line 73:                 #line hidden
Line 74:             }
Line 75:
Line 76:             protected override void FrameworkInitialize() {
Line 77:                 this.__BuildControlTree(this);
Line 78:                 this.FileDependencies = ASP.chyba_aspx.__fileDependencies;
Line 79:                 this.EnableViewStateMac = true;
Line 80:                 this.Request.ValidateInput();
Line 81:             }
Line 82:
Line 83:             public override int GetTypeHashCode() {
Line 84:                 return 5381;
Line 85:             }
Line 86:         }
Line 87:     }
```

Vidíme, že z 87 riadkov kódu, len riadok 69 sme museli napísať sami. Všetko ostatné zariadila platforma ASP.NET.

Cyklus a podmienka

V našom predstavovaní syntaxe programovacích jazykov pre ASP.NET sa zameriame ďalšie veľmi potrebné a často používané programové konštrukcie, na cyklus a podmienku.

Príklad v jazyku Visual Basic *(súbor druhaVB.aspx)*

```
<%@ Page Language="VisualBasic" %>
<%
  'cyklus
   Dim nI As Integer
   For nI = 0 To 3
     Response.Write(nI &" iteracia<BR>")
   Next

'podmienka
   If  (nI = 4)
     Response.Write("Cyklus ukonceny")
   End If
%>
```

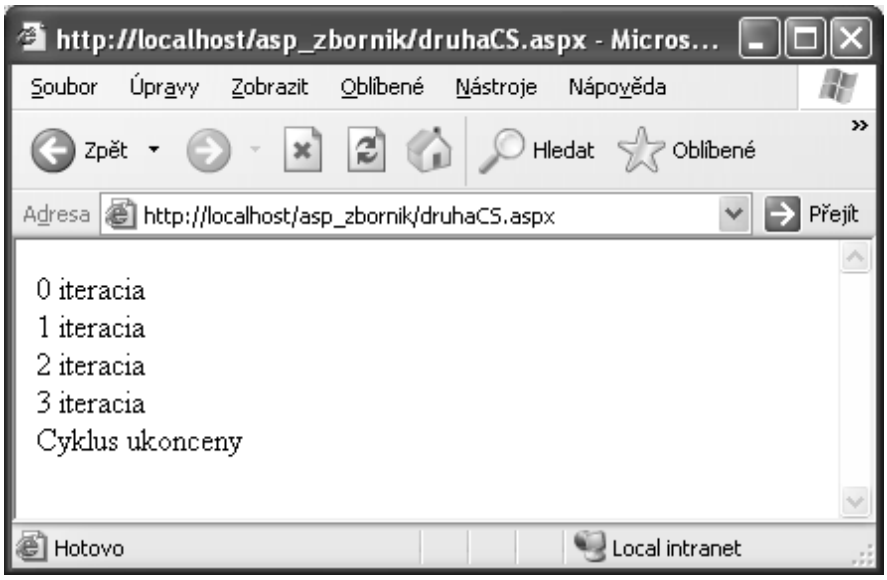
Príklad v jazyku JScript *(súbor druhaJS.aspx)*

```
<%@ Page Language="JScript" %>
<%
 //cyklus
   for (var nI : int = 0; nI <= 3; nI++)
   {
     Response.Write(nI + " iteracia<BR>");
   }

 //podmienka
   if  (nI == 4)
     Response.Write("Cyklus ukonceny");
%>
```

Príklad v jazyku C# *(súbor druhaCS.aspx)*

```
<%@ Page Language="C#" %>
<%
   int nI;
 //cyklus
   for (nI = 0; nI <= 3; nI++)
   {
     Response.Write(nI + " iteracia<BR>");
   }

 //podmienka
   if  (nI == 4)
     Response.Write("Cyklus ukonceny");
%>
```

Klientovi sa zobrazí HTML stránka v tvare:



Príklad pre cyklus a podmienku

Ak by sme mali skúmať, ktoré programátorské konštrukcie sa pri vývoji webových aplikácii používajú najčastejšie, pravdepodobne by sme dospeli k záveru, že spájanie alebo zlučovanie obsahu reťazcov. Preto sa naša tretia séria príkladov bude venovať práve tejto problematike. Aby sme ukázali aj spôsob práce s objektami a triedami, ukážeme okrem klasického spájania reťazcov pomocou operátorov ukážeme aj spájanie pomocou triedy StringBuilder. Je to trieda pre manipuláciu s reťazcami. V tejto časti príkladu najskôr pomocou konštruktoru vytvoríme novú inštanciu triedy StringBuilder, keďže sme nezadali parameter, bude trieda po vytvorení zapúzdrovať prázdny reťazec. Postupne pomocou metódy Append budeme pridávať jednotlivé časti reťazca.

Príklad v jazyku Visual Basic (*súbor textVB.aspx)*

```
<%@ Page Language="VisualBasic" %>
<%

  'spajanie retazcov
   Dim s1, s2 As String
   s2 = "Dobrý"
   s2 &= " deň"
   s1 = s2 & " čitatelia, "

  'spajanie pomocou triedy StringBuilder
   Dim s3 As New StringBuilder()
   s3.Append("dobrý")
   s3.Append("  deň")
   s3.Append(" vývojári...")

   Response.Write (s1)
   Response.Write (s3)
%>
```

Príklad v jazyku JScript *(súbor textJS.aspx)*

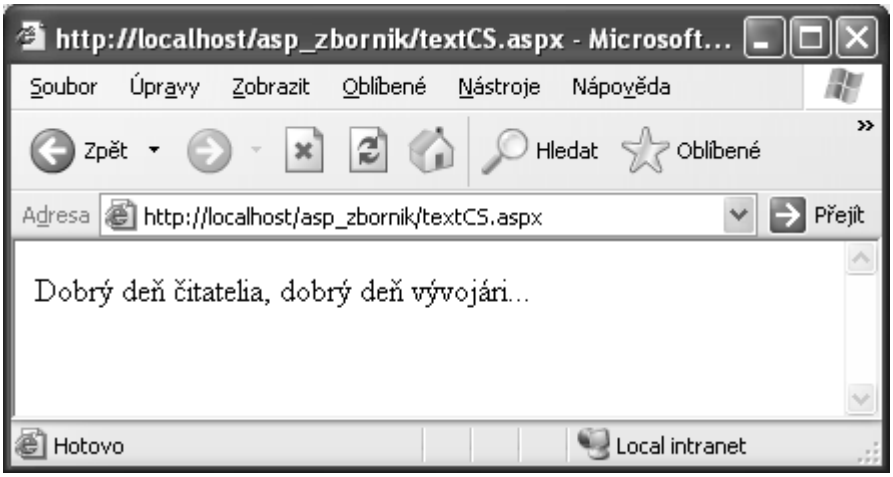
```
<%@ Page Language="JScript" %>
<%
//spajanie retazcov
   var s1 : String;
```

```
  var s2 : String = "Dobrý";
   s2 += " deň";
   s1 = s2 + " čitatelia, ";

//spajanie pomocou triedy StringBuilder
   var s3:StringBuilder = new StringBuilder();
   s3.Append("dobrý");
   s3.Append(" deň");
   s3.Append(" vývojári...");

   Response.Write (s1);
   Response.Write (s3);
%>
```

Príklad v jazyku C# *(súbor textCS.aspx)*

```
<%@ Page Language="C#" %>
<%
//spajanie retazcov
   String s1;
   String s2 = "Dobrý";
   s2 += " deň";
   s1 = s2 + " čitatelia, ";

//spajanie pomocou triedy StringBuilder
   StringBuilder s3 = new StringBuilder();
   s3.Append("dobrý");
   s3.Append("  deň");
   s3.Append(" vývojári...");

   Response.Write (s1);
   Response.Write (s3);
%>
```

Klientovi sa zobrazí HTML stránka v tvare:



Príklad pre spájanie textových reťazcov

ASP.NET kód verzus HTML tagy

Dosiaľ sme hlavne s ohľadom na migrujúcich čitateľov používali spôsob miešania serverového kódu s HTML kódom. Je to zaužívaný spôsob u systémov ASP, PHP, JSP... V praxi to vyzeralo asi takto:

```
<%@ Page Language="C#" %>
<%
  ...skriptovy kod...
%>
<h1> HTML kod </h1>
... html kod
<%
  ...skriptovy kod...
%>
...
```

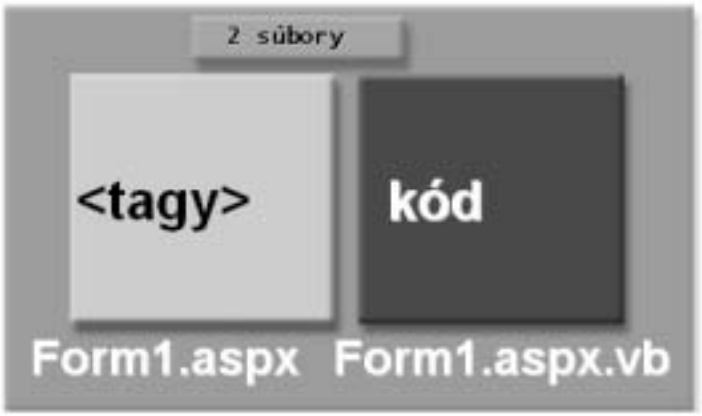
čiže vyjadrené schématicky

Miešanie serverového kódu a HTML

Pri malom rozsahu kódu môže byť tento spôsob dokonca prehľadnejší než mať kód projektu vo viacerých súboroch. Za jednoduchými zadaniami typu „internetové kníhkupectvo“, „internetový časopis“, „zoznamka“, „stránka zberateľa“ sa ale obvykle skrývajú stovky až tisícky riadkov kódu. V takomto prípade už rozdelenie projektu na viac súborov prinesie oveľa väčšiu prehľadnosť. Ak za zamyslíte nad stránkami aplikácií spomínaných typov, ktoré poznáte z Internetu, za každou takou vydarenou webovou aplikáciou sa skrývajú dve veci

– perfektne fungujúci kód
– estetický a funkčný dizajn

Na webovom projekte teda kooperujú minimálne dva tímy - vývojári a dizajnéri. Vývojári pracujú na serverovom kóde, a dizajnéri zasa navrhujú dizajn HTML stránok a formulárov. Aby si navzájom „neliezli do kapusty“ je výhodné oddeliť programový kód od HTML kódu. Veď stačí aby dizajnér spustil v nejakom pokročilom programe optimalizáciu kódu, pri ktorej sa nezriedka vymažú „zbytočné komentáre“ a dokážeme si ľahko predstaviť kam sa podel náš kód... Preto je výhodnejšie striktne rozdeliť súbory projektu na dve časti, na HTML kód a programový kód v príslušnom programovacom jazyku.



Miešanie serverového kódu a HTML

Toto rozdelenie môže byť jednak striktné, kedy máme aj fyzicky dva súbory, alebo len zdanlivé ako uvidíme u vývojového prostredia ASP.NET Web Matrix, kedy dokážeme obsah jedného fyzického súboru zobraziť v dvoch oknách.

Výber programovacieho jazyka

Dosiaľ sme sa mohli pokojne zaobísť bez vývojového prostredia. Tých pár riadkov sa dalo ľahko napísať v textovom editore Precvičovať ďalšie programové konštrukcie ako napríklad webové formuláre, validátory a podobne bez vývojového prostredia by už nebolo také pohodlné, preto najskôr predstavíme vývojové prostredia ASP.NET Web Matrix.

Problém by mohol nastať aj s výberom programovacieho jazyka. Nie že by nebolo z čoho vyberať, skôr naopak. Azda najpoužívanejšími programovacími jazykmi na ASP.NET stránkach budú Visual Basic a C#. U pragmatikov by pravdepodobne zvíťazila novinka C#, ale obidva spomínané programovacie jazyky majú svoje oblasti použitia a okruhy používateľov, takže nie je možné jednoznačne stanoviť „naj“ programovací jazyk

C#

Čo sa týka migrujúcih vývojárov, ktorí prechádzajú na technológiu ASP.NET, programovací jazyk C# osloví hlavne programátorov, ktorí dosiaľ pracovali s C++ a s Javou. A samozrejme tých, ktorí používali PHP, nakoľko syntax programovacieho jazyka používaného v PHP skriptoch je veľmi blízka Jave a objektovému céčku.

Visual Basic

Ak by sme chceli uplatniť analógiu z desktopových aplikácií, tak programovací jazyk Visual Basic bol určený na tvorbu jednak rôznych prototypových a testovacích aplikácií, prípadne príležitostných aplikácií, u ktorých bola dôležitá rýchlosť vývoja. Visual Basic sa ale uplatňoval aj u častí náročných projektov, hlavne u tých aplikácií, ktorých úlohou bola interakcia s používateľom. Vývojárov, ktorí pracovali s týmto programovacím jazykom je pomerne veľa a nie je žiadny dôvod, prečo by svoje skúsenosti nemali ďalej používať v ASP.NET projektoch.

Preto bude väčšina kódov príkladov v tejto publikácii uvedená v obidvoch programovacích jazykoch. Ak by sme mali odpovedať na otázku nováčikov, s čím teda začať, tu by odpoveď na túto otázku inklinovala skôr k programovaciemu jazyku C#

KAPITOLA 3:

Vývojové prostredie ASP.NET Web Matrix

Prelomovým bodom šírenia technológie ASP.NET a teda oným povestným chýbajúcim ohnivkom reťazca webovej aplikácie by mohlo byť práve vývojové prostredie WebMatrix, ktoré sa dá stiahnuť z adresy http://www.asp.net/webmatrix, prípadne nainštalovať z priloženého CD. Skeptici možno namietnu, že ak by aj Visual Studio bolo k dispozícii na webe, sťahovať obsah troch CD je spravidla nad možnosti „domácich vývojárov“. Web Matrix rieši aj túto záležitosť veľmi uspokojivo. Inštalačný súbor má menej ako 1.5 MB.

Úvodná obrazovka programu ASP.NET Web Matrix

Na webovej stránke Web Matrixu je okrem inštalačného súboru, dokumentácie, on-line tutoriálu aj 70 stránková elektronická kniha **Inside ASP.NET Web Matrix** z vydavateľstva Wrox vo formáte PDF.

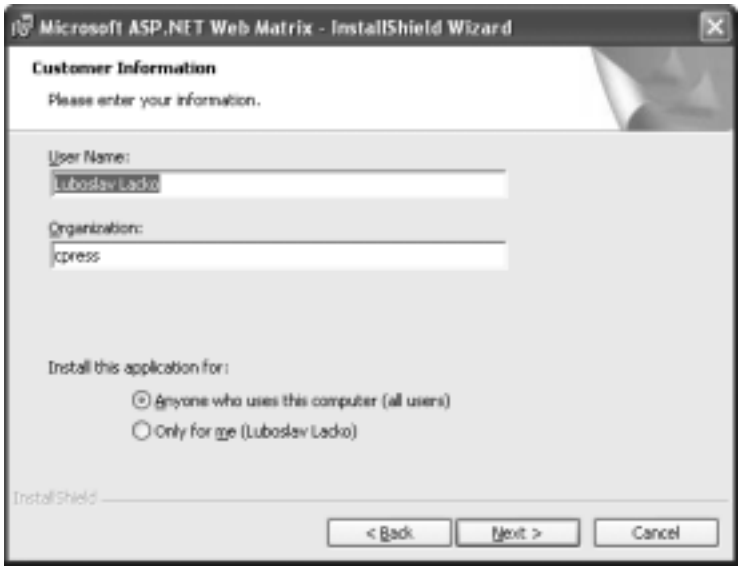
Kniha Inside ASP.NET Web Matrix

WebMatrix verzus Visual Studio.NET

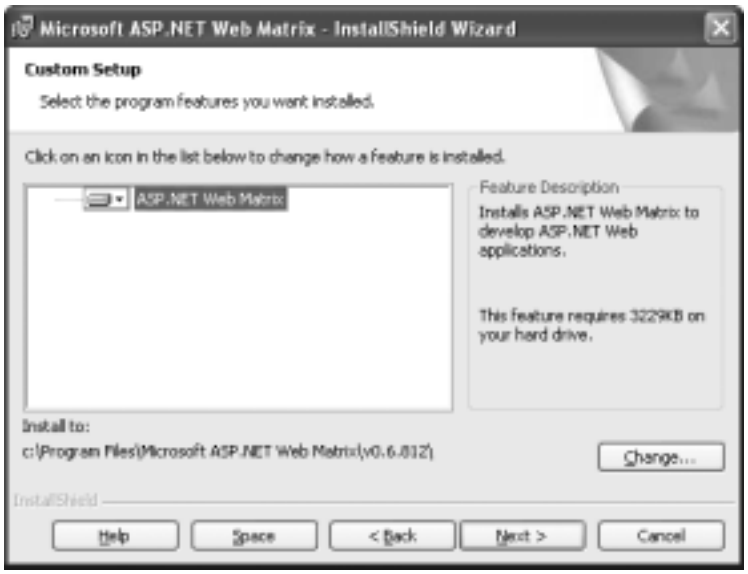
Pokúsme sa naznačiť hlavné rozdiely medzi vývojovými prostrediami Visual Studio.NET a Web Matrixom. Prvý veľmi príjemný rozdiel sme už uviedli - Web Matrix je zadarmo. Samozrejme vždy platí zásada, že niečo za niečo a tak ďalšie rozdiely trochu Web Matrix hendikepujú, ale veď komu by to vadilo, môže si doplatiť. A nemusí to byť zrovna Visual Studio.NET Enterprise Edition kedy sa predpokladá, že sa niekoľko desaťtisícová investícia zvýšením efektívnosti vývoja aplikácií v podnikovom prostredí rýchlo vráti. Za niekoľko tisíc korún sa dá kúpiť ako samostatný balík Visual C#, alebo Visual Basic.
Visual Studio je projektovo orientované vývojové prostredie, to znamená že so súbormi tvoriacimi projekt pracujeme ako s jedným kompaktným celkom. Web Matrix je súborovo orientované vývojové prostredie - to znamená že s projektom pracujeme po jednotlivých súboroch. Príjemnou správou však je, že WebMatrix umožňuje oddelenie HTML kódu a skriptového kódu. Výhody sú nesporné. Na webovom projekte pracuje spravidla viac ľudí, takto môže nezávisle pracovať dizajnér na HTML stránkach a programátor nad kódom. Ďalším rozdielom je, že Visual Studio podporuje v súčastnej dobe viac než dvadsať programovacích jazykov. WebMatrix podporuje tri programovacie jazyky. Dobrou správou je, že sú to tie najdôležitejšie. Visual C#, Visual Basic a Visual J#. Mnohonásobne menší inštalačný súbor WebMatrixu logicky svedčí o menej komplexnom používateľskom rozhraní. Všetky základné funkcionality moderného vývojového prostredia však zostali zachované. Namiesto chýbajúcej komplexnej dokumentácie odporúčame vytlačiť si spomínanú knihu Inside ASP.NET Web Matrix.

Inštalácia

Podmienkou pre nainštalovanie vývojového prostredia ASP.NET Web Matrix je mať nainštalovaný prehliadač webových stránok Internet Explorer 5.5 alebo vyšší a samozrejme platformu .NET Framework, nakoľko Web Matrix je .NET aplikácia. Inštalácia produktu je veľmi jednoduchá a okrem potvrdenia licenčných podmienok nevyžaduje od používateľa žiadny podstatný zásah. Stačí súhlasiť s licenčnými podmienkami a rozhodnúť sa či aplikáciu bude používať len aktuálne prihlásený používateľ, alebo všetci používatelia predmetného počítača.



Inštalácia ASP.NET Web Matrix

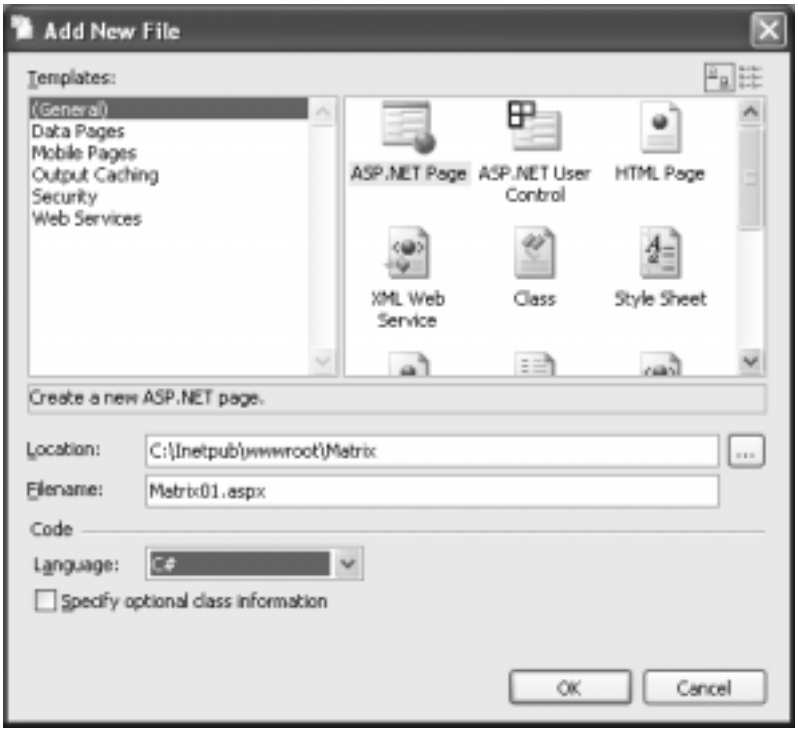


Inštalácia ASP.NET Web Matrix

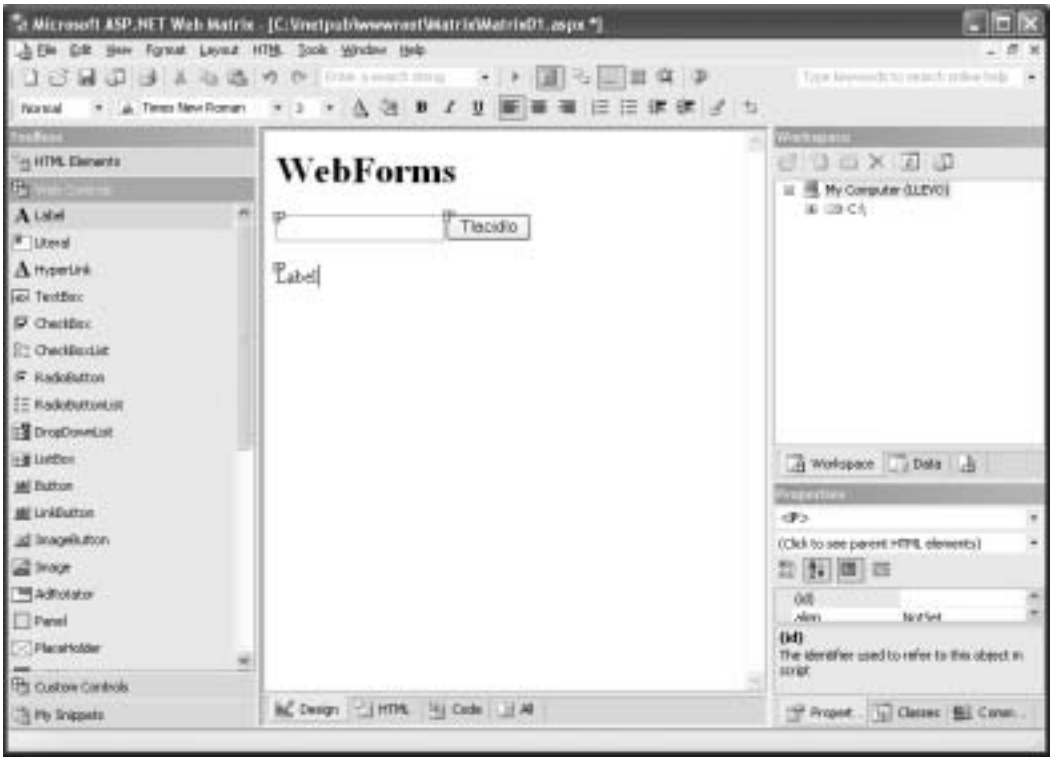
Pre inštaláciu sa vyžaduje zhruba 3.2 megabajtu voľného miesta na pevnom disku.

Začíname

Po spustení vývojového prostredia ASP.NET Web Matrix sa zobrazí dialógové okno "Add New File" v ktorom špecifikujeme, o aký súbor sa bude jednať. Už v stati, kde sme preberali rozdiely medzi Web Matrixom a vývojovým prostredím Visual Studio .NET sme sa dozvedeli, že vývojové prostredie ASP.NET Web Matrix je súborovo (nie projektovo) orientované. Zložitejšie projekty teda musíme na viac súborov rozvrhnúť my sami a tieto súbory aj vhodne pospájať



Prvý projekt



Pracovné okno vývojového prostredia Web Matrix

Okno aplikácie obsahuje okrem hlavného pracovného okna ešte integrované okná Toolbox, Workspace a Properites. Hlavné pracovné okno obsahuje štyri záložky: Design, HTML, Code a All, to znamená, že môžeme pracovať v štyroch režimoch, alebo inak povedané, štyri okná v záložkách nám poskytujú štyri rôzne pohľady na ten istý dokument. V návrhovom zobrazení záložky **Design** môžeme umiestňovať komponenty na vhodné miesto webového formulára, prípadne môžeme ich rozmiestnenie kedykoľvek poopraviť

V záložke **HTML** môžeme editovať kód HTML stránky vrátane kódu ASP komponentov. Po založení nového projektu obsahuje kód HTML stránky len základné tagy.

```
<html>
<head>
</head>
<body>
    <form runat="server">
        <!-- Insert content here -->
    </form>
</body>
</html>
```

Kód v záhlaví stránky je prístupný cez záložku **Code**.

```
// Insert page code here
//
```

S kompletným kódom celej ASP.NET stránky môžeme pracovať v záložke **All**.

```
<%@ Page Language="C#" %>
<script runat="server">

    // Insert page code here
    //

</script>
<html>
<head>
</head>
<body>
    <form runat="server">
        <!-- Insert content here -->
    </form>
</body>
</html>
```

Našu prázdnu aplikáciu dotvoríme na jednoduchý formulár. Na plochu formulára umiestnime komponenty Text Box, Button a Label. V okne HTML bude zobrazený HTML kód

```
<html><head></head>
<body>
    <form runat="server">
        <h1>WebForms
        </h1>
        <p>
            <asp:TextBox id="TextBox1" runat="server"></asp:TextBox>
            <asp:Button id="Button1" runat="server" Text="Tlacidlo"></asp:Button>
        </p>
        <p>
            <asp:Label id="Label1" runat="server">Label</asp:Label>
        </p>
```

```
        <!-- Insert content here -->
    </form>
</body>
</html>
```

Ak klikneme na tlačidlo, môžeme v záložke Code dopísať kód pre obsluhu udalosti zatlačenia tlačidla. Telo procedúry vygeneruje vývojové prostredie, my len dopíšeme jeden riadok aktívneho kódu, kedy text z komponenty TextBox prenesieme do komponenty Label.

```
Sub Button1_Click(sender As Object, e As EventArgs)
    Label1.Text = TextBox1.Text
End Sub
```

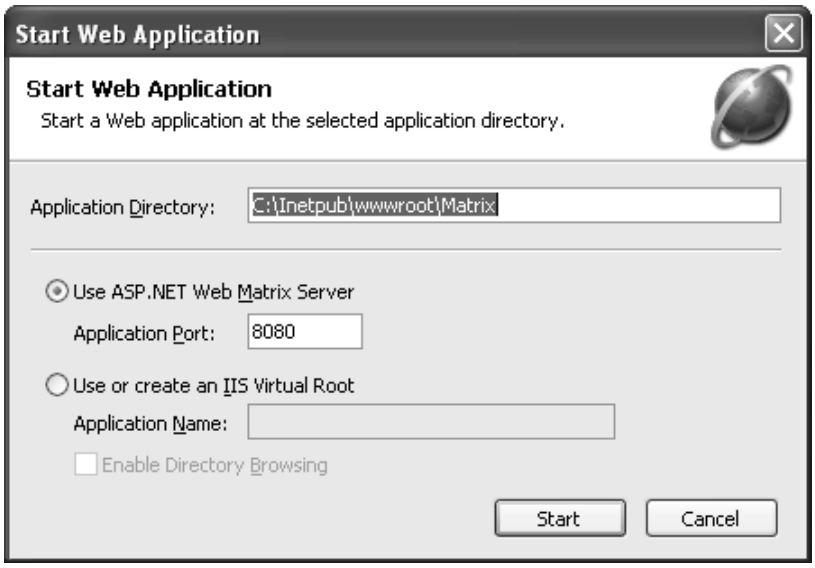
Kompletný kód celej aplikácie, teda súboru **Matrix1.aspx** bude

```
<%@ Page Language="C#" %>
<script runat="server">

    // Insert page code here
    //

    void Button1_Click(object sender, EventArgs e)
    {
      Label1.Text = TextBox1.Text;
    }

</script>

<html><head></head>
<body>
    <form runat="server">
        <h1>WebForms
        </h1>
        <p>
            <asp:TextBox id="TextBox1" runat="server"></asp:TextBox>
            <asp:Button id="Button1" onclick="Button1_Click" runat="server"
Text="Tlacidlo"></asp:Button>
        </p>
        <p>
            <asp:Label id="Label1" runat="server">Label</asp:Label>
        </p>
        <!-- Insert content here -->
    </form>
</body>
</html>
```

Celý kód má 30 riadkov, pričom nám stačilo z nich napísať jeden jediný a zvyšok tvorby kódu aplikácie spočíva v jednoduchom presune komponentov z toolboxu na plochu webového formulára a nastavení ich parametrov.

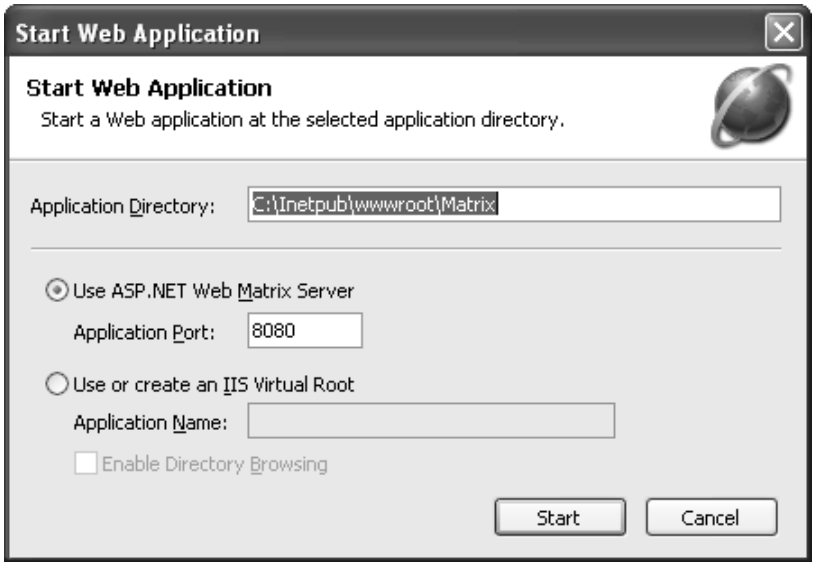
Spustenie kódu

Po aktivovaní menu View - Start, alebo po zatlačení klávesy F5 sa začne spúšťať nami vyvinutá webová aplikácia. Pre jej spustenie máme v zásade dve možnosti. Buď použiť web server, ktorý je súčasťou inštalácie Web Matrixu, alebo predtým nainštalovaný Internet Information Server



Dialóg pre výber webového servera

Webový server Web Matrixu „počúva" na porte 8080, to znamená adresa pre spustenie aplikácie v okne prehliadača HTML stránok bude http://localhost:8080/Matrix01.aspx
Ak sa rozhodneme pre Internet Information Server, použijeme adresu http://localhost/Matrix/Matrix01.aspx.



Ladenie aplikácie

Kód stránky, ktorý sa vygeneruje pre klienta bude

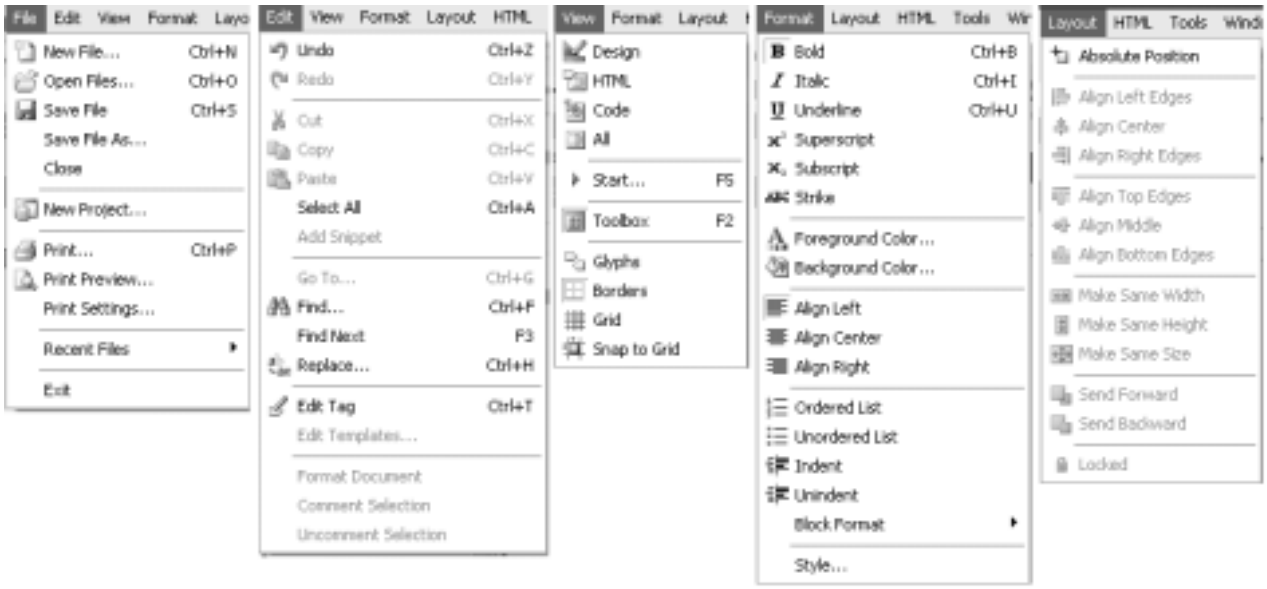
```
<html>
<head>
</head>
<body>
     <form name="_ctl0" method="post" action="Matrix01.aspx" id="_ctl0">
<input type="hidden" name="__VIEWSTATE"
value="dDwtMTA4MzE0MjEwNTt0PDtsPGk8MT47PjtsPHQ8O2w8aTw1Pjs+O2w8dDxwPHA8bDxUZXh0Oz47bDxObyBuYXpkYX
I7Pj47Pjs7Pjs+Pjs+Pjs+BOU/mja9Zm5zrP3eqcWV3+GNN3Q=" />

         <h1>WebForms
         </h1>
         <p>
```

```
        <input name="TextBox1" type="text" value="No nazdar" id="TextBox1" />
             <input type="submit" name="Button1" value="Tlacidlo" id="Button1" />
         </p>
         <p>
             <span id="Label1">No nazdar</span>
         </p>
         <!-- Insert content here -->
    </form>
</body>
</html>
```

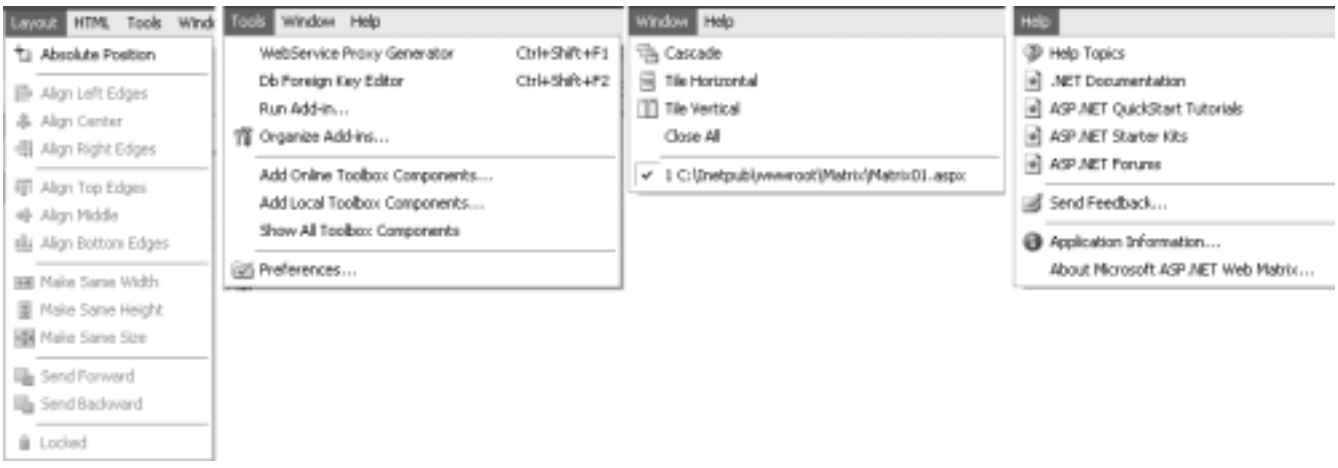
Po vyplnení textového okna a odoslaní hodnoty tlačidlom sa aktivuje tá istá stránka Matrix2.aspx, pričom stav sa odovzdá parametrom **"dDwtMTA4MzE0MjEwNTt0PDtsPGk8MT47PjtsPHQ8O2w8aTw1Pjs+O2w8dDxwPHA8bDxUZXh0Oz47bDxObyBuYXpkYXI7Pj47Pjs7Pjs+Pjs+Pjs+BOU/mja9Zm5zrP3eqcWV3+GNN3Q=".**

Ovládanie vývojového prostredia

Vývojové prostredie Web Matrix sa ovláda pomocou menu a toolbarov. „Expandovaný" systém menu v ktorom je možné vidieť jednotlivé funkcie dostupné z menu a ich prípadné klávesové skratky je na obrázku.



ASP.NET Web Matrix - Hlavné menu



ASP.NET Web Matrix - vodorovné pokračovanie hlavného menu

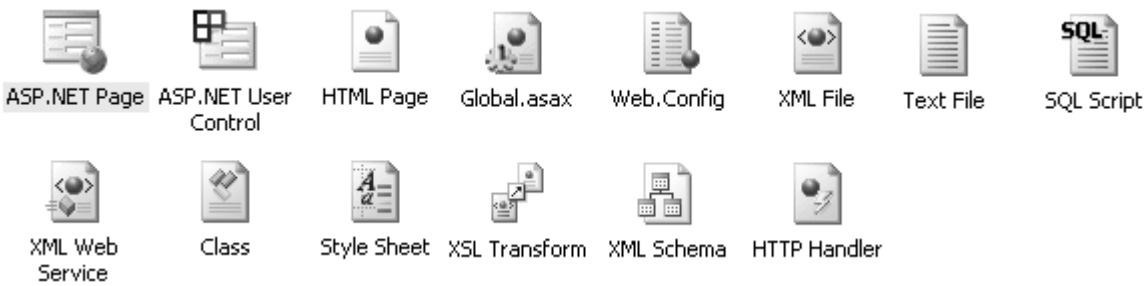
Z tlačidiel nás asi najviac bude zaujímať tlačidlo štart (šípka doprava) a ovládacie prvky pre formátovanie textu v HTML stránke.

ASP.NET Web Matrix - vodorovné pokračovanie hlavného menu

ASP.NET Web Matrix - elementy pre návrh HTML stránok

Typy súborov v zložke General

V zložke General môžeme vytvoriť nasledovné typy súborov



Typy súborov v zložke General

ASP.NET Page - pomocou tejto voľby vytvoríme súbor s príponou **.aspx**. Súbor obsahuje direktívu @Page directive, HTML tagy a server-side <form>.

```
<%@ Page Language="C#" %>
<script runat="server">

    // Insert page code here
    //

</script>
<html>
<head>
</head>
<body>
    <form runat="server">
        <!-- Insert content here -->
    </form>
</body>
</html>
```

ASP.NET User Control - pomocou tejto voľby vytvoríme súbor s príponou **.ascx**. Súbor obsahuje len direktívu @Control.

```
<%@ Control Language="C#" %>
<script runat="server">
    // Insert user control code here
    //
</script>
<!-- Insert content here -->
```

HTML Page - pomocou tejto voľby vytvoríme súbor s príponou **.htm**. Súbor obsahuje otváracie a zatváracie tagy.

```
<html><head></head>
<body>
    <!-- Insert content here -->
</body>
</html>
```

XML Web Service - pomocou tejto voľby vytvoríme súbor webovej služby s príponou .asmx. Súbor obsahuje direktívu @WebService. Ak zadáme CLASS WS_class a Namespace WS_namespace, vývojové prostredie vygeneruje kód:

```
<%@ WebService language="C#" class="WS_class" %>

using System;
using System.Web.Services;
using System.Xml.Serialization;

public class WS_class {

    [WebMethod]
    public int Add(int a, int b) {
        return a + b;
    }
}
```

Class - pomocou tejto voľby vytvoríme súbor s príponou .vb alebo .cs (podľa použitého programovacieho jazyka),

```
// NewFile.cs
//

namespace C_namespace {
    using System;

    /// <summary>
    /// Summary description for C_class.
    /// </summary>
    public class C_class {

        /// <summary>
        /// Creates a new instance of C_class
        /// </summary>
        public C_class() {
        }
    }
}
```

Style Sheet - pomocou tejto voľby vytvoríme súbor s príponou .css. Súbor obsahuje len prázdny selektor BODY{}

```
BODY {
}
```

Global.asax - pomocou tejto voľby vytvoríme súbor s príponou .asax, ktorý obsahuje direktívu @Application a sekciu <script>

```
<%@ Application language="C#" %>

<script runat="server">

    public void Application_Start(Object sender, EventArgs e) {
        // Code that runs on application startup
    }

    public void Application_End(Object sender, EventArgs e) {
        // Code that runs on application shutdown
    }

    public void Application_Error(Object sender, EventArgs e) {
        // Code that runs when an unhandled error occurs
    }

    public void Session_Start(Object sender, EventArgs e) {
        // Code that runs when a new session is started
    }

    public void Session_End(Object sender, EventArgs e) {
        // Code that runs when a session ends
    }

</script>
```

Web.Config - pomocou tejto voľby vytvoríme súbor s názvom web.config, ktorý obsahuje sekcie <configuration>, <appSettings> a <system.web>. Vo vnútri sekcie <system.web> sú elementy <sessionState>, <customErrors>, <authentication> a <authorization>. Súbor je vo formáte XML a jednotlivé sekcie a elementy sú okomentované

```
 <?xml version="1.0" encoding="UTF-8" ?>
<configuration>
    <!--
         The <appSettings> section is used to configure application-specific configuration
         settings.  These can be fetched from within apps by calling the
         "ConfigurationSettings.AppSettings(key)" method:

         <appSettings>
            <add key="connectionstring"
value="server=localhost;trusted_connection=true;database=pubs"/>
         </appSettings>
    -->

    <system.web>
        <!--
            The <sessionState" section is used to configure session state for the application.
            It supports four modes: "Off", "InProc", "StateServer", and "SqlServer".  The
            later two modes enable session state to be stored off the web server machine -
            allowing failure redundancy and web farm session state scenarios.

            <sessionState mode="InProc"
                          stateConnectionString="tcpip=127.0.0.1:42424"
                          sqlConnectionString="data source=127.0.0.1;trusted_connection=true"
                          cookieless="false"
                          timeout="20" />
        -->
        <!--
            The <customErrors> section enables configuration of what to do if/when an
            unhandled error occurs during the execution of a request.  Specifically, it
            enables developers to configure html error pages to be displayed in place of
            a error stack trace:

            <customErrors mode="RemoteOnly" defaultRedirect="GenericErrorPage.htm">
               <error statusCode="403" redirect="NoAccess.htm"/>
               <error statusCode="404" redirect="FileNotFound.htm"/>
            <customErrors>
        -->
        <!--
            The <authentication> section enables configuration of the security authentication
            mode used by ASP.NET to identify an incoming user.  It supports a "mode"
            attribute with four valid values: "Windows", "Forms", "Passport" and "None":

            The <forms> section is a sub-section of the <authentication> section,
            and supports configuring the authentication values used when Forms
            authentication is enabled above:

            <authentication mode="Windows">

                    <forms name=".ASPXAUTH"
                           loginUrl="login.aspx"
                           protection="Validation"
                           timeout="999999" />

             </authentication>
        -->

        <!--
```

```
            The <authorization> section enables developers/administrators to configure
            whether a user or role has access to a particular page or resource.  This is
            accomplished by adding "<allow>" and "<deny>" sub-tags beneath the <authorization>
            section - specifically detailing the users/roles allowed or denied access.

            Note: The "?" character indicates "anonymous" users (ie: non authenticated users).
            The "*" character indicates "all" users.

            <authorization>
               <allow users="joeuser" />
               <allow roles="Admins" />
               <deny users="*" />
            </authorization>
        -->
    </system.web>
</configuration>
```

XML File - pomocou tejto voľby vytvoríme súbor s príponou .xml.

```
<?xml version="1.0" encoding="utf-8" ?>
```

XSL Transform - pomocou tejto voľby vytvoríme XSLT stylesheet súbor s príponou .xslt.

```
<?xml version="1.0" encoding="utf-8" ?>
<stylesheet version="1.0" xmlns="http://www.w3.org/1999/XSL/Transform">
</stylesheet>
```

XML Schema - pomocou tejto voľby vytvoríme súbor XML (XSD) schémy s príponou .xsd.

```
<?xml version="1.0" encoding="utf-8" ?>
<xsd:schema targetNamespace="http://tempuri.org/NewFile.xsd"
  xmlns="http://tempuri.org/NewFile.xsd"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
</xsd:schema>
```

ASP.NET HTTP Handler - pomocou tejto voľby vytvoríme súbor s príponou ashx. Súbor obsahuje direktívu @WebHandler

```
<%@ WebHandler language="C#" class="C_namespace.C_class" %>

using System;
using System.Web;

namespace C_namespace {

    public class C_class : IHttpHandler {

        public void ProcessRequest(HttpContext context) {
            // TODO: Write request handling code here
        }

        public bool IsReusable {
            get {
                return true;
            }
        }
    }
}
```

Text File - pomocou tejto voľby vytvoríme prázdny súbor s príponou .txt.
SQL Script - pomocou tejto voľby vytvoríme súbor s príponou SQL, ktorý obsahuje len komentár "/* New SQL script */".

Typy súborov v zložke Data Pages
V zložke Data Pages môžeme vytvoriť nasledovné typy súborov

Typy súborov v zložke Data Pages

Jednotlivé druhy tabuliek a reportov budú ukázané v kapitole o vývoji databázových aplikácií. Súbory v ostatných zložkách zatiaľ len pre orientáciu vymenujeme

Typy súborov v zložke Mobile Pages
V zložke Mobile Pages môžeme vytvoriť dva typy súborov

Typy súborov v zložke Mobile Pages

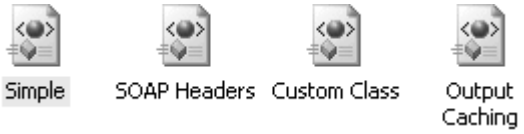
Typy súborov v zložke Output Caching
V zložke Output Caching môžeme vytvárať tieto štyri typy súborov

Typy súborov v zložke Output Caching

Typy súborov v zložke Security
V zložke Security môžeme vytvárať tri typy súborov

Typy súborov v zložke Security

Typy súborov v zložke Web Services
V zložke WebServices môžeme vytvárať štyri typy súborov



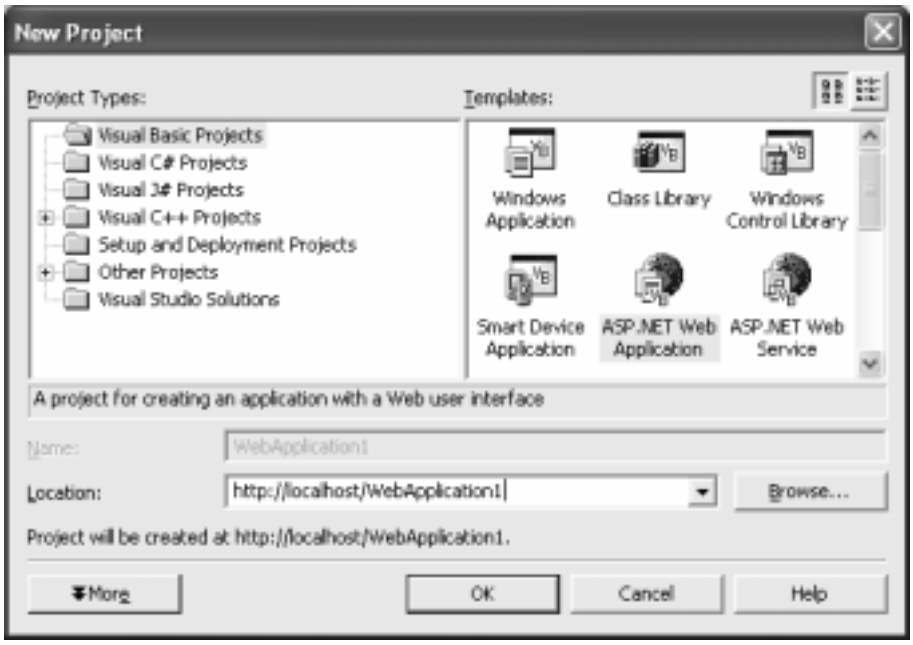
Typy súborov v zložke Web Services

Možnosti migrácie - Visual Studio.NET 2003

Hoci primárne je táto publikácia určená vývojárom „hobby" aplikácií, niekedy sa stane, že naše hobby alebo pokusná webová aplikácia, napríklad zásielková služba a podobne prerastie do seriózneho biznisu a vtedy je potrebné webovú aplikáciu škálovať, prípadne migrovať do výkonnejšieho serverového prostredia. Pre vývoj rozsiahlejších, zložitejších a robustnejších ASP.NET aplikácií (a .NET aplikácií vo všeobecnosti) sa používa vývojové prostredie Visual Studio .NET, pričom jeho najnovšia verzia má označenie 2003. Jednak z tohto dôvodu a jednak pre možnosť porovnávania vývojových prostred Visual Studio .NET a Web Matrix sme zaradili do publikácie aj túto kapitolu, teda veľmi stručné predstavenie Visual Studia .NET vo vzťahu k technológii ASP.NET.

Začíname - nový projekt

Po aktivácii položky menu New Project máme na výber dve skupiny možností, ktoré sa líšia navonok len v použitom programovacom jazyku. Všimnime si, že aj ikony v obidvoch zložkách sú prakticky rovnaké, líšia sa len znakom VB, alebo #. Použiť môžeme teda buď Visual Basic a potom zvolíme zložku Visual Basic Projects, alebo Visual C#. Preto zvolíme záložku Visual C# Projects, a konkrétne ikonu ASP.NET Web Application. Vedľajšia ikona zasa ponúka možnosť vytvorenie webovej služby.



Nový projekt vo Visual Studiu

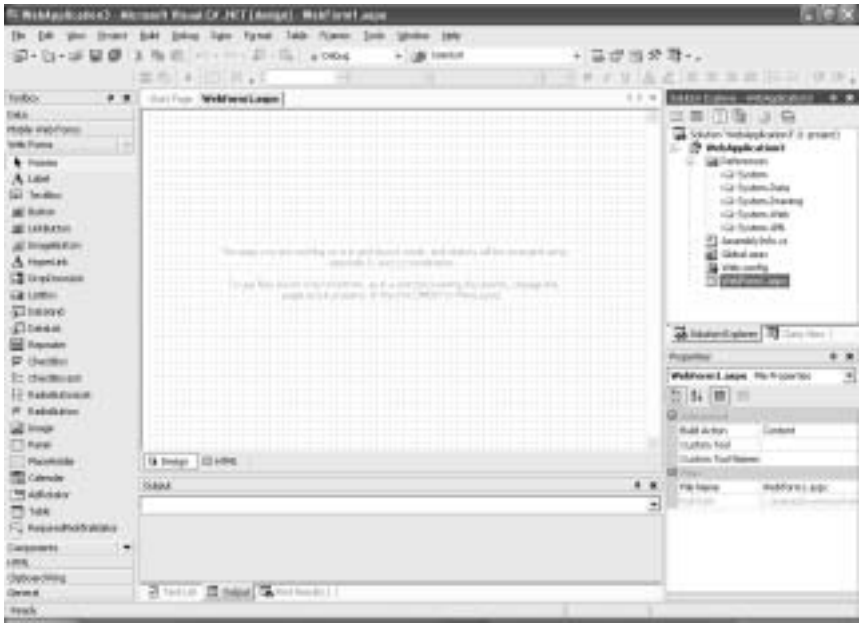
Projekt nejako pomenujeme, napríklad Projekt1. Novo založený projekt bude potom k dispozícii pre vývoj, ladenie a testovanie na lokálnej adrese http://localhost/Projekt1

Súbory tvoriace projekt budú fyzicky umiestnené v adresári pre publikovanie webových dokumentov, v našom prípade v adresári C:\Inetpub\wwwroot vznikol nový podadresár s názvom zhodným s názvom nášho projektu, teda Projekt. V podadresári projektu je niekoľko súborov:

```
AssemblyInfo.cs
Global.asax
Global.asax.cs
Global.asax.resx
Projekt1.csproj
Projekt1.csproj.webinfo
Projekt1.vsdisco
Web.config
WebForm1.aspx
WebForm1.aspx.cs
WebForm1.aspx.resx
```

Nás bude zaujímať z celého projektu zatiaľ hlavne súbor WebForm1.aspx, teda „klasická" ASP.NET stránka.

Základné rozloženie pracovnej obrazovky a ovládacie prvky sú veľmi podobne rozmiestnené u Visual Studia aj u Web Matrixu. Samozrejme ponuka možností Visual Studia je nepomerne bohatšia.. V ľavej časti máme Toolbox s ponukou komponentov, v strednej časti je pracovné okno, v ktorom môžeme mať zobrazený buď zdrojový kód, alebo návrhové zobrazenie a v pravej časti vidíme dva okná: Solution Explorer a Properties.



Obr3 - Visual Studio.NET, vývoj webovej aplikácie ASP.NET

Režim zobrazenia v strednej časti pracovnej plochy Visual Studia.NET nastavujeme pomocou záložiek Design a HTML v jeho spodnej časti. V režime Design máme zobrazený (zatiaľ) prázdny formulár, ktorý budeme neskôr dotvárať a v režime HTML môžeme písať prípadne editovať zdrojový kód ASP.NET stránky.
Tu by sa zdalo, že Web Matrix má lepšie možnosti, veď dokáže oddeliť HTML kód od ASP.NET kódu, pretože má ešte záložky Code a ALL. Opak je však pravdou. Projekt Web Matrixu má implicitne fyzicky umiestnený kód aj HTML stránku v jednom súbore a oddelenie kódu je len zdanlivé na úrovni zobrazení časti kódu.

Pri vytváraní aplikácie nezačíname z nuly, sprievodca vytvorením aplikácie (Application Wizard) už v etape vytvárania novej aplikácie vygeneroval v súbore WebForm1.aspx kód:

```
<%@ Page language="c#" Codebehind="WebForm1.aspx.cs" AutoEventWireup="false"
InheritsProjekt1.WebForm1" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
  <HEAD>
    <title>WebForm1</title>
    <meta name="GENERATOR" Content="Microsoft Visual Studio 7.0">
    <meta name="CODE_LANGUAGE" Content="C#">
    <meta name="vs_defaultClientScript" content="JavaScript">
    <meta name="vs_targetSchema"      content="http://schemas.microsoft.com/intellisense/ie5">
   </HEAD>
   <body MS_POSITIONING="GridLayout">
       <form id="Form1" method="post" runat="server">
       </form>
   </body>
</HTML>
```

Na rozdiel od Web Matrixu, kde je ASP.NET aj HTML kód v jednom súbore a kompilácia kódu nastáva až pri prvom zobrazení, u Visual Studia .NET je potrebné najskôr projekt zostaviť (menu menu Build - Build Solution), čím sa preloží ASP.NET kód napísaný vo Visual Basicu alebo C# do managed kódu.

Ako sme už zdôraznili, Visual Studio .NET pri návrhu projektu dôsledne dodržiava oddelenie HTML kód od kódu pre obsluhu udalostí. Návrh formulárov je v súbore s príponou ASPX kdežto programový kód v jazyku C# je sústredený v samostatnom súbore s príponou CS.

KAPITOLA 4:

Vývoj ASP.NET aplikácií

Táto kapitola spolu s kapitolou o databázových aplikáciách tvoria akési jadro publikácie o vývoji webových aplikácií s využitím technológie ASP.NET. Nakoľko chceme osloviť aj segment začínajúcich vývojárov a vývojárov „hobby" aplikácií použijeme ako vývojové prostredie Web Matrix. Prechod na komerčne dodávané vývojové prostredie Visual Studio .NET je samozrejme bezproblémový a toto vývojové prostredie nám pochopiteľne poskytne viac komfortu než voľne šíriteľný Web Matrix

Webové formuláre

Okrem aplikačnej logiky budeme samozrejme potrebovať aj rôzne ovládacie prvky, napríklad pre zadávanie údajov a podobne. Na platforme ASP.NET sú k dispozícii všetky druhy ovládacích prvkov, ktoré poznáme z klasických HTML formulárov, ale aj z desktopových aplikácií napísaných vo vyšších programovacích jazykoch. Pretože jednotlivé komponenty formulárov sú vlastne základnými stavebnými kameňmi webových stránok, budeme im v tejto publikácii venovať väčšiu pozornosť.

Serverové ovládacie prvky

Jedná sa o ovládacie prvky, ktoré dôverne poznáme z HTML stránok, hlavne z rôznych formulárov. Pridaním klauzuly **runat=server** však zabezpečíme, že sa kód vykoná na strane servera. Skôr než budeme spracovávať používateľom zadané údaje z formulára, ukážeme pravdepodobne najjednoduchší kód pre výpis naformátovaného textu, teda použijeme komponentu Label)

```
<%@ Page Language="C#" %>
<html>
   <body>
       <h1><font face="Verdana">Web Controls</font></h1>
       <asp:label
           font-size="16" forecolor="blue"  runat=server>Formátovaný text
       </asp:label>
   </body>
</html>
```

Tento kód môžeme napísať pomocou ľubovoľného textového editora do súboru kod1.aspx v našom prípade sme tento súbor umiestnili do adresára (takto je to aj na CD) C:\Inetpub\wwwroot\ASP_zbornik\k4 .
Na HTML stránke klienta sa po zadaní URL adresy http://localhost/asp_zbornik/k4/kod1.aspx zobrazí požadovaný text.

Najjednoduchší príklad pre Web Controls

Zatiaľ sme sa k žiadnej výraznej novej funkcionalite nedopracovali. Ak porovnáme zdrojový kód a serverom vygenerovanú HTML stránku u klienta

```
<html><body>
        <h1><font face="Verdana">Web Controls</font></h1>
        <span style="color:Blue;font-size:16pt;">Formátovaný text
        </span>
    </body>
</html>
```

je to takmer to isté. Na tomto jednoduchom príklade sa to neprejaví, ale vo všeobecnosti nám serverové prvky umožňujú programovú manipuláciu s elementami HTML. Je to vlastne objekt (z hľadiska terminológie objektovo orientovaného programovania), inými slovami povedané - každý prvok má určité vlastnosti a metódy. Používať na príklady tohoto rozsahu vývojové prostredie snáď ani nie je potrebné, no pre napísanie taketjo aplikácie vo Web Matrixe by sme museli urobiť jeden jediný úkon, presunúť ikonku Web komponenty Label na pracovnú plochu aplikácie a zmeniť text.

Ovládacie prvky sa veľmi často zoskupujú do formulárov. Formulár sa zobrazí používateľovi na HTML stránke. Po jeho vyplnení používateľom zadané údaje odošleme na server a následne spracujeme. Aby sme ukázali výhody novej technológie ukážeme typický príklad ako sa spracovávali údaje z formulára na klasických ASP stránkach.

Príklad formulára na ASP stránke
Aplikačná logika v tomto prípade pozostávala z dvoch častí. Na prvej ASP, alebo HTML stránke bol HTML kód formulára. Po zatlačení potvrdzovacieho tlačidla sa údaje odoslali na server, kde boli spracované kódom na inej ASP stránke. Miniaplikácia pre spracovanie údajov z formulára teda pozostávala z dvoch častí. Kódu formulára (súbor FORMULAR.HTML)

```
<html><body>
<h1>Klasicky formular</h1>
<P></P>
  <FORM ACTION=formular.asp METHOD=post>
     Vaše meno:<INPUT id=meno name=meno>  
              <INPUT type=submit value="Potvrď">
  </FORM>
</body></html>
```

a ASP stránky (súbor FORMULAR.ASP)

```
<html><body>
<h1>Klasicky formular - spracovanie</h1>
<%
   Response.write ("Nazdar " & Request.Form("heslo"))
%>
</body></html>
```

Príklad formulára realizovaného pomocou ASP.NET
Pre riešenie tej istej úlohy pomocou ASP.NET potrebujeme len jednu stránku (súbor formular1.aspx). Táto stránka obsahuje jednak HTML kód formulára, jednak kód pre spracovanie používateľom zadaných údajov

```
<html>
    <script language="C#" runat="server">
        void Kliknutie(Object Src, EventArgs E)
        {
            Message.Text = "Nazdar " + Name.Text;
        }
    </script>
```

```
    <body>
       <h1>Príklad formulára</font></h1>
       <form action="controls3.aspx" runat=server>
          Vase meno: <asp:textbox id="Name" runat=server/>
                     <asp:button text="Potvrď"
                      Onclick=" Kliknutie " runat=server/>
                     <asp:label id="Message" runat=server/>
       </form>
    </body>
</html>
```

Najskôr sa presvedčíme, či to funguje, a potom sa budeme venovať vysvetleniu princípu.

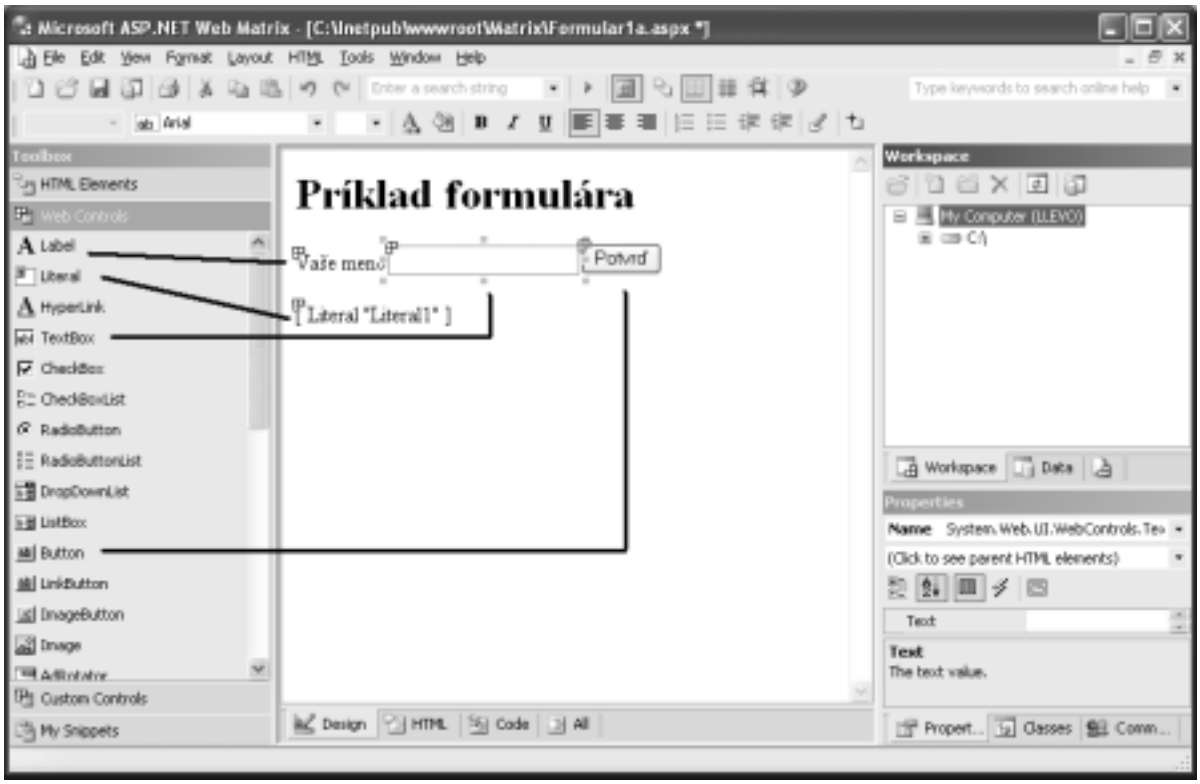
Formulár

Veľmi nám pomôže, ak si zobrazíme zdrojový kód stránky, ktorá bude vygenerovaná pre klientov prehliadač.

```
<html> <body>
       <h1>Príklad formulára</font></h1>
       <form name="_ctl0" method="post" action="formular1.aspx" id="_ctl0">
<input type="hidden" name="__VIEWSTATE"
value="dDwxMzc4MDMwNTk1O3Q8O2w8aTwyPjs+O2w8dDw7bDxpPDU+Oz47bDx0PHA8cDxsPFRleHQ7PjtsPE5hemRhciBNaW
xhbjs+Pjs+Ozs+Oz4+Oz4+Oz5IITb0PWSI79ZAVATTIYfkzl7YDw==" />
          Vase meno: <input name="Name" type="text" value="Milan" id="Name" />
                     <input type="submit" name="_ctl1" value="Potvrď" />
                     <span id="Message">Nazdar Milan</span>
       </form>
</body></html>
```

V prvom rade je potrebné vyzdvihnúť, že pre klienta bola vygenerovaná úplne štandardná HTML stránka bez akýchkoľvek prvkov, ktoré by mohli spôsobovať potenciálnu nekompatibilitu pri jej zobrazení v rôznych prehliadačoch. Nie sú tu nijaké ActiveX komponenty, Java applety, len čistý HTML kód, s ktorým si rovnako dobre poradí akýkoľvek prehliadač webových stránok. Všetky informácie o druhu a stave komponenty budú serveru odovzdané pomocou skrytého parametra formulára s názvom `__VIEWSTATE`.

Už pre takýto jednoduchý formulár je výhodnejšie použiť vývojové prostredie Web Matrix. V okne Design ho vytvoríme pomocou Web komponentov Label, Text Box, Button a Literal. Nadpis bol vpísaný priamo do kontextu HTML stránky a naformátovaný ako `<h1>`.



Návrh formulára vo vývojovom prostredí Web Matrix

Po niekoľkých jednoduchých presunoch komponentov pomocou kurzoru myši vznikol prakticky totožný kód, z toho vyplýva, že vývojové prostredie pracuje pomerne efektívne.

```
<%@ Page Language="C#" %>
<script runat="server">

    // Insert page code here
    //

</script>
<html><head></head>
<body>
    <form runat="server">
        <h1>Príklad formulára
        </h1>
        <p>
            <asp:Label id="Label2" runat="server">Vaše meno</asp:Label>
            <asp:TextBox id="Name" runat="server"></asp:TextBox>
            <asp:Button id="Button1" onclick="Button1_Click" runat="server"
Text="Potvrď"></asp:Button>
        </p>
        <p>
            <asp:Literal id="Literal1" runat="server"></asp:Literal>
            <!-- Insert content here -->
        </p>
    </form>
</body></html>
```

Takto vytvorený kód zobrazí formulár, no zatiaľ neobsahuje kód pre spracovanie používateľom zadaných hodnôt. Ten dopíšeme na miesto naznačené komentárom

```
// Insert page code here
//
```

V tomto okamihu by bolo ešte zbytočné pustiť sa do písania jadra procedúry, ktorou ošetrujeme udalosť zatlačenia tlačidla. Vývojové prostredie nám totiž dokáže vygenerovať aj jej kostru. Stačí dvojklik na tlačidlo v návrhovom formulári.

```
// Insert page code here
//
    void Button1_Click(Object sender, EventArgs e) {

    }
```

Telo procedúry už napíšeme sami. Zadaný text z komponenty TextBox vypíšeme do komponenty Literal

```
// Insert page code here
    //

    void Button1_Click(Object sender, EventArgs e) {
      Literal1.Text = "Nazdar " + Name.Text;;
    }
```

Prvkov, ktoré sú nám takto k dispozícii je niekoľko desiatok. Zatiaľ sme použili prvky TextBox, Button, Label a Literal. V stručnom prehľade ukážeme syntax a príklady použitia pre najviac používané komponenty. Prehľad všetkých komponentov dostupných cez Toolbox vývojového prostredia Web Matrix je na obrázku

Komponenty dostupné cez toolbox vo vývojovom prostredí Web Matrix

TextBox

Jednoduché alebo viacriadkové editačné okno pre zadávanie textových údajov. Zmenu textu, teda napísanie alebo vymazanie niektorého znaku môžeme monitorovať pomocou metódy OnTextChanged.

Definícia syntaxe

```
<asp:TextBox id="value"
      AutoPostBack="True|False"
      Columns="characters"
      MaxLength="characters"
      Rows="rows"
      Text="text"
      TextMode="SingleLine | MultiLine | Password"
      Wrap="True|False"
      OnTextChanged="OnTextChangedMethod"
      runat="server"/>
```

Kód vytvorený Web Matrixom po presunutí komponenty na pracovnú plochu

```
  <form runat="server">
        <asp:TextBox id="TextBox1" runat="server"></asp:TextBox>
        <!-- Insert content here -->
    </form>
```

Príklad použitia

```
<asp:TextBox runat=server
      id="txtLogin"
      Text=""
      Font_Face="Arial" Font_Size="3"
      BackColor="lightblue"
      TextMode="MultiLine"
      Height="10"
      OnTextChanged="txtLogin_Changed"/>
```

Button

Klasické tlačidlo, ktoré poznáme z webových stránok aj z desktopových aplikácií, po zatlačení ktorého môžeme realizovať rôzne akcie.

Definícia syntaxe

```
<asp:Button id="MyButton"
      Text="label"
      CommandName="command"
      CommandArgument="commandargument"
      CausesValidation="true | false"
      OnClick="OnClickMethod"
      runat="server"/>
```

Kód vytvorený Web Matrixom po presunutí komponenty na pracovnú plochu

```
   <form runat="server">
        <asp:Button id="Button1" onclick="Button1_Click" runat="server"
Text="Button"></asp:Button>
        <!-- Insert content here -->
    </form>
```

Šablóna kódu vytvoreného Web Matrixom pre obsluhu udalosti - zatlačenia tlačidla

```
[C#]    void Button1_Click(Object sender, EventArgs e) {}

[VB]    Sub Button1_Click(sender As Object, e As EventArgs)
        End Sub
```

Príklad použitia pre **Submit** button

```
<asp:Button id="SubmitButton"
     Text="Submit"
     OnClick="SubmitBtn_Click"
     runat="server"/>
```

Príklad použitia pre **Command** button

```
<asp:Button id="SortAscendingButton"
     Text="Sort Ascending"
     CommandName="Sort"
     CommandArgument="Ascending"
     OnCommand="CommandBtn_Click"
     runat="server"/>
```

Príklad obsluhy udalosti [Visual Basic]

```
Sub CommandBtn_Click(sender As Object, e As CommandEventArgs)
   Message.Text = "You clicked the " & e.CommandName & _
                  " - " & e.CommandArgument & " button."
End Sub
```

Príklad obsluhy udalosti [C#]

```
void CommandBtn_Click(Object sender, CommandEventArgs e)
{
   Message.Text = "You clicked the " + e.CommandName +
                  " - " + e.CommandArgument + " button.";
}
```

Label

Prvok Label umožní zobrazenie textového výpisu, napríklad hodnoty nejakej textovej premennej a podobne.

Definícia syntaxe

```
<asp:Label id="Label1"
     Text="label"
     runat="server"/>
```

Kód vytvorený Web Matrixom po presunutí komponenty na pracovnú plochu

```
    <form runat="server">
        <asp:Label id="Label1" runat="server">Label</asp:Label>
        <!-- Insert content here -->
    </form>
```

Literal

Táto komponenta sa používa pre rendrovanie statického textu na webovej stránke.

Definícia syntaxe

```
<asp:Literal id="Literal1"
     Text="Text"
     runat="server"/>
```

Kód vytvorený Web Matrixom po presunutí komponenty na pracovnú plochu

```
    <form runat="server">
        <asp:Literal id="Literal1" runat="server"></asp:Literal>
        <!-- Insert content here -->
    </form>
```

Príklad použitia

```
<html><head></head>
<body>
   <form runat="server">
      <h3>Literal Example</h3>
      <asp:Literal id="Literal1"
           Text="Hello World!!"
           runat="server"/>
   </form>
</body></html>
```

DropDownList

Prvok umožňuje zoznam možností, ktorý je možné rozvinúť a vybrať jednu z možností. Možnosti môžeme definovať jednak priamo v kóde, prípadne môžeme DropDownList napojiť na vhodný stĺpec databázovej tabuľky.

Definícia syntaxe

```
<asp:DropDownList id="DropDownList1" runat="server"
      DataSource="<% databindingexpression %>"
      DataTextField="DataSourceField"
      DataValueField="DataSourceField"
      AutoPostBack="True|False"
      OnSelectedIndexChanged="OnSelectedIndexChangedMethod">
   <asp:ListItem value="value" selected="True|False">
      Text
   </asp:ListItem>
</asp:DropDownList>
```

Kód vytvorený Web Matrixom po presunutí komponenty na pracovnú plochu

```
   <form runat="server">
        <asp:DropDownList id="DropDownList1" runat="server"
OnSelectedIndexChanged="DropDownList1_SelectedIndexChanged"></asp:DropDownList>
        <!-- Insert content here -->
    </form>
```

Šablóna kódu vytvoreného Web Matrixom pre obsluhu udalosti - zmeny vybraného objektu

```
[C#]    void DropDownList1_SelectedIndexChanged(Object sender, EventArgs e) {}

[VB]    Sub DropDownList1_SelectedIndexChanged(sender As Object, e As EventArgs)
        End Sub
```

Príklad použitia

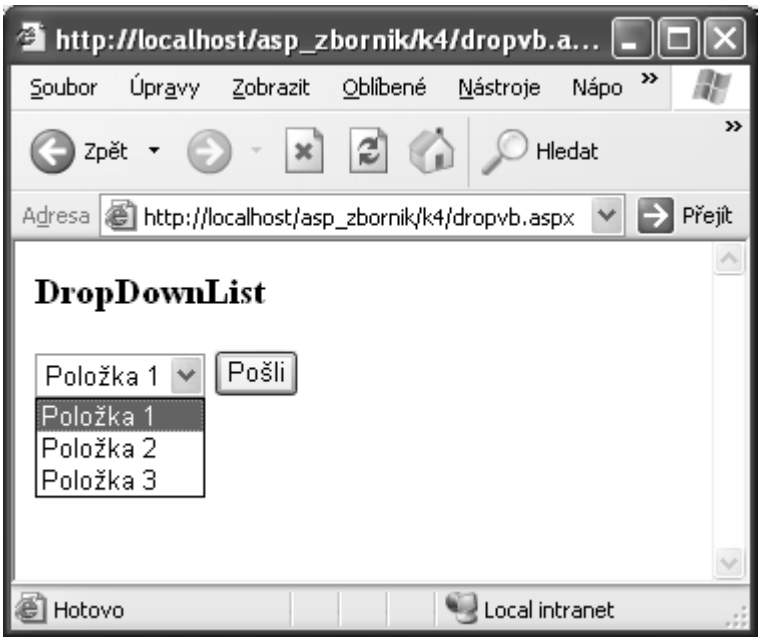
[Visual Basic]

```
<%@ Page Language="VB" AutoEventWireup="True" %>
<html><head>
   <script runat="server">
      Sub Button_Click(sender As Object, e As EventArgs)
         Label1.Text = "Váš výber " & DropDownList1.SelectedItem.Text & "."
      End Sub 'Button_Click
```

```
  </script>
</head>
<body>
    <form runat="server">
       <h3>DropDownList</h3>
       <asp:DropDownList id="DropDownList1" runat="server">
          <asp:ListItem>Položka 1</asp:ListItem>
          <asp:ListItem>Položka 2</asp:ListItem>
          <asp:ListItem>Položka 3</asp:ListItem>
       </asp:DropDownList>
       <asp:Button id="Button1"
            Text="Pošli"
            OnClick="Button_Click"
            runat="server"/>
       <br><br>
       <asp:Label id="Label1"
            runat="server"/>
    </form>
</body></html>
```

[C#] (je uvedená len obslužná procedúra udalosti kliknutia na tlačidlo, zvyšok ASP stránky je rovnaký ako pre Visual Basic

```
<%@ Page Language="C#" AutoEventWireup="True" %>
<html><head>
    <script runat="server">
       void Button_Click(Object sender, EventArgs e)
       {
          Label1.Text = "Váš výber: " +
          dropdownlist1.SelectedItem.Text + ".";
       }
    </script>
</head>
<body>
    ...
</body></html>
```



DropDownList

Po výbere jednej s ponúkaných možností a zatlačení tlačidla Odošli sa na mieste HTML stránky, ktoré je špecifikované kódom <asp:Label id="Label1" runat="server"/> objaví výpis v závislosti od vybranej položky, napríklad: Váš výber: Položka 1

ListBox
Pomocou tejto komponenty môžeme realizovať výber jednej alebo viacerých položiek z ponúkaných možností. Aj túto komponentu môžeme napojiť na vhodný stĺpec databázovej tabuľky.

Definícia syntaxe

```
<asp:ListBox id="Listbox1"
      DataSource="<% databindingexpression %>"
      DataTextField="DataSourceField"
      DataValueField="DataSourceField"
      AutoPostBack="True|False"
      Rows="rowcount"
      SelectionMode="Single|Multiple"
      OnSelectedIndexChanged="OnSelectedIndexChangedMethod"
      runat="server">

   <asp:ListItem value="value" selected="True|False">
      Text
   </asp:ListItem>

</asp:ListBox>
```

Kód vytvorený Web Matrixom po presunutí komponenty na pracovnú plochu

```
   <form runat="server">
        <asp:ListBox id="ListBox1" runat="server"
OnSelectedIndexChanged="ListBox1_SelectedIndexChanged"></asp:ListBox>
        <!-- Insert content here -->
    </form>
```

Šablóna kódu vytvoreného Web Matrixom pre obsluhu udalosti - zmeny vybraného objektu

```
[C#]      void ListBox1_SelectedIndexChanged(Object sender, EventArgs e) {}

[VB]      Sub ListBox1_SelectedIndexChanged(sender As Object, e As EventArgs)
          End Sub
```

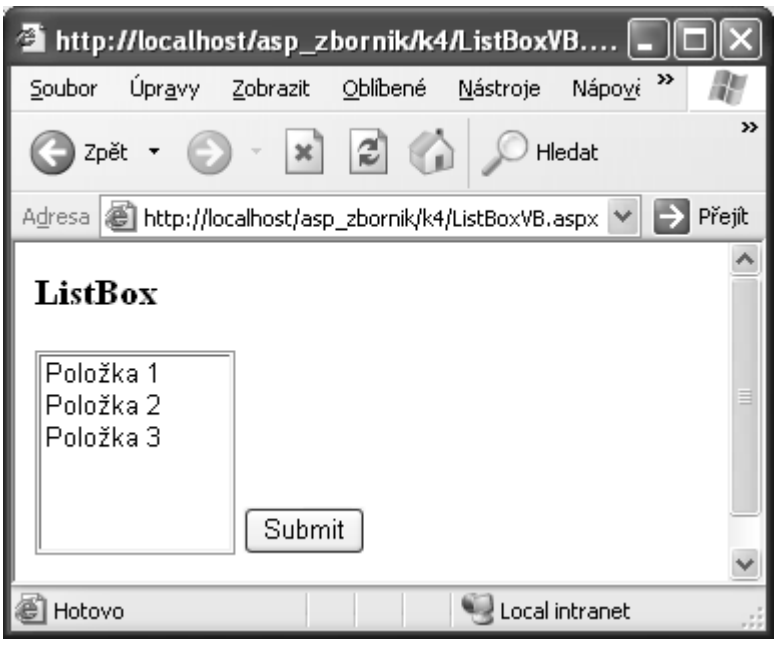
Príklad použitia
[Visual Basic]

```
<%@ Page Language="VB" AutoEventWireup="True" %>
<html><head>
   <script runat="server">
      Sub SubmitBtn_Click(sender As Object, e As EventArgs)
         If ListBox1.SelectedIndex > - 1 Then
            Label1.Text = "You chose: " & ListBox1.SelectedItem.Text
         End If
      End Sub 'SubmitBtn_Click
   </script>
</head>
<body>
   <h3>ListBox</h3>
   <p>
   <form runat="server">
      <asp:ListBox id="ListBox1"
      Rows="6"
```

```
            Width="100px"
            SelectionMode="Single"
            runat="server">
          <asp:ListItem>Položka 1</asp:ListItem>
          <asp:ListItem>Položka 2</asp:ListItem>
          <asp:ListItem>Položka 3</asp:ListItem>
       </asp:ListBox>
       <asp:button id="Button1"
            Text="Submit"
            OnClick="SubmitBtn_Click"
            runat="server" />
       <p>
       <asp:Label id="Label1" runat="server"/>
   </form>
</body></html>
```

[C#] (je uvedená len obslužná procedúra udalosti kliknutia na tlačidlo, zvyšok ASP.NET stránky je rovnaký ako pre Visual Basic

```
<%@ Page Language="C#" AutoEventWireup="True" %>
<html>
<head>
   <script runat="server">
      void SubmitBtn_Click(Object sender, EventArgs e)
      {
         if (ListBox1.SelectedIndex > -1)
            Label1.Text="The first item you chose: " +
                        ListBox1.SelectedItem.Text;
      }
   </script>
</head>
<body>
   ...
   ...
</body></html>
```



ListBox

CheckBox
Tento jednoduchý ovládací prvok umožňuje prepínať medzi dvomi hodnotami pravda - nepravda (TRUE-FALSE)

Definícia syntaxe

```
<asp:CheckBox id="CheckBox1"
      AutoPostBack="True|False"
      Text="Label"
      TextAlign="Right|Left"
      Checked="True|False"
      OnCheckedChanged="OnCheckedChangedMethod"
      runat="server"/>
```

Kód vytvorený Web Matrixom po presunutí komponenty na pracovnú plochu

```
   <form runat="server">
         <asp:CheckBox id="CheckBox1" runat="server"
OnCheckedChanged="CheckBox1_CheckedChanged"></asp:CheckBox>
         <!-- Insert content here -->
     </form>
```

Šablóna kódu vytvoreného Web Matrixom pre obsluhu udalosti - zaškrtnutie elementu

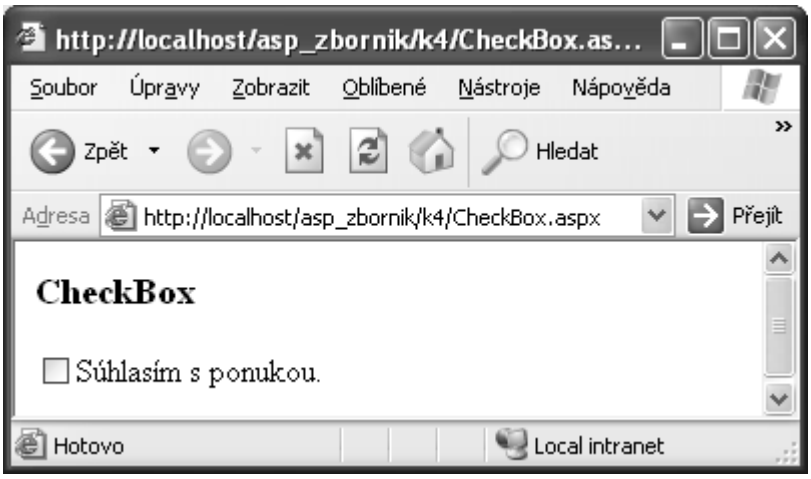
```
[C#]    void CheckBox1_CheckedChanged(Object sender, EventArgs e) {}

[VB]    Sub CheckBox1_CheckedChanged(sender As Object, e As EventArgs)
        End Sub
```

Príklad použitia (obslužná procedúra pre zmenu stavu check boxu obsahuje v tomto príklade len telo bez výkonného kódu)
[Visual Basic]

```
<%@ Page Language="VB" AutoEventWireup="True" %>
<html><head>
<script runat="server">
       Sub Check_Clicked(sender As Object, e As EventArgs)

       End Sub
</script>

</head><body>
    <form runat="server">
       <h3>CheckBox</h3>
       <asp:CheckBox id="SameCheckBox"
             AutoPostBack="True"
             Text="Súhlasím s ponukou."
             TextAlign="Right"
             OnCheckedChanged="Check_Clicked"
             runat="server"/>
    </form>
</body></html>
```

[C#] (je uvedená len obslužná procedúra udalosti kliknutia na tlačidlo, zvyšok ASP.NET stránky je rovnaký ako pre Visual Basic

```
<%@ Page Language="C#" AutoEventWireup="True" %>
<html><head>
   <script runat="server">

       void Check_Clicked(Object sender, EventArgs e)
       {

       }

    </script>
</head>
...
```



CheckBox

CheckBoxList
Ovládací prvok umožní výber jednej alebo viacerých možností z ponúkaného zoznamu. Výber sa vykonáva zaškrtnutím políčka vedľa textu. Komponentu je možné naviazať na databázovú tabuľku

Definícia syntaxe

```
<asp:CheckBoxList id="CheckBoxList1"
      AutoPostBack="True|False"
      CellPadding="Pixels"
      DataSource='<% databindingexpression %>'
      DataTextField="DataSourceField"
      DataValueField="DataSourceField"
      RepeatColumns="ColumnCount"
      RepeatDirection="Vertical|Horizontal"
      RepeatLayout="Flow|Table"
      TextAlign="Right|Left"
      OnSelectedIndexChanged="OnSelectedIndexChangedMethod"
      runat="server">

   <asp:ListItem value="value"
         selected="True|False">
      Text
   </asp:ListItem>

</asp:CheckBoxList>
```

Kód vytvorený Web Matrixom po presunutí komponenty na pracovnú plochu

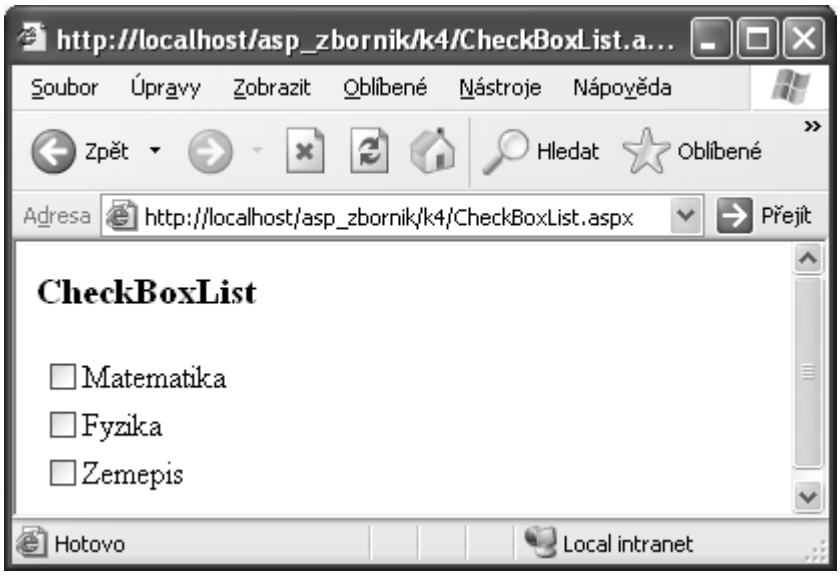
```
      <form runat="server">
         <asp:CheckBoxList id="CheckBoxList1" runat="server"
OnSelectedIndexChanged="CheckBoxList1_SelectedIndexChanged"></asp:CheckBoxList>
         <!-- Insert content here -->
     </form>
```

Šablóna kódu vytvoreného Web Matrixom pre obsluhu udalosti - zaškrtnutie elementu

```
[C#]    void CheckBoxList1_SelectedIndexChanged(Object sender, EventArgs e) {}

[VB]    Sub CheckBoxList1_SelectedIndexChanged(sender As Object, e As EventArgs)
        End Sub
```

Príklad použitia

```
<%@ Page Language="VB" AutoEventWireup="True" %>
<html><body>
   <form runat="server">
      <h3>CheckBoxList</h3>
    <asp:CheckBoxList runat=server>
       <asp:ListItem>Matematika</asp:ListItem>
       <asp:ListItem>Fyzika</asp:ListItem>
       <asp:ListItem>Zemepis</asp:ListItem>
   </asp:CheckBoxList>

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



CheckBoxList

RadioButton

Ovládací prvok RadioButton sám o sebe veľký zmysel nemá, je potrebné použiť týchto prvkov niekoľko, potom fungujú ako prepínač (odtiaľ analógia RadioButton - tlačidlová sada pre prepínanie rozsahov rozhlasového prijímača)

Definícia syntaxe

```
<asp:RadioButton id="RadioButton1"
      AutoPostBack="True|False"
      Checked="True|False"
      GroupName="GroupName"
```

```
      Text="label"
      TextAlign="Right|Left"
      OnCheckedChanged="OnCheckedChangedMethod"
      runat="server"/>
```

Kód vytvorený Web Matrixom po presunutí komponenty na pracovnú plochu

```
      <form runat="server">
         <asp:RadioButton id="RadioButton1" runat="server"
OnCheckedChanged="RadioButton1_CheckedChanged"></asp:RadioButton>
         <!-- Insert content here -->
     </form>
```

Šablóna kódu vytvoreného Web Matrixom pre obsluhu udalosti - prepnutie prepínača

```
[C#]    void RadioButton1_CheckedChanged(Object sender, EventArgs e) {}

[VB]    Sub RadioButton1_CheckedChanged(sender As Object, e As EventArgs)
        End Sub
```

Príklad použitia

[Visual Basic]

```
<%@ Page Language="VB" AutoEventWireup="True" %>
<html>
<head>
   <script runat="server">
      Sub SubmitBtn_Click(Sender As Object, e As EventArgs)
         If Radio1.Checked Then
            Label1.Text = "Váš výber " & Radio1.Text
         ElseIf Radio2.Checked Then
            Label1.Text = "Váš výber " & Radio2.Text
         ElseIf Radio3.Checked Then
            Label1.Text = "Váš výber " & Radio3.Text
         End If
      End Sub
   </script>
</head>
<body>
   <h3>RadioButton</h3>
   <form runat="server">
      <h4>Objednávam konfiguráciu:</h4>
      <asp:RadioButton id=Radio1
            Text="Low - End"
            Checked="True"
            GroupName="RadioGroup1"
            runat="server" /><br>Za málo peňazí málo muziky, ale lepšie ako nič.<p>
      <asp:RadioButton id=Radio2
            Text="Optimal"
            GroupName="RadioGroup1"
            runat="server"/><br>Optimálna konfigurácia pre väčšinu používateľov.<p>
      <asp:RadioButton id=Radio3
            Text="Enterprise"
            GroupName="RadioGroup1"
            runat="server" /><br>Do najlepších firiem len to najlepšie.<p>
      <asp:Button text="Objednal"
            OnClick="SubmitBtn_Click"
            runat=server/>
      <asp:Label id=Label1
```

```
            Font-Bold="true"
            runat="server" />
    </form>
</body></html>
```

[C#] (je uvedená len obslužná procedúra udalosti kliknutia na tlačidlo, zvyšok ASP.NET stránky je rovnaký ako pre Visual Basic

```
<%@ Page Language="C#" AutoEventWireup="True" %>
<html>
<head>

<script runat="server">
       void SubmitBtn_Click(Object Sender, EventArgs e)
       {
          if (Radio1.Checked)
          {
             Label1.Text = "Váš výber " + Radio1.Text;
          }
          else if (Radio2.Checked)
          {
             Label1.Text = "Váš výber " + Radio2.Text;
          }
          else if (Radio3.Checked)
          {
             Label1.Text = "Váš výber " + Radio3.Text;
          }
       }
    </script>

</head>
```

RadioButton

RadioButtonList

Ako vyplýva z názvu, tento ovládací prvok funguje podobne ako prepínacie tlačidlá vlnových rozsahov na starších rádioprijímačoch. Môžeme ho naplniť položkami priamo v kóde, alebo naviazať na databázovú tabuľku

Definícia syntaxe

```
<asp:RadioButtonList id="RadioButtonList1"
      AutoPostBack="True|False"
      CellPadding="Pixels"
      DataSource="<% databindingexpression %>"
      DataTextField="DataSourceField"
      DataValueField="DataSourceField"
      RepeatColumns="ColumnCount"
      RepeatDirection="Vertical|Horizontal"
      RepeatLayout="Flow|Table"
      TextAlign="Right|Left"
      OnSelectedIndexChanged="OnSelectedIndexChangedMethod"
      runat="server">

   <asp:ListItem Text="label"
         Value="value"
         Selected="True|False" />

</asp:RadioButtonList>
```

Kód vytvorený Web Matrixom po presunutí komponenty na pracovnú plochu

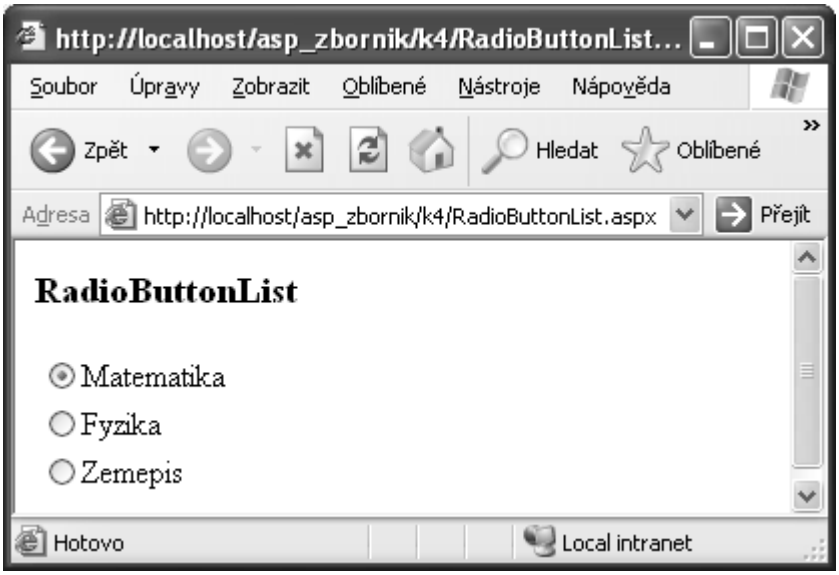
```
      <form runat="server">
         <asp:RadioButtonList id="RadioButtonList1" runat="server"
OnSelectedIndexChanged="RadioButtonList1_SelectedIndexChanged"></asp:RadioButtonList>
         <!-- Insert content here -->
     </form>
```

Šablóna kódu vytvoreného Web Matrixom pre obsluhu udalosti - prepnutie prepínača

```
[C#]     void RadioButtonList1_SelectedIndexChanged(Object sender, EventArgs e) {}

[VB]    Sub RadioButtonList1_SelectedIndexChanged(sender As Object, e As EventArgs)
        End Sub
```

Príklad použitia

```
<%@ Page Language="VB" AutoEventWireup="True" %>
<html><body>
   <form runat="server">
      <h3>RadioButtonList</h3>
    <asp:CheckBoxList runat=server>
       <asp:ListItem>Matematika</asp:ListItem>
       <asp:ListItem>Fyzika</asp:ListItem>
       <asp:ListItem>Zemepis</asp:ListItem>
   </asp:CheckBoxList>

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



RadioButtonList

Image
Ovládací prvok Image slúži pre zobrazenie obrázku

Definícia syntaxe

```
<asp:Image id="Image1" runat="server"
      ImageUrl="string"
      AlternateText="string"
      ImageAlign="NotSet|AbsBottom|AbsMiddle|BaseLine|
                  Bottom|Left|Middle|Right|TextTop|Top"/>
```

Kód vytvorený Web Matrixom po presunutí komponenty na pracovnú plochu

```
      <form runat="server">
         <asp:Image id="Image1" runat="server"></asp:Image>
         <!-- Insert content here -->
     </form>
```

Príklad použitia

```
<%@ Page Language="VB" AutoEventWireup="True" %>
<html><body>
    <form runat="server">
       <h3>Image</h3>
     <asp:Image  runat=server
             AlternateText="Snehulienka"
             ImageAlign="left"
             ImageUrl="obr4.jpg"
       />
    </form>
</body></html>
```

Image

ImageButton
Na rozdiel od prvku Image, prvok ImageButton umožňuje obslúžiť udalosť kliknutia.

Definícia syntaxe

```
<asp:ImageButton id="ImageButton1"
      ImageUrl="string"
      Command="Command"
      CommandArgument="CommandArgument"
      CausesValidation="true | false"
      OnClick="OnClickMethod"
      runat="server"/>
```

Kód vytvorený Web Matrixom po presunutí komponenty na pracovnú plochu

```
      <form runat="server">
         <asp:ImageButton id="ImageButton1" onclick="ImageButton1_Click"
runat="server"></asp:ImageButton>
         <!-- Insert content here -->
     </form>
```

Šablóna kódu vytvoreného Web Matrixom pre obsluhu udalosti - zatlačenie tlačidla

```
[C#]    void ImageButton1_Click(Object sender, ImageClickEventArgs e) {}

[VB]    Sub ImageButton1_Click(sender As Object, e As ImageClickEventArgs)
        End Sub
```

Na jednoduchom príklade ukážeme ako zistiť súradnice bodu na obrázku, na ktorý používateľ klikol. Bod so súradnicami 0,0 je v ľavom hornom rohu obrázku

Príklad použitia **[Visual Basic]**

```
<%@ Page Language="VB" AutoEventWireup="True" %>
<html><head>
   <script runat="server">
      Sub ImageButton_Click(sender As Object, e As ImageClickEventArgs)
         Label1.Text="Súradnice: (" & e.X.ToString() & ", " & _
                     e.Y.ToString() & ")"
      End Sub
   </script>
</head><body>
   <form runat="server">
      <h3>ImageButton</h3>
```

```
      Kliknite na obrázok<br><br>
      <asp:ImageButton id="imagebutton1" runat="server"
            AlternateText="ImageButton 1"
            ImageAlign="left"
            ImageUrl="obr4.jpg"
            OnClick="ImageButton_Click"/>
      <br><br>
      <asp:label id="Label1" runat="server"/>
   </form>
</body></html>
```

Príklad použitia **[C#]**

```
<%@ Page Language="C#" AutoEventWireup="True" %>
<html>
<head>
   <script runat="server">
      void ImageButton_Click(object Source, ImageClickEventArgs e)
      {
         Label1.Text="Súradnice: (" + e.X.ToString() + ", " +
                     e.Y.ToString() + ")";
      }
   </script>
</head>
<body>
   <form runat="server">
      <h3>ImageButton</h3>
      Kliknite na obrázok.<br><br>
      <asp:ImageButton id="imagebutton1"
            AlternateText="ImageButton 1"
            ImageAlign="left"
            ImageUrl="obr4.jpg"
            OnClick="ImageButton_Click"
            runat="server"/>
      <br><br>
      <asp:Label id="Label1"
            runat="server"/>
   </form>
</body></html>
```

ImageButton

AdRotator

Typickým príkladom použitia ovládacieho prvku AdRotator je rotácia reklamných bannerov.

Definícia syntaxe

```
<asp:AdRotator
      id="Value"
      AdvertisementFile="AdvertisementFile"
      KeyWordFilter="KeyWord"
      Target="Target"
      OnAdCreated="OnAdCreatedMethod"
      runat="server"/>
```

Priaznivci “well formated" XML si prídu na svoje práve pri popise tohto prvku, pretože predpis pre rotáciu bannerov taktiež vo formáte XML. Jeho schéma (neúplná) je nasledovná:

```
<Advertisements>
   <Ad>
      <ImageUrl>
         URL adresa obrázku pre inzerát
      </ImageUrl>
      <NavigateUrl>
         URL adresa stránky, kam má používateľa "zvies inzerát"
      </NavigateUrl>
      <AlternateText>
         Text ktorý sa zobrazí ako tooltip
      </AlternateText>
      <Keyword>
         Kľúčové slovo pre možnos filtrovania textu
      </Keyword>
     ..
   </Ad>
</Advertisements>
```

Kód vytvorený Web Matrixom po presunutí komponenty na pracovnú plochu

```
    <form runat="server">
         <asp:AdRotator id="AdRotator1" runat="server" Width="468px" Height="60px"
OnAdCreated="AdRotator1_AdCreated"></asp:AdRotator>
         <!-- Insert content here -->
     </form>
```

Šablóna kódu vytvoreného Web Matrixom pre obsluhu udalosti

```
[C#]    void AdRotator1_AdCreated(Object sender, AdCreatedEventArgs e) {}

[VB]    Sub AdRotator1_AdCreated(sender As Object, e As AdCreatedEventArgs)
        End Sub
```

Pre rotáciu dvoch reklamných bannerov sme použili súbor REKLAMA.XML s nasledovným obsahom

```
<Advertisements>
   <Ad>
      <ImageUrl>obr1.jpg</ImageUrl>
      <NavigateUrl>http://www.pcrevue.sk</NavigateUrl>
      <AlternateText>Webové sídlo PCREVUE</AlternateText>
      <Impressions>80</Impressions>
      <Keyword>Časopis</Keyword>
      <Caption>Popis pre prvý inzerát</Caption>
```

```
    </Ad>

    <Ad>
        <ImageUrl>obr2.jpg</ImageUrl>
        <NavigateUrl>http://www.pc.sk</NavigateUrl>
        <AlternateText>Počítačový denník</AlternateText>
        <Impressions>80</Impressions>
        <Keyword>Webový denník</Keyword>
        <Caption>Popis pre druhy inzerat</Caption>
    </Ad>
</Advertisements>
```

Kód ASP.NET pre rotáciu bannerov bude potom:

```
<asp:AdRotator runat=server
      AdvertisementFile = "reklama.xml"
      Target="_newwwindow"
      OnAdCreated="AdCreated_Event"
/>
```

Calendar

Okrem jednoduchých prvkov formulára pre zadávanie textu, alebo pre zapnutie, vypnutie, alebo prepnutie nejakých volieb veľmi často potrebujeme zadávať kalendárne údaje. Nie je problém požadovať zadanie týchto údajov v klasickom editačnom okne formulára, no jednak tým utrpí presnosť zadania (ako uvidíme neskôr, toto by sme pomerne pracne mohli ošetriť pomocou validátorov), no hlavne utrpí používateľský komfort. A to už nehovoríme o rozličných formátoch dátumu a času používaných v anglosaských krajinách a u nás, o oddeľovačoch dní a podobne. Ovládací prvok Calendar zobrazí mesačný kalendár, pomocou ktorého môže klient jednoducho a interaktívne zadať požadovaný dátum, alebo špecifikovať nejaké časové obdobie. Použitie tohoto ovládacieho prvku je veľmi jednoduché

Definícia syntaxe

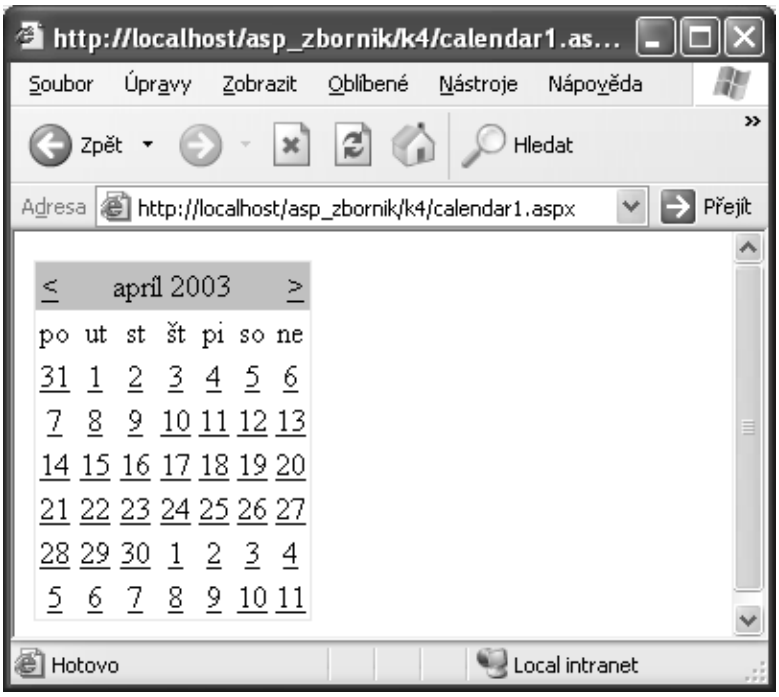
```
<asp:Calendar id="Calendar1"
      CellPadding="pixels"
      CellSpacing="pixels"
      DayNameFormat="FirstLetter|FirstTwoLetters|Full|Short"
      FirstDayOfWeek="Default|Monday|Tuesday|Wednesday|Thursday|Friday|Saturday|Sunday"
      NextMonthText="HTML text"
      NextPrevFormat="ShortMonth|FullMonth|CustomText"
      PrevMonthText="HTML text"
      SelectedDate="date"
      SelectionMode="None|Day|DayWeek|DayWeekMonth"
      SelectMonthText="HTML text"
      SelectWeekText="HTML text"
      ShowDayHeader="True|False"
      ShowGridLines="True|False"
      ShowNextPrevMonth="True|False"
      ShowTitle="True|False"
      TitleFormat="Month|MonthYear"
      TodaysDate="date"
      VisibleDate="date"
      OnDayRender="OnDayRenderMethod"
      OnSelectionChanged="OnSelectionChangedMethod"
      OnVisibleMonthChanged="OnVisibleMonthChangedMethod"
      runat="server">

   <TodayDayStyle property="value"/>
   <DayHeaderStyle property="value"/>
   <DayStyle property="value"/>
```

```
   <NextPrevStyle property="value"/>
    <OtherMonthDayStyle property="value"/>
    <SelectedDayStyle property="value"/>
    <SelectorStyle property="value"/>
    <TitleStyle property="value"/>
    <TodayDayStyle property="value"/>
    <WeekendDayStyle property="value"/>

</asp:Calendar>
```

Pomerne rozsiahlej definície syntaxe sa nemusíme príliš obávať, najjednoduchší kód pre vytvorenie kalendárovej komponenty je

```
<html><body>
   <form runat="server">
      <asp:Calendar ID="Calendar1" runat="server">
      </asp:Calendar>
   </form>
</body></html>
```

samozrejme v takomto prípade aj výsledok, ktorý sa zobrazí u klienta úplne zodpovedá vynaloženej námahe.



najjednoduchšie použitie ovládacieho prvku Calendar

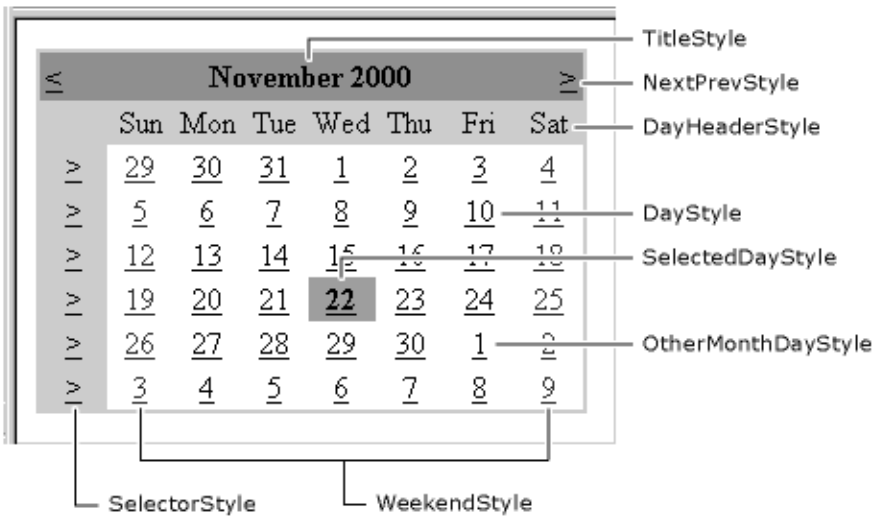
Kód vytvorený Web Matrixom po presunutí komponenty na pracovnú plochu

```
   <form runat="server">
         <asp:Calendar id="Calendar1" runat="server"
OnSelectionChanged="Calendar1_SelectionChanged"></asp:Calendar>
         <!-- Insert content here -->
     </form>
```

Šablóna kódu vytvoreného Web Matrixom pre obsluhu udalosti - zatlačenie tlačidla

```
[C#]    void Calendar1_SelectionChanged(Object sender, EventArgs e) {}

[VB]    ...
        End Sub
```

U každej komponenty, ktorú predložíme klientovi na našich stránkach musíme dbať na dve veci, ktoré sa dokonca vzájomne podporujú. Jednou z nich je estetický vzhľad, s ktorým ide ruka v ruke aj používateľský komfort a v neposlednom rade aj presnosť zadaných údajov. Komponenta Calendar nám poskytuje rozsiahle možnosti pre vylepšenie jej návrhového zobrazenia a prispôsobenie funkčnosti. Ako už vieme z predchádzajúceho dielu, kód pre definíciu vlastnej komponenty je vo formáte XML, stačí teda len pridať do tohoto kódu ďalšie elementy, pomocou ktorých definujeme vzhľad a funkčnosť



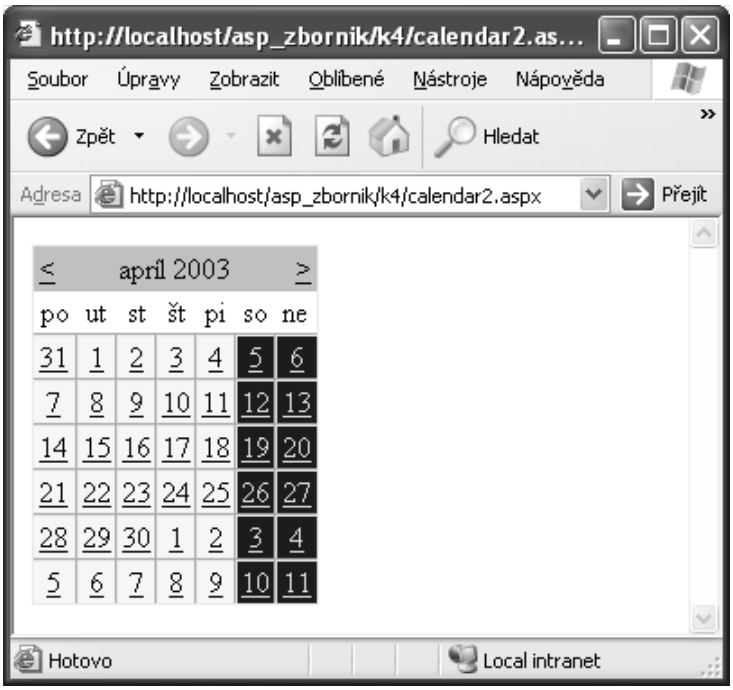
názvy elementov pre prispôsobenie vzhľadu ovládacieho prvku Calendar

Význam a funkcia jednotlivých elementov je zrejmá z obrázku a tabuľky:

Element	Popis
DayStyle	Formát zobrazenia dní v aktuálnom mesiaci. Niektoré dni, ako napríklad dnešný dátum, a víkendové dni sa formátujú pomocou osobitných elementov. Taktiež sa pomocou osobitných elementov formátujú dni, ktoré do aktuálneho mesiaca nepatria
DayHeaderStyle	Riadok v ktorom sú zobrazené názvy dní v týždni.
NextPrevStyle	Formát prvkov (najčastejšie šípok) pre navigáciu na predchádzajúci, alebo nasledujúci mesiac
OtherMonthDayStyle	Formát zobrazenie dní, ktoré nepatria do aktuálneho mesiaca.
SelectedDayStyle	Formát zobrazenia používateľom vybratého dňa.
SelectorStyle	Formát výberu jedného dňa, prípadne viacerých dní.
TitleStyle	Formát záhlavia mesačného kalendára.
TodayDayStyle	Formát pre zobrazenie dnešného dátumu.
WeekendDayStyle	Formát pre zobrazenie víkendových dní.

Uvedieme jednoduchý príklad použitia elementu DayStyle pre definíciu zobrazenia dní a ak chceme zvlášť zvýrazniť víkendové dni, využijeme element WeekendDayStyle.

```
<asp:Calendar ID="Calendar1" runat="server">
       <DayStyle
          BackColor="lightcyan"
          BorderWidth="2"
          BorderStyle="Groove">
       </DayStyle>

<WeekendDayStyle
          BackColor="Blue"
          ForeColor="Yellow">
</WeekendDayStyle>

</asp:Calendar>
```



prispôsobenie vzhľadu ovládacieho prvku Calendar

Čo sa týka obsluhy udalostí, u ovládacieho prvku typu kalendár sa jedná hlavne o zistenie dátumu, prípadne dátumového intervalu, ktorý bol vybraný používateľom.

Pre monitorovanie zmeny stavu ovládacieho prvku Calendar slúžia udalosti

DayRender	udalosť nastane po vykreslení kalendára
SelectionChanged	udalosť nastane pri výbere konkrétneho dňa klientom
VisibleMonthChanged	udalosť nastane pri zmene aktuálneho mesiaca

Vytvoríme jednoduchú procedúru **Selection_Change**, ktorá sa automaticky vyvolá pri vzniku udalosti **SelectionChanged**.

```
<html><head>
   <script language="C#" runat="server">
      void Selection_Change(Object sender, EventArgs e)
      {
        Label1.Text = "Udalos nastane " + Cal1.SelectedDate.ToShortDateString();
      }
   </script>

</head><body>
   <form runat="server">
      <asp:Calendar ID="Cal1" runat="server"
            SelectionMode="Day"
            ShowGridLines="True"
            OnSelectionChanged="Selection_Change">

            <SelectedDayStyle
              BackColor="Yellow"
              ForeColor="Red">
            </SelectedDayStyle>

      </asp:Calendar>
```

```
      <hr><br>
      <asp:Label id="Label1" runat=server />
   </form>
</body></html>
```

Klientovi bude potom každý jeho výber dňa potvrdený výpisom na HTML stránke (pomocou ovládacieho prvku Label)

výber konkrétneho dňa

Na tomto mieste si mnohí čitatelia, ktorí migrujú z ASP povedia: „Aha, Active- X komponenta. Keď už takýto potenciálne nekompatibilný prvok použili, mohli ho aspoň dizajnovo a funkčne vyšperkovať " Omyl. Všetky stránky vytvorené a vyrendrované pomocou technológie ASP.NET sú „čisté“ HTML stránky bez akýchkoľvek prvkov, ktoré by mohli spôsobovať potenciálnu nekompatibilitu. Nie sú tu nijaké ActiveX komponenty, Java applety, len čistý HTML kód, prípadne nejaké Java Scripty s ktorým si rovnako dobre poradí akýkoľvek prehliadač webových stránok.“ Pozrime sa pre zaujímavosť na zdrojový kód vygenerovanej stránky.

```
...
<td align="Center" style="width:70%;">apríl 2002</td>
<td align="Right" style="width:15%;"><a href="javascript:__doPostBack('Cal1','V851')"
style="color:Black">&gt;</a></td></tr>
       </table></td></tr><tr>
<td align="Center">po</td>
<td align="Center">ut</td>
<td align="Center">st</td>
...
```

Už z tohto malého fragmentu zdrojového kódu je zrejmé, že sa jedná o čistý HTML kód pre vytvorenie tabuľky s použitím JavaScriptu, teda zrozumiteľný pre akýkoľvek prehliadač.

Table

Veľmi používaným zobrazovacím prvkom na webových stránkach sú tabuľky. Komponenta Table umožňuje programovo vytvárať rôzne tabuľky a manipulovať s nimi pomocou ovládacieho prvku Table. S týmto ovládacím prvkom úzko súvisia ďalšie dva ovládacie prvky **TableRow** a **TableCell** pre vytvorenie a manipuláciu riadkov a buniek tabuľky

Definícia syntaxe

```
<asp:Table id="Table1"
      BackImageUrl="url"
      CellSpacing="cellspacing"
      CellPadding="cellpadding"
      GridLines="None|Horizontal|Vertical|Both"
      HorizontalAlign="Center|Justify|Left|NotSet|Right"
      runat="server">

   <asp:TableRow>
      <asp:TableCell>
         Cell text
      </asp:TableCell>
   </asp:TableRow>

</asp:Table>
```

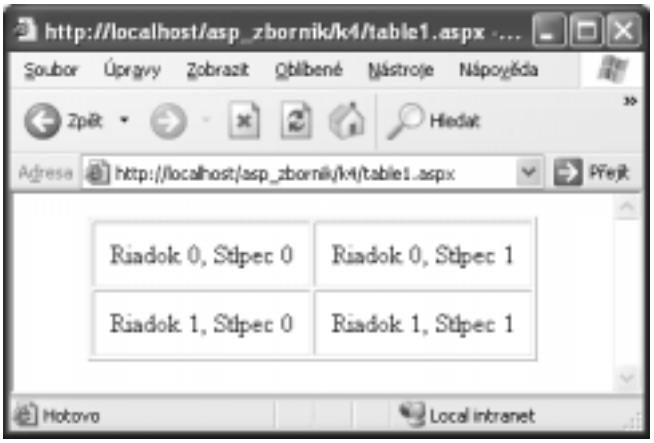
Kód vytvorený Web Matrixom po presunutí komponenty na pracovnú plochu

```
      <form runat="server">
         <asp:Table id="Table1" runat="server"></asp:Table>
         <!-- Insert content here -->
     </form>
```

Najskôr ukážeme príklad pre vytvorenie jednoduchej „statickej" tabuľky o rozmeroch 2 x 2.

```
<%@ Page Language="C#" %>
 <html><body>
 <form runat=server>
 <asp:Table id="Table1" runat="server"
    CellPadding=10
    GridLines="Both"
    HorizontalAlign="Center">
    <asp:TableRow>
       <asp:TableCell> Riadok 0, Stlpec 0 </asp:TableCell>
       <asp:TableCell> Riadok 0, Stlpec 1 </asp:TableCell>
    </asp:TableRow>
    <asp:TableRow>
       <asp:TableCell> Riadok 1, Stlpec 0 </asp:TableCell>
       <asp:TableCell> Riadok 1, Stlpec 1 </asp:TableCell>
    </asp:TableRow>
 </asp:Table>
 </form>
 </body></html>
```

Na HTML stránke u klienta sa zobrazí jednoduchá tabuľka



Tabuľka 4x4 vytvorená pomocou komponenty Table

Znalci syntaxe jazyka HTML si asi pri pohľade na tento kód nepomyslia nič úctivé, rovnaký výsledok sa dá dosiahnuť pomocou jednoduchého kódu (pozrite si zdrojový kód vygenerovanej stránky)

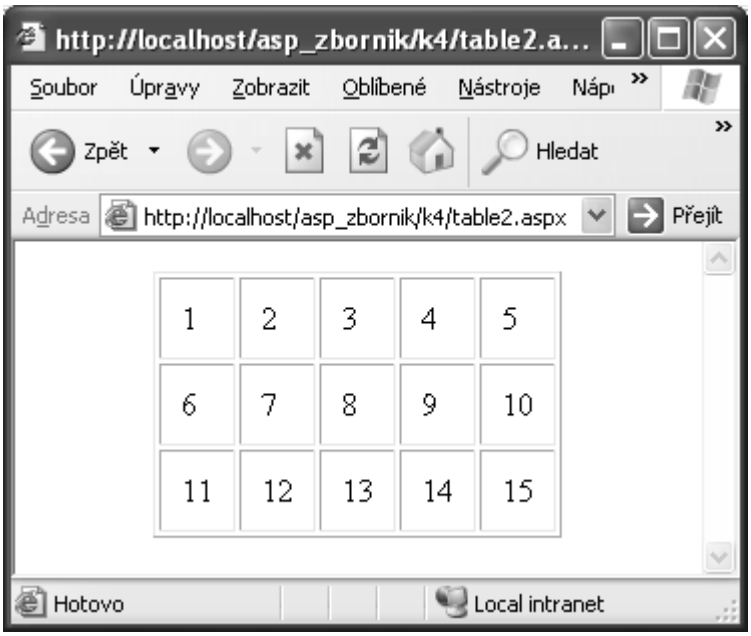
```
<html><body>
 <form name="_ctl0" method="post" action="table1.aspx" id="_ctl0">
<input type="hidden" name="__VIEWSTATE" value="dDwtODIyNzU3MTUyOzs+38i/Y4zieNklikYZbNp3aiE3rhA="
/>

 <table id="Table1" cellpadding="10" align="Center" rules="all" border="1">
       <tr><td> Riadok 0, Stlpec 0 </td><td> Riadok 0, Stlpec 1 </td></tr>
        <tr><td> Riadok 1, Stlpec 0 </td><td> Riadok 1, Stlpec 1 </td></tr>
    </table>
 </form>
 </body></html>
```

Výhody ovládacích prvkov **Table**, **TableRow** a **TableCell** sa naplno prejavia až pri dynamickom generovaní tabuľky a jej obsahu. Ukážeme jednoduchý príklad pre vygenerovanie tabuľky o rozmeroch 3 x 5, ktorá bude obsahovať zaradom očíslované bunky

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

tabuľka je vytvorená pomocou zdrojového kódu

```
<%@ Page Language="C#" %>
<html><head>
    <script runat="server">

        void Page_Load(Object sender, EventArgs e)
        {
            int nRiadkov = 3;
            int nBuniek = 5;
            int nPocitadlo = 1;

            for (int j=0; j<nRiadkov; j++)
            {
                TableRow r = new TableRow();
                for (int i=0; i<nBuniek; i++)
                {
                    TableCell c = new TableCell();
                    c.Text=nPocitadlo.ToString();
                    nPocitadlo++;
                    r.Cells.Add(c);
                }
                Table1.Rows.Add(r);
            }
        }

    </script>
 </head><body>
    <form runat="server">
     <asp:Table id="Table1" runat="server"
        CellPadding=10
        GridLines="Both"
        HorizontalAlign="Center">
    </asp:Table>
 </form>
 </body> </html>
```



Tabuľka dynamicky vytvorená pomocou komponenty Table

Vlastnosti ovládacieho prvku Table

Vlastnosť	Popis
BackImageUrl	Adresa obrázku, ktorý bude zobrazený na pozadí tabuľky
CellPadding	Vnútorné okraje buniek tabuľky
CellSpacing	Medzery medzi bunkami
GridLines	Štýl, pre zobrazenie mriežky tabuľky. (Both, Horizontal, None, Vertical).
HorizontalAlign	Horizontálne zarovnanie tabuľky na HTML stránke
Rows	Vráti objekt TableRowCollection, ktorý obsahuje všetky bunky tabuľky.

Vlastnosti ovládacieho prvku TableRow

Vlastnosť	Popis
Cells	Vracia objekt TableCellCollection, ktorý obsahuje všetky bunky príslušného riadku
HorizontalAlign	Horizontálne zarovnanie obsahu v bunkách riadku
VerticalAlign	Vertikálne zarovnanie obsahu v bunkách riadku.

Vlastnosti ovládacieho prvku TableCell

Vlastnosť	Popis
RowSpan	Výška bunky vyjadrená počtom riadkov, ktoré prekrýva
ColSpan	Šírka bunky vyjadrená počtom stĺpcov, ktoré prekrýva
Text	Text v bunke
Wrap	Zalamovanie textu
HorizontalAlign	Horizontálne zarovnanie textu v bunke tabuľky
VerticalAlign	Vertikálne zarovnanie textu v bunke tabuľky.

V doterajšom prehľade sa vyskytli niektoré komponenty, ktoré bolo možné používať jednak sólo Zo zložitejších ovládacích prvkov nám zostali prvky DataGrid a DataList, ktoré preberieme po vysvetlení filozofie prístupu k údajom v databázach ADO.NET

Validátory

Na formuláre na webových stránkach sú kladené určité požiadavky, hlavne vzhľadom na komfort používateľov. Hlavne v aplikáciách typu e-business je dôležité, aby údaje zadané používateľom (údaje, rodné číslo, číslo objednávky, číslo kreditnej karty....) Od bežného klienta, napríklad návštevníka internetového obchodu samozrejme nemôžeme chcieť, aby poznal všetky záludnosti našej aplikácie. Pre vývojára je „samozrejmé" že sa bude používať desatinná bodka, matematicky „odchovaný“ klient neprogramátor tam celkom pochopiteľne zadá čiarku, problémy sú zo zadávaním formátu dátumu a času a podobne. Preto musíme používateľovi pomôcť nielen vhodným a jednoznačným návrhom formulára pre zadávanie údajov, ale aj zadávané údaje kontrolovať a používateľa usmerňovať tak, aby výsledkom jeho snaženia bola úspešná transakcia. Na vytvorenie interaktívnej stránky sme pred érou .NET mali k dispozícii skripty na strane servera a taktiež skripty na strane klienta. Teoreticky by stačilo použiť len skripty na strane servera, ale to by vyžadovalo neustálu interakciu medzi klientom a serverom pri vyplňovaní a kontrole každého vstupného poľa formulára. Skripty na strane klienta na túto úlohu vždy nestačia, pretože niekedy je potrebné kontrolovať údaje aj na duplicitu. Ideálnym riešením je kombinácia skriptov na strane klienta so skriptami na strane servera. V ASP.NET pre tento účel slúžia objekty zvané validátory.

Validátory je možné do aplikácie vložiť metódou drag and drop z Toolbox vývojového prostredia WebMatrix

Validátory dostupné cez Toolbox vývojového prostredia WebMatrix

Najskôr uvedieme prehľad validátorov vo forme tabuľky

Typ validácie	Validátor
Nutnosť zadať hodnotu	**RequiredFieldValidator**
Porovnanie zadanej hodnoty	**CompareValidator**
Kontrola rozsahu	**RangeValidator**
Kontrola vzorov v reťazcoch	**RegularExpressionValidator**
Používateľsky definovaný	**CustomValidator**

RequiredFieldValidator
Ešte predtým, než budeme kontrolovať, či používateľ zadal správne údaje, začneme tou najjednoduchšou a najtriviálnejšou úlohou, totiž či používateľ zadal vôbec nejaké údaje. Hodí sa to napríklad pri zadávaní mena, kde iné kontrolné kritériá pravdepodobne ani neprichádzajú do úvahy. Princíp činnosti je tohoto validátora je veľmi jednoduchý. RequiredFieldValidator kontroluje neprázdnosť zadávacieho poľa.

Definícia syntaxe

```
<asp:RequiredFieldValidator
      id="ID validátora"
      ControlToValidate=" ID prvku, ktorého obsah chceme kontrolova□"
      InitialValue="hodnota"
      ErrorMessage="Oznam, ktorý sa vypíše v prípade neplatnej validácie"
      Text="text prvku"
      ForeColor="hodnota"
      BackColor="hodnota" …
      runat="server" >
</asp:RequiredFieldValidator>
```

Kód vytvorený Web Matrixom po presunutí validátora na pracovnú plochu

```
      <form runat="server">
         <asp:RequiredFieldValidator id="RequiredFieldValidator1" runat="server"
ErrorMessage="RequiredFieldValidator"></asp:RequiredFieldValidator>
         <!-- Insert content here -->
     </form>
```

Príklad použitia

Vytvoríme najjednoduchší formulár pre zadanie mena. Pomocou RequiredFieldValidatora budeme kontrolovať, či používateľ do editačného okna napísal nejaký text, v tomto prípade meno (súbor valid1.aspx)

```
<html>
     <script language="C#" runat="server">
          void Kliknutie(Object Src, EventArgs E)
          {
               Message.Text = "Zadali ste meno: " + Name.Text;
          }
     </script>

     <body>
        <h1>Pr9klad pre RequiredFieldValidator</font></h1>
        <form action="controls3.aspx" runat=server>
           Meno: <asp:textbox id="Name" runat=server/>
                 <asp:button text="Potvrď"
                       Onclick=" Kliknutie " runat=server/>
                 <asp:label id="Message" runat=server/>
                 <asp:RequiredFieldValidator id="RequiredFieldValidator1"
                       ControlToValidate="Name"
                       Text="Zadaje svoje meno!"
                       runat="server"/>
        </form>
     </body>
</html>
```

Najskôr si overíme fungovanie nášho prvého použitého validátora a potom sa vrátime k princípu a syntaxi. V prípade, že používateľ nechá nevyplnené políčko Name, validátor v tomto prípade vypíše varovný oznam Zadajte svoje meno!.

RequiredFieldValidator

CompareValidator

Tento validátor kontroluje zadanú hodnotu s referenčnou, prípadne s hodnotou iného poľa. Notoricky známym a v praxi veľmi často používaným príkladom je overovanie hesla pri jeho zadaní, či sme sa napríklad nedopustili preklepu, ktorý by nám mohol v budúcnosti veľmi znepríjemniť život. Všeobecný syntaktický predpis pre použitie tohoto validátora je nasledovný:

Definícia syntaxe

```
<asp:CompareValidator
      id="ID validátora"
      ControlToValidate="ID prvku, ktorého obsah chceme porovnáva"
      ValueToCompare="hodnota"
      ControlToCompare="hodnota"
      Type="dátový typ"
      Operator="hodnota operátora"
      ErrorMessage="Oznam, ktorý sa vypíše v prípade neplatnej validácie"
      Text="text prvku"
      ForeColor="hodnota"
      BackColor="hodnota" …
      runat="server" >
</asp:CompareValidator>
```

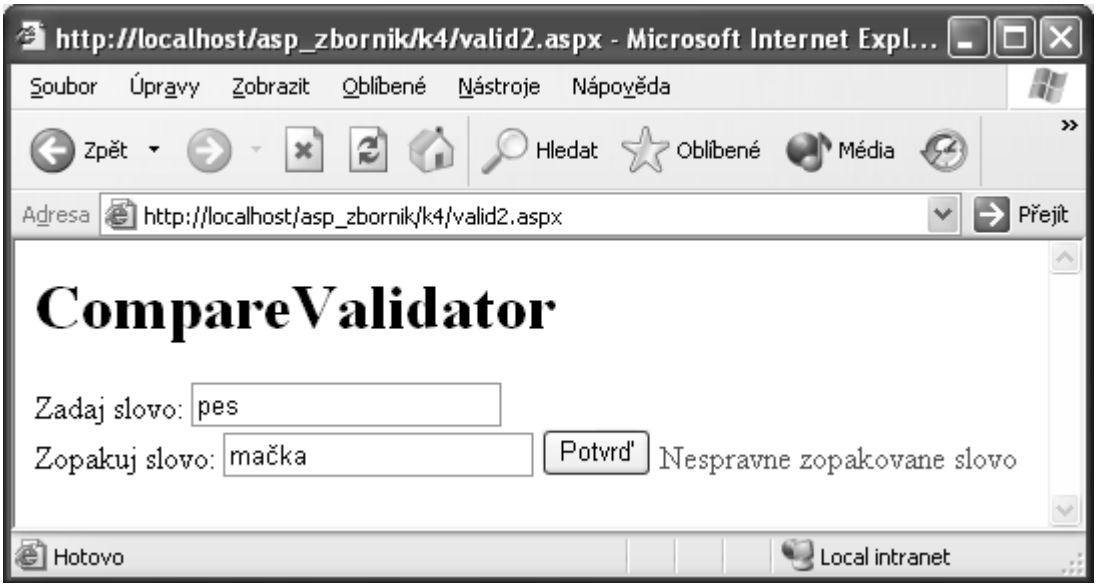
Kód vytvorený Web Matrixom po presunutí validátora na pracovnú plochu

```
      < <form runat="server">
         <asp:CompareValidator id="CompareValidator1" runat="server"
ErrorMessage="CompareValidator"></asp:CompareValidator>
         <!-- Insert content here -->
     </form>
```

Príklad použitia

V tomto príklade vzájomne porovnávame dve zadané hodnoty, napríklad pri overovaní hesla. (*súbor valid2.aspx*):

```
<html>
     <body>
        <h1>CompareValidator</font></h1>
        <form action="controls3.aspx" runat=server>
           Zadaj slovo:    <asp:textbox id="Heslo1" runat=server/><br>
           Zopakuj slovo:  <asp:textbox id="Heslo2" runat=server/>
                            <asp:button text="Potvrď" runat=server/>

                  <asp:CompareValidator id="Compare1"
                        ControlToValidate="Heslo1"
                        ControlToCompare="Heslo2"
                        Type="String"
                        EnableClientScript="false"
                        Text="Nespravne potvrdene heslo"
                        runat="server"/>
        </form>
     </body>
</html>
```



CompareValidator

V tomto príklade sme ukázali porovnanie hodnôt dvoch prvkov medzi sebou. Častým prípadom bude určite porovnanie zadaného obsahu textového prvku s konštantou, alebo s obsahom premennej. V ďalšom príklade ukážeme komplexné využitie CompareValidatora. Je to akási testovacia aplikácia pre overenie vlastností tohoto validátora (*súbor valid3.aspx*).

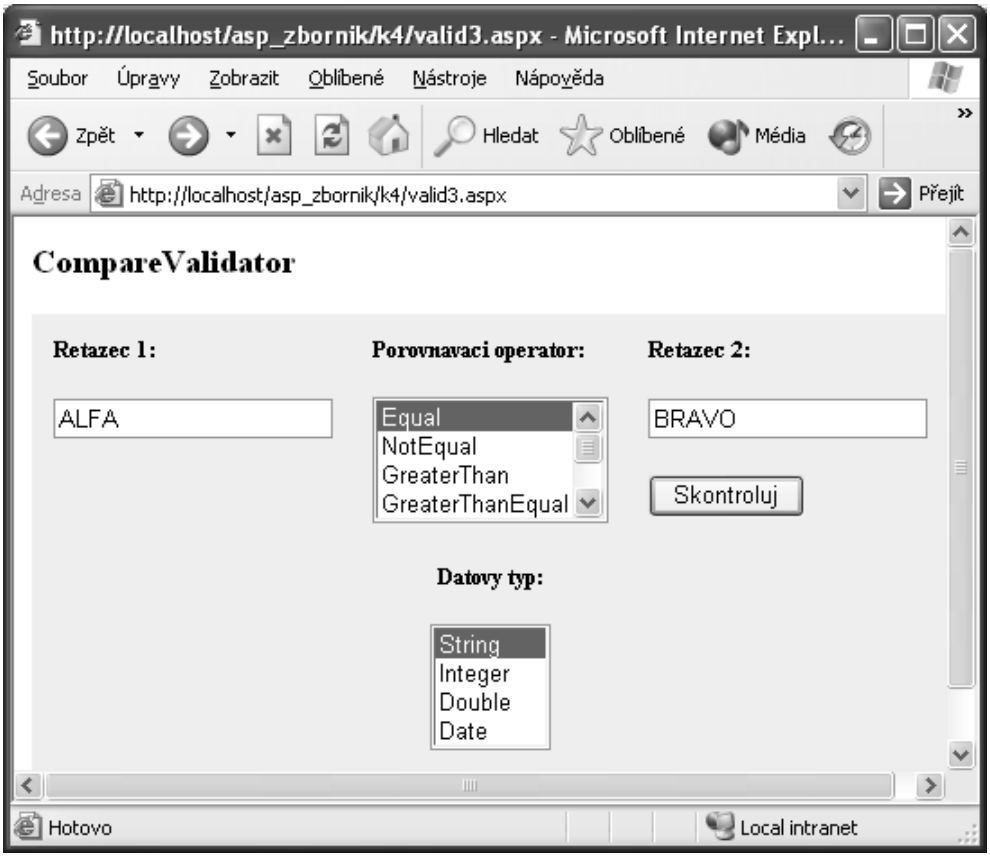
V záhlaví ASP.NET stránky sú použité tri procedúry. Procedúra Button_Click "zviditeľní" rezultát validátora. Pomocou procedúry Operator_Index_Changed nastavíme príslušný operátor ako parameter validátora a následne validátor aktivujeme. Nastavenie požadovaného dátového typu a následnú aktiváciu validátora má na starosti procedúra Type_Index_Changed (*súbor valid3.aspx*).

```
<%@ Page Language="C#" %>
<html><head><script runat="server">

       void Button_Click(Object sender, EventArgs e)
       {// *************
          if (Page.IsValid) {lblOutput.Text = "Rovnost!";}
          else              {lblOutput.Text = "Nerovnost!";}

       }

       void Operator_Index_Changed(Object sender, EventArgs e)
       {//  **********************
        // Aktivuj validator pri zmene operatora
          Compare1.Operator = (ValidationCompareOperator)ListOperator.SelectedIndex;
          Compare1.Validate();

       }

       void Type_Index_Changed(Object sender, EventArgs e)
       {//  **********************
        // Aktivuj validator pri zmene datoveho typu
          Compare1.Type = (ValidationDataType)ListType.SelectedIndex;
          Compare2.Type = (ValidationDataType)ListType.SelectedIndex;
          Compare1.Validate();

       }
   </script></head>
```

V tele stránky sú inicializované a naplnené dva prvky typu ListBox, jeden pre výber operátora a druhý pre výber dátového typu.

```
<body>
   <form runat="server">
      <h3>CompareValidator</h3>
      <table bgcolor="#eeeeee" cellpadding="10">
      <tr valign="top">
         <td>
            <h5>Retazec 1:</h5>
            <asp:TextBox id="TextBox1" runat="server"/><br>
            <asp:CompareValidator id="Compare1"
                  ControlToValidate="TextBox1"
                  ControlToCompare="TextBox2"
                  Type="String"
                  EnableClientScript="false"
                  Text="Failed Validation"
                  runat="server"/>
         </td>
         <td>
            <h5>Porovnavaci operator:</h5>
            <asp:ListBox id="ListOperator"
                  OnSelectedIndexChanged="Operator_Index_Changed"
                  runat="server">
           <asp:ListItem Selected Value="Equal" >Equal</asp:ListItem>
           <asp:ListItem Value="NotEqual" >NotEqual</asp:ListItem>
           <asp:ListItem Value="GreaterThan" >GreaterThan</asp:ListItem>
           <asp:ListItem Value="GreaterThanEqual" >GreaterThanEqual </asp:ListItem>
           <asp:ListItem Value="LessThan" >LessThan</asp:ListItem>
           <asp:ListItem Value="LessThanEqual" >LessThanEqual </asp:ListItem>
           <asp:ListItem Value="DataTypeCheck" >DataTypeCheck</asp:ListItem>
           </asp:ListBox>
           </td>

           <td>
            <h5>Retazec 2:</h5>
            <asp:TextBox id="TextBox2" runat="server"/><br>
            <asp:CompareValidator id="Compare2"
                  ControlToValidate="TextBox2"
                  Operator="DataTypeCheck"
                  EnableClientScript="false"
                  Text="Invalid Data Type"
                  runat="server"/>
            <br>
            <asp:Button id="Button1"
                  Text="Skontroluj"
                  OnClick="Button_Click"
                  runat="server"/>
        </td>
      </tr>
      <tr>
         <td colspan="3" align="center">
            <h5>Datovy typ:</h5>
            <asp:ListBox id="ListType"
                  OnSelectedIndexChanged="Type_Index_Changed" runat="server">
               <asp:ListItem Selected Value="String" >String</asp:ListItem>
               <asp:ListItem Value="Integer" >Integer</asp:ListItem>
               <asp:ListItem Value="Double" >Double</asp:ListItem>
```

```
                <asp:ListItem Value="Date" >Date</asp:ListItem>
                </asp:ListBox>
            </td>
        </tr>
    </table>
    <br>

    <asp:Label id="lblOutput"
          Font-Name="verdana"
          Font-Size="10pt"
          runat="server"/>

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



Testovacia stránka pre CompareValidator

RangeValidator

U číselných hodnôt veľmi často potrebujeme skontrolovať, či je zadaná hodnota v určitom číselnom intervale, ktorý je obvykle daný aplikačnou logikou.

Definícia syntaxe

```
<asp:RangeValidator
      id="ID alidatora"
      ControlToValidate=" ID prvku, ktorého rozsah chceme kontrolova "
      MinimumValue="hodnota"
      MaximumValue="hodnota"
      Type="dátový typ"
      ErrorMessage=" Oznam, ktorý sa vypíše v prípade neplatnej validácie "
      Text="text prvku"
      ForeColor="hodnota"
```

```
      BackColor="hodnota" …
      runat="server" >
</asp:RangeValidator>
```

Kód vytvorený Web Matrixom po presunutí validátora na pracovnú plochu

```
      <form runat="server">
         <asp:RangeValidator id="RangeValidator1" runat="server"
ErrorMessage="RangeValidator"></asp:RangeValidator>
         <!-- Insert content here -->
     </form>
```

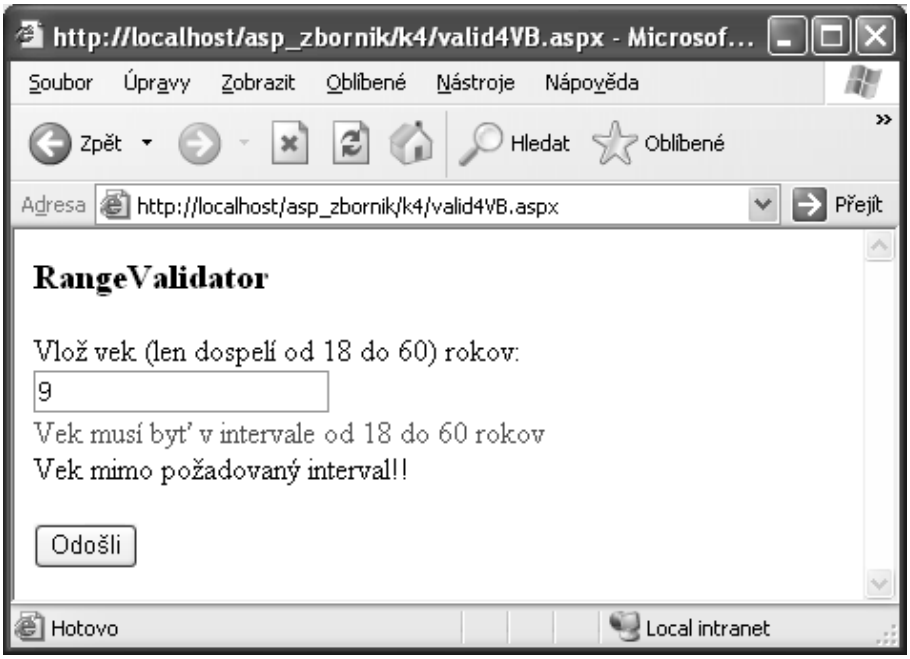
Príklad použitia
[Visual Basic] (*súbor valid4VB.aspx*)

```
<%@ Page Language="VB" AutoEventWireup="True" %>
<html><head>
   <script runat="server">
      Sub ButtonClick(sender As Object, e As EventArgs)
         If Page.IsValid Then
            Label1.Text="Správne zadaný vek."
         Else
            Label1.Text="Vek mimo požadovaný interval!!"
         End If
      End Sub
   </script>

</head><body>
   <form runat="server">
      <h3>RangeValidator</h3>
      Vlož vek (len dospelí od 18 do 60) rokov: <br>
      <asp:TextBox id="TextBox1" runat="server"/><br>
      <asp:RangeValidator id="Range1"
            ControlToValidate="TextBox1"
            MinimumValue="18"
            MaximumValue="60"
            Type="Integer"
            EnableClientScript="false"
            Text="Vek musí by□ v intervale od 18 do 60 rokov"
            runat="server"/><br>

      <asp:Label id="Label1" runat="server"/><br><br>
      <asp:Button id="Button1"
            Text="Odošli"
            OnClick="ButtonClick"
            runat="server"/>
   </form>
</body></html>
```

[C#] (*súbor valid4VS.aspx*)

```
<%@ Page Language="C#" AutoEventWireup="True" %>
<html><head>
   <script runat="server">
      void ButtonClick(Object sender, EventArgs e)
      {
         if (Page.IsValid) {Label1.Text="Správne zadaný vek.";}
```

```
            else                {Label1.Text="Vek mimo požadovaný interval!!";}
        }
    </script>

</head><body>
    <form runat="server">
        <h3>RangeValidator</h3>
        Vlož vek (len dospelí od 18 do 60) rokov:<br>

        <asp:TextBox id="TextBox1" runat="server"/>
        <asp:RangeValidator id="Range1"
              ControlToValidate="TextBox1"
              MinimumValue="18"
              MaximumValue="60"
              Type="Integer"
              EnableClientScript="false"
              Text="Vek musí by□ v intervale od 18 do 60 rokov!"
              runat="server"/><br><br>

        <asp:Label id="Label1" runat="server"/><br><br>

        <asp:Button id="Button1"
              Text="Submit"
              OnClick="ButtonClick"
              runat="server"/>
    </form>
</body></html>
```



RangeValidator

RegularExpressionValidator

Ako vyplýva z názvu, tento validátor slúži pre overenie, či zadaná hodnota vyhovuje regulárnemu výrazu. Typickým príkladom regulárneho výrazu je napríklad mailová adresa.

Definícia syntaxe

```
<asp:RegularExpressionValidator
      id="ID validátora"
      ControlToValidate=" ID prvku, ktorého obsah chceme kontrolova□"
```

```
      ValidationExpression="výraz"
      ErrorMessage="Oznam, ktorý sa vypíše v prípade neplatnej validácie"
      Text="text prvku"
      ForeColor="hodnota"
      BackColor="hodnota" …
      runat="server">
</asp: RegularExpressionValidator>
```

Kód vytvorený Web Matrixom po presunutí validátora na pracovnú plochu

```
      <form runat="server">
         <asp:RegularExpressionValidator id="RegularExpressionValidator1" runat="server"
ErrorMessage="RegularExpressionValidator"></asp:RegularExpressionValidator>
         <!-- Insert content here -->
     </form>
```

Predpis pre overenie regulárneho výrazu sa vytvára pomocou kombinácie špeciálnych znakov. Ich kompletný zoznam je v dokumentácii. Vo forme tabuľky ukážeme niekoľko najpoužívanejších.

Znak	Popis
*	Žiadny znak, prípadne viac znakov.
+	Jeden alebo viac znakov.
\	Prefix pred špeciálnym znakom, napríklad '\n' znamená znak CR. Používa sa aj pre aplikáciu špeciálnych znakov ako bežných
.	Jeden ľubovoľný znak okrem CR.
\d	Číslica.
\D	Iný znak ako číslica
\t	Tabulátor.

Príklad použitia (kontrola používateľom zadanej mailovej adresy)
Kompletná tabuľka obsahuje niekoľko desiatok riadkov, no popis bez praktickej ukážky by nebol príliš účelný. Ukážeme to radšej na niekoľkých príkladoch. Predpis pre kontrolu mailovej adresy bude:
'.*@.*\..*', po preklade do ľudskej reči to znamená:

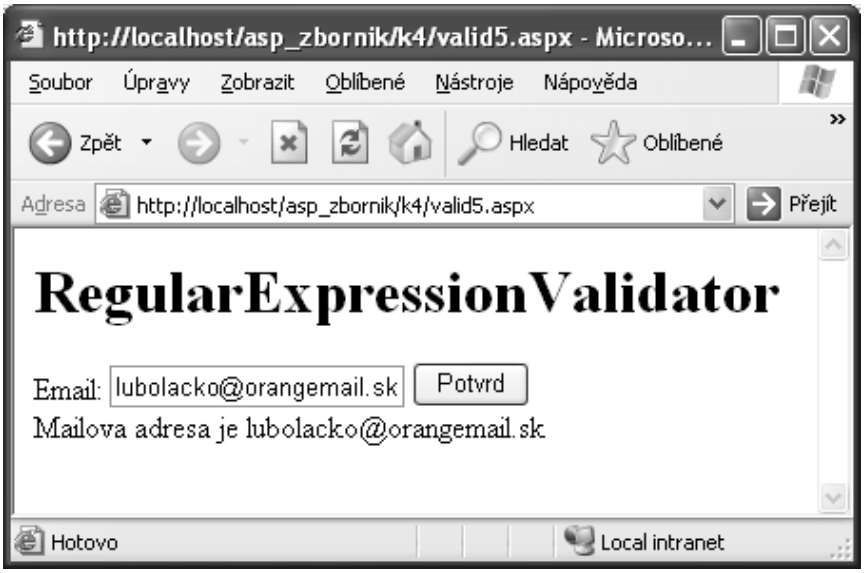
. jeden ľubovoľný znak
* ľubovoľný počet znakov (aj žiadne).
@ konkrétny znak, v našom prípade populárny zavináč
. jeden ľubovoľný znak
* ľubovoľný počet znakov (aj žiadne).
\. konkrétny znak, v tomto prípade bodka. Pretože bodka je aj špeciálny znak, je pred ňou lomítko
. jeden ľubovoľný znak
* ľubovoľný počet znakov (aj žiadne).

Reťazec teda musí začínať jedným, alebo viacerými znakmi, za nimi musí byť znak zavináč, pokračujeme ďalším minimálne jednoznakovým reťazcom za ktorým musí byť bodka. Mailová adresa končí ukončovacím, minimálne jednoznakovým reťazcom. (s*úbor valid5.aspx*)

```
<%@ Page Language="c#" %>
<script runat="server">
     void Kliknutie(Object Src, EventArgs E)
     {
          Message.Text = "Mailova adresa je " + Mail.Text;
     }
</script>

<html><head></head>
<body>
     <h1>RegularExpressionValidator
```

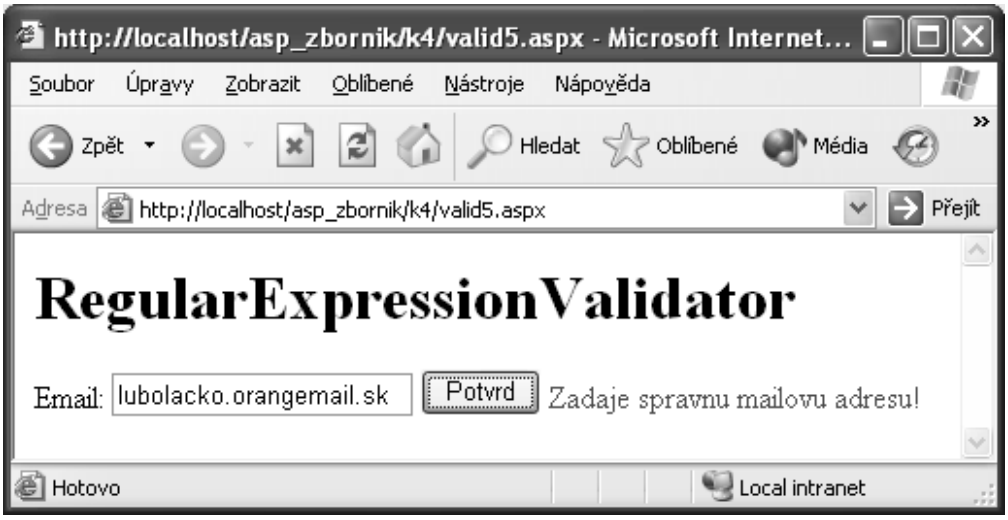
```
    </h1>
    <form action="controls3.aspx" runat="server">
        Email:
        <asp:textbox id="Mail" runat="server"></asp:textbox>
        <asp:button id="Button1" onclick=" Kliknutie " runat="server" text="Potvrd"
Width="62px"></asp:button>
        <asp:label id="Message" runat="server"></asp:label>
        <asp:RegularExpressionValidator
             id="RegularExpressionValidator1"
             runat="server" Text="Zadaje spravnu mailovu adresu!"
             ControlToValidate="Mail"
             ValidationExpression='.*@.*\..*'>
        </asp:RegularExpressionValidator>
    </form>
</body></html>
```

Skript funguje veľmi jednoducho. Ak zadáme do editačného okna mailovú adresu v správnom formáte, napríklad lubolacko@orangemail.sk po zatlačení tlačidla POTVRĎ, bude táto adresa vypísaná.



príklad so správne zadanou mailovou adresou

Ak však zadáme hodnotu, ktorá pravidlám nevyhovuje, validator okamžite zasiahne.



príklad s nesprávne zadanou emailovou adresou

Custom Validator

Doteraz preberané validátory boli svojimi tvorcami navrhnuté tak, aby pomáhali validovať najčastejšie sa vyskytujúce situácie. Veľakrát sme postavení pred situáciu, validovať nejakú súvislosť, ktorá vyplýva z aplikačnej logiky. Potrebujeme napríklad skontrolovať rodné číslo, číslo kreditnej karty a podobne. Napríklad Rodné číslo v Českej aj Slovenskej republike sa zadáva v tvare napríklad 650327/6153. Kontrolný mechanizmus u rodných čísel osôb po roku asi 1953 spočíva v kontrole deliteľnosti celého rodného čísla jedenástimi. Podobné závislosti platia aj pre základnú kontrolu čísla kreditnej karty a podobne. Nejedná sa o nijakú podrobnú kontrolu a už vôbec nie validáciu, no už aj táto jednoduchá kontrola v prípade rodného čísla s pomerne vysokou pravdepodobnosťou odhalí náhodne zadané rodné číslo a samozrejme pomôže aj odhaliť preklep pri jeho zadávaní.

Definícia syntaxe

```
<asp:CustomValidator
      id="ID validátora"
      ControlToValidate=" ID prvku, ktorého obsah chceme kontrolova□"
      ClientValidationFunction="ID validačnej funkcie u klienta"
      OnServerValidate="Validačná_funkcia"
      ErrorMessage="Oznam, ktorý sa vypíše v prípade neplatnej validácie"
      Text="text prvku"
      ForeColor="hodnota"
      BackColor="hodnota" …
      runat="server" >
</asp:CustomValidator>
```

Kód vytvorený Web Matrixom po presunutí validátora na pracovnú plochu

```
      <form runat="server">
         <asp:TextBox id="TextBox1" runat="server"></asp:TextBox>
         <asp:CustomValidator id="CustomValidator1" runat="server"
ErrorMessage="CustomValidator"></asp:CustomValidator>
         <!-- Insert content here -->
     </form>
```

Pre napísanie vlastnej validačnej funkcie máme dve možnosti. Prostredníctvom parametra ClientValidationFunction naviažeme validátor na validačnú funkciu u klienta, ktorá býva napísaná obvykle v JavaScripte, alebo VB Scripte.

Validačná funkcia v JavaScripte:

```
Function ValidationFunctionName (source, arguments)
```

Validačná funkcia vo VBScripte:

```
Sub ValidationFunctionName (source, arguments)
```

Prostredníctvom parametra OnServerValidate môžeme naviazať validátor na validačnú funkciu na serveri.

Aj syntaktické formy pre serverové validačné funkcie sú predpísané. Pre obidva najpoužívanejšie programovacie jazyky Visual Basic a C# sú tieto formy nasledovné:

[Visual Basic]

```
Sub ServerValidation (source As object, args As ServerValidateEventArgs)
   args.IsValid = True/Talse
End Sub
```

[C#]

```
void ServerValidation (object source, ServerValidateEventArgs args)
{
   args.IsValid = true/false;
}
```

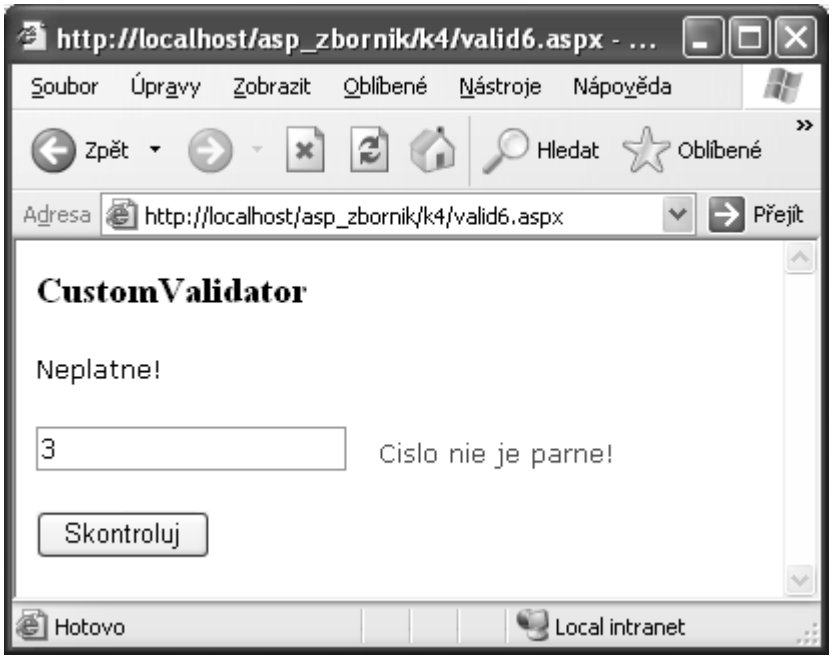
Ako príklad sme vybrali jednoduchý test, či je zadané číslo párne. Validačná funkcia v tomto prípade pobeží na serveri.

```
    void ServerValidation (object source, ServerValidateEventArgs arguments)
    {
       int i = int.Parse(arguments.Value);
       arguments.IsValid = ((i%2) == 0);
    }
```

Kompletný kód jednoduchej miniaplikácie na testovanie skutočnosti, či bolo zadané párne číslo je nasledovný (validačná funkcia a CustomValidator sú v zdrojovke zvýraznené):

```
<%@ Page Language="C#" %>
<script runat="server">

    void ValidateBtn_OnClick(object sender, EventArgs e)
    {
       if(Page.IsValid) {lbl.Text = "Platne!";}
       else             {lbl.Text = "Neplatne!";}
    }

    void ServerValidation (object source, ServerValidateEventArgs arguments)
    {
       int i = int.Parse(arguments.Value);
       arguments.IsValid = ((i%2) == 0);
    }

</script>
<html><head></head>
<body>
    <form runat="server">
        <h3>CustomValidator</h3>
        <asp:Label id="lbl" runat="server" Font-Size="10pt" Font-Name="Verdana" Text="Zadajte
parne cislo:"></asp:Label>
        <br/>
        <p><asp:TextBox id="Text1" runat="server"></asp:TextBox>   
        <asp:CustomValidator
           id="CustomValidator1" runat="server"
           Font-Size="10pt" Font-Name="verdana" ForeColor="green"
           ErrorMessage="Cislo nie je parne!"
           Display="Static" OnServerValidate="ServerValidation"
           ControlToValidate="Text1">
           </asp:CustomValidator></p>
        <p><asp:Button id="Button1" onclick="ValidateBtn_OnClick" runat="server"
Text="Skontroluj"></asp:Button></p>
    </form>
</body></html>
```

Skúška validátora je v tomto prípade veľmi jednoduchá.



Custom validator - test na párne číslo

ValidationSummary Control
Zatiaľ sme testovali jednotlivé validátory izolovane po jednom. Ak však navrhujeme formulár na ASP.NET stránke, ktorý má klient vyplniť, tento formulár obsahuje spravila viac zadávacích prvkov. Keďže základom stabilnej a spoľahlivej aplikácie je určitá paranoja voči používateľovi, spravidla kontrolujeme väčšinu zadaných údajov. Každý z údajov však pritom môže byť ošetrený validátorom iného typu. Pomocou prvku ValidationSummary dokážeme vypísať oznamy všetkých validátorov na jednom mieste.

Definícia syntaxe

```
<asp:ValidationSummary
  id=" ID prvku"
  DisplayMode="BulletList | List | SingleParagraph"
  EnableClientScript="true | false"
  ShowSummary="true | false"
  ShowMessageBox="true | false"
  HeaderText="Text ktorý sa zobrazí v záhlaví výpisu oznamov validátorov"
  runat="server"/>
```

Funkciu tohto validátora najlepšie pochopíme na príklade jednoduchého formulára s dvoma polami pre zadanie údajov. Na každé z polí je naviazaný RequiredField Validator. V tomto prípade okrem správne vyplneného formulára, teda takého kde sú vyplnené obidva polia sa môže používateľ dopustiť troch druhov chýb. Môže vyplniť len jedno z polí, prípadne môže nechať obidva polia nevyplnené. Túto situáciu riešime pomocou ValidationSummary

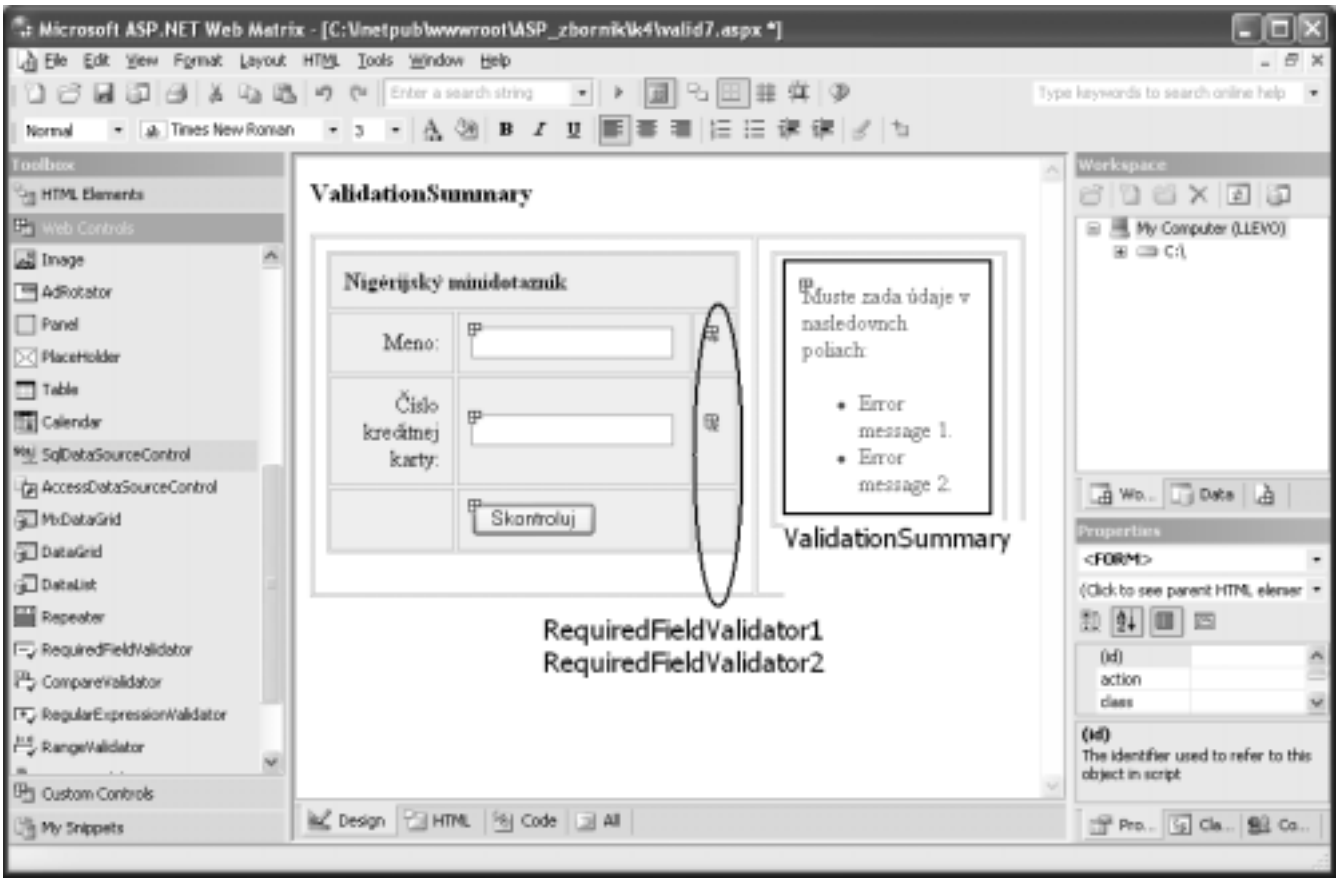
Príklad použitia (*súbor valid7.aspx*):
Vo Web Matrixe vytvoríme formulár s dvoma textovými poliami,

```
<asp:TextBox id="TextBox1"
     runat="server">
</asp:TextBox>
```

pričom na každé z nich bude naviazaný samostatný RequiredField Validator.

```
<asp:RequiredFieldValidator id="RequiredFieldValidator1"
     runat="server" Text="*" Width="100%" InitialValue=""
     Display="Static" ErrorMessage="Meno."
     ControlToValidate="TextBox1">
</asp:RequiredFieldValidator>
```

Validátor umiestnime vedľa editačného poľa a ako text jeho chybovej správy dáme napríklad hviezdičku (*), ktorá bude v návrhovom zobrazení prítomnosť validátora identifikovať



Vytvorenie príkazu na Validation Summary

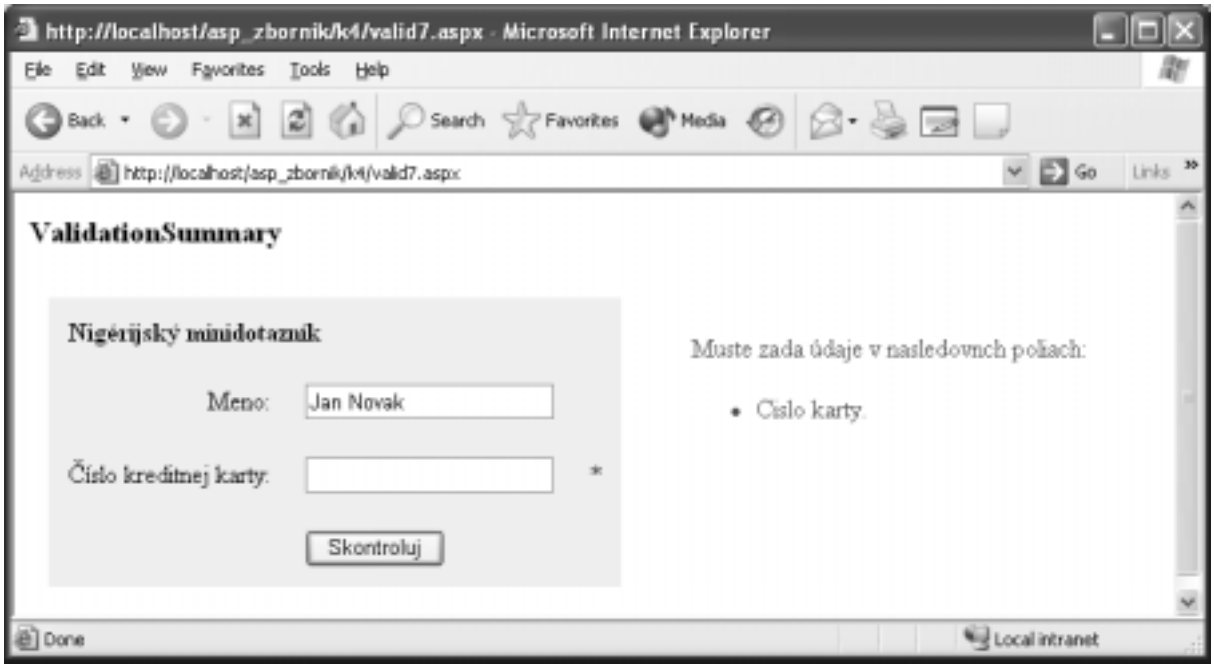
Výpis chybových stavov bude mať na starosti komponenta ValidationSummary,

```
<asp:ValidationSummary id="valSum" runat="server"
     HeaderText="Muste zada údaje v nasledovnch poliach:"
     EnableClientScript="true" DisplayMode="BulletList">
</asp:ValidationSummary>
```

Kompletný kód miniaplikácie potom bude:

```
<html><head></head>
<body><h3>ValidationSummary</h3><p></p>
    <form runat="server">
        <table cellpadding="10"><tbody>
            <tr><td><table cellpadding="10" bgcolor="#eeeeee"><tbody>
            <tr><td colspan="3"><b>Nigérijský minidotazník</b></td></tr>

            <tr><td align="right">Meno:</td>
                <td><asp:TextBox id="TextBox1" runat="server"></asp:TextBox></td>
                <td align="middle" rowspan="1">
                    <asp:RequiredFieldValidator id="RequiredFieldValidator1"
                        runat="server" Text="*" Width="100%" InitialValue=""
                        Display="Static" ErrorMessage="Meno."
                        ControlToValidate="TextBox1">
                     </asp:RequiredFieldValidator></td></tr>
```

```
        <tr><td align="right">Číslo kreditnej karty: </td>
                 <td><asp:TextBox id="TextBox2" runat="server"></asp:TextBox></td>
                 <td><asp:RequiredFieldValidator id="RequiredFieldValidator2"
                          runat="server" Text="*" Width="100%" Display="Static"
                           ErrorMessage="Cislo karty. " ControlToValidate="TextBox2">
                      </asp:RequiredFieldValidator></td></tr>
          <tr><td></td>
          <td><asp:Button id="Button1" runat="server" Text="Skontroluj"></asp:Button></td>
          <td></td></tr>
          </tbody></table></td>

          <td valign="top"><table cellpadding="20"><tbody>
              <tr><td><asp:ValidationSummary id="valSum" runat="server"
                           HeaderText="Muste zada údaje v nasledovnch poliach:"
                           EnableClientScript="true" DisplayMode="BulletList">
                      </asp:ValidationSummary></td></tr>
            </tbody></table></td></tr>
        </tbody></table>
    </form>
</body></html>
```



Validation Summary

Vývoj aplikácií pre mobilné zariadenia

Hneď na úvod dôležité upozornenie. Pre vývoj webových aplikácií pre mobilné zariadenia je potrebné nainštalovať ***Microsoft Mobile Internet Toolkit****. Microsoft Mobile Internet Toolkit sa ištaluje ako doplnok ke staršej verzii Visual Studia .NET, V najnovšej verzii Visual Studia .NET 2003 je už priamo implementovaná technológioa* ***ASP.NET Mobile Controls****, ktorá je implementovaná aj do .NET Frameworku 1.1. MMIT je možné stiahnuť z webu Microsoftu, konkrétne zo sekcie Download z adresy http://download.microsoft.com/download/VisualStudioNET/InstalI/RC/NT45XP/EN-US/MobileIT.exe. Netreba mať ani prílišné obavy z jeho sťahovania, nakoľko súbor má len niečo vyše 4MB.*

Pre vývoj a ladenie aplikácií pre platformu Pocket PC budeme ešte potrebovať pravdepodobne ***emulátor Pocket PC 2002*** *(rovnaký ako je dodávaný v eMVT). Tento emulátor môžeme voľne stiahnuť z webovej adresy http://www.microsoft.com/mobile/downloads/emvt30.asp alebo z inej webovej adresy http://support.microsoft.com/directory/article.asp?id=kb;en-us;Q296904*
Pre emulovanie WAP telefónov môžeme použiť napríklad Openwave emulátor, ktorý môžeme získať z webu z adresy www.openwave.com

Okrem aplikácií pre "klasického" tenkého klienta teda prehliadač webových stránok bude čoraz častejšie potrebné vyvíjať aj webové aplikácie ku ktorým sa bude pristupovať z mobilných tenkých klientov, teda z prístrojov PDA (osobný digitálny asistent) a mobilných telefónov.
Aby sa nám jednotlivé typy mobilných klientov neplietli uvedieme stručný prehľad platforiem Pocket PC , Pocket PC Phone Edition a Windows Powered Smartphone 2000,.

Platformy mobilných zariadení s operačným systémom od Microsoftu

Pocket PC
Túto platformu azda netreba príliš predstavovať. Je to prístroj s farebným dotykovým displejom a niekoľkými ďalšími ovládacími prvkami. Presadil sa ako personálny digitálny asistent. Rozšírenie komunikačných možností, napríklad o Wireles LAN, Bluetooth, GSM/GPRS docielime externou kartou (PCMCIA, CF,ba dokonca aj SD). Novšie prístroje majú niektoré zo spomínaných komunikačných modulov už vstavané.

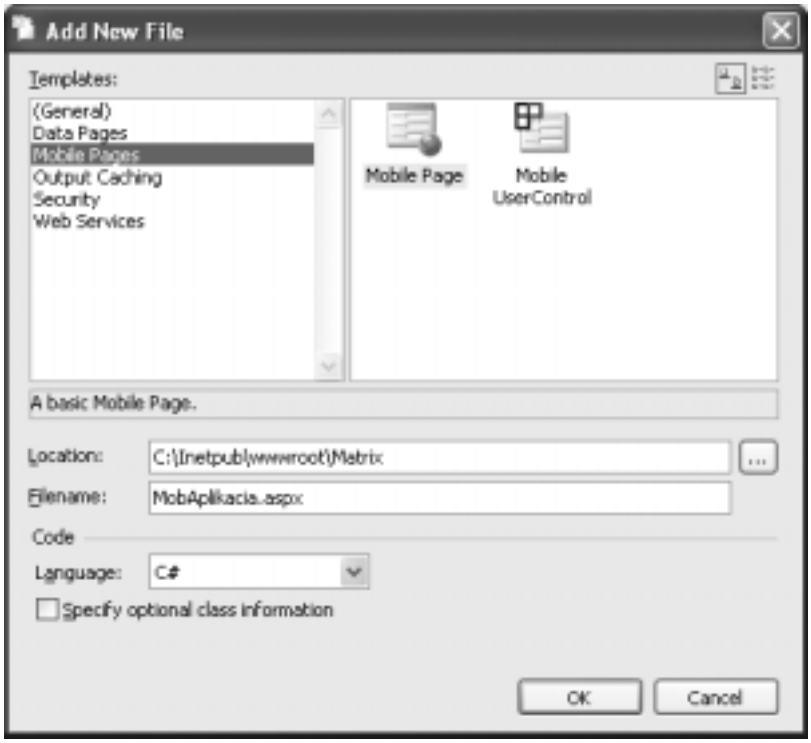
Pocket PC Phone Edition
Zjednodušene sa dá povedať, že prístroj triedy Pocket PC 2002 Phone Edition je klasické PDA s integrovaným GPRS mobilným telefónom. Navonok sa obvykle na prvý pohľad spozná podľa anténky.

Smartphone 2002
Jedná sa o mobilný telefón, teda presnejšie o prístroj ktorý konštrukciou oveľa viac pripomína mobilný telefón než PDA. Platforma Smartphone 2002 vychádza v podstate z Windows CE (Pocket PC) pričom by mala kombinovať to najlepšie z PDA a mobilných telefónov. Táto platforma by mala poskytovať inteligentné spojenie, to jest hlasom, e-mailom alebo inými prostriedkami a čo je najdôležitejšie kdekoľvek a kedykoľvek. Smartphone 2002 okrem elektronickej pošty ponúka aj možnosť prehliadania webového obsahu. Využíva pritom také používateľské rozhraní, aké zákazníci od mobilného telefónu očakávajú. Platforma **Smartphone 2002** nakoľko je implementovaná v mobilných telefónoch má určité špecifiká, hlavným rozdielom oproti Pocket PC 2002 je absencia dotykového displeja. Mobilné telefóny sa totiž ovládajú hlavne klávesnicou a aj tá býva počas nosenia vo vrecku zablokovaná, takže dotykový displej bez odklopného krytu by bol nezmysel. Displej bez možnosti ovládania dotykom môže mať jednak odolné čelné sklo, ale zníži sa tým aj cena zariadenia a jeho spotreba. Rozlíšenie displeja je spravidla 176 x 220 pixelov. Displej môže byť farebný alebo čiernobiely.

Cieľom nášho snaženia bude vývoj serverovej aplikácie s použitím technológie ASP.NET vo vývojovom prostredí Web Matrix

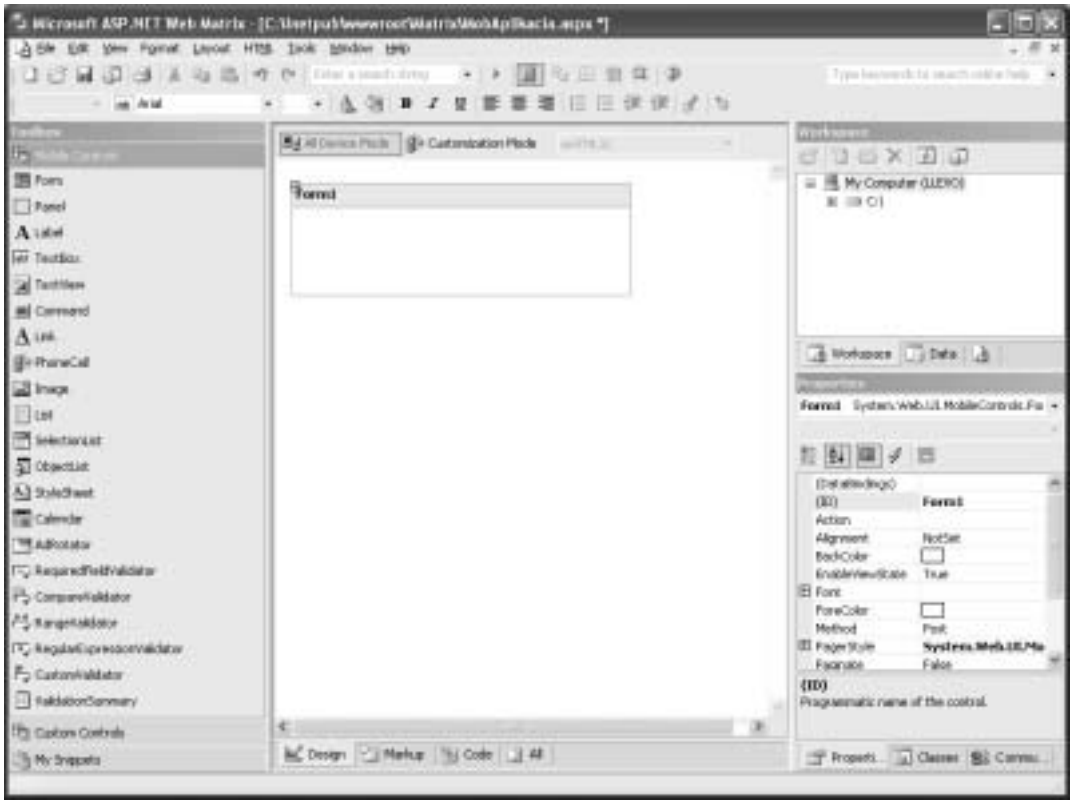
Vytvorenie novej aplikácie typu Single Mobile Page

Novú „mobilnú“ aplikáciu vytvoríme v zložke *Mobile Pages* dialógu *Add New File*. Ako typ projektu zvolíme *Simple Mobile Pages*. Projekt nazveme napríklad MobAplikacia.aspx a súbor s rovnakým názvom bude umiestnený do pracovného adresára Internet Information Servera (IIS), implicitne to bude adresár *C:\Inetpub\wwwroot*. Z uvedených faktov vyplýva, že k tejto mobilnej aplikácii budeme pristupovať na lokálnom počítači cez URL adresu http://localhost/MobAplikacia.aspx .



Vytvorenie projektu typu Simple Mobile Page

V návrhovom zobrazení záložky **Design** môžeme umiestňovať komponenty na vhodné miesto mobilného formulára, prípadne môžeme ich rozmiestnenie kedykoľvek poopraviť. Všimnime si na obrázku že tento formulár je aj v návrhovom zobrazení podstatne menší, vyplýva to pochopiteľne zo zobrazovacích možností mobilných zariadení.



Základná obrazovka vývojového prostredia Web Matrix pri vývoji mobilnej aplikácie

V záložke **HTML** môžeme editovať kód HTML stránky vrátane kódu ASP.NET komponentov.

```
<mobile:Form id="Form1" runat="server">

</mobile:Form>
```

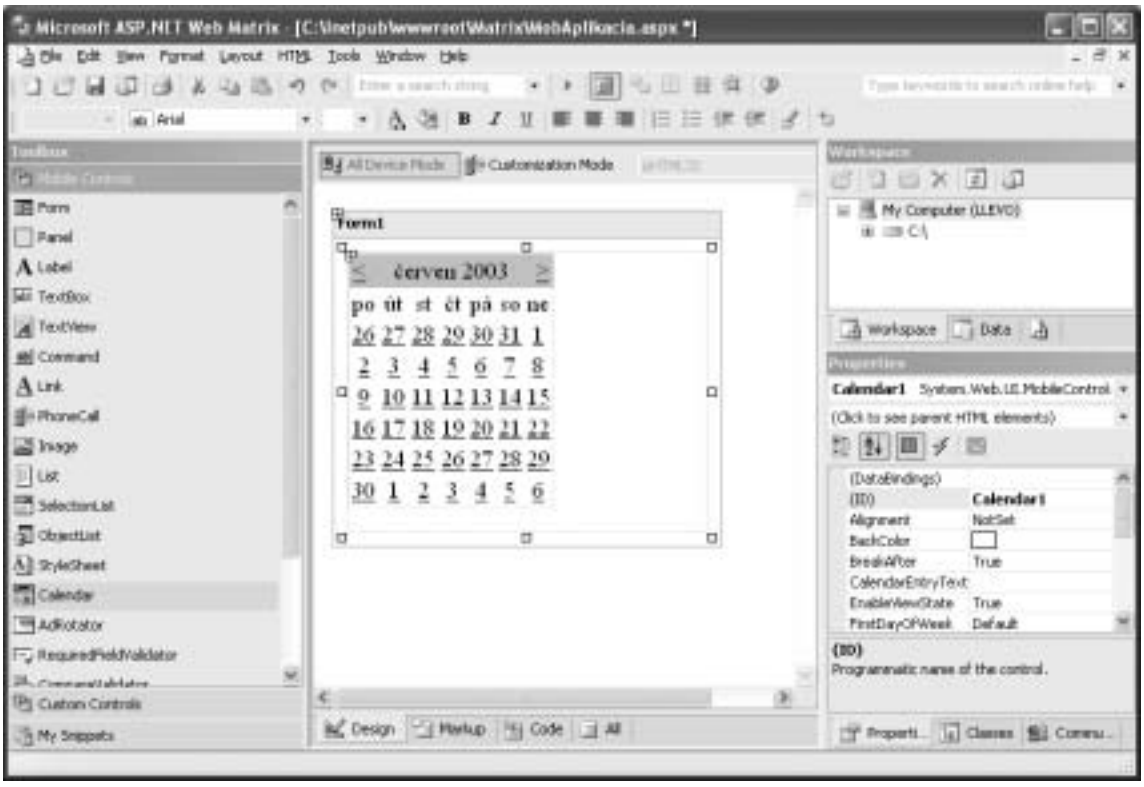
Kód v záhlaví stránky je prístupný cez záložku **Code**. Po vytvorení nového projektu sa na tomto mieste nachádza len komentár (v našom prípade podľa syntaktických pravidiel jazyka C#)

```
// Insert page code here
//
```

S kompletným kódom môžeme pracovať v záložke All.

```
<%@ Page Inherits="Microsoft.Saturn.Framework.Mobile.UI.MobilePage,
Microsoft.Saturn.Framework.Mobile, Version=0.5.464.0, Culture=neutral,
PublicKeyToken=6f763c9966660626" Language="C#" %>
<%@ Register TagPrefix="Mobile" Namespace="System.Web.UI.MobileControls"
Assembly="System.Web.Mobile" %>
<script runat="server">

    // Insert page code here
    //

</script>
<mobile:Form id="Form1" runat="server">

</mobile:Form>
```

Návrh aplikačného formulára
Azda najjednoduchšou a zároveň najpresvedčivejšou ukážkou je komponeta Calendar v kontexte mobilnej webovej stránky.



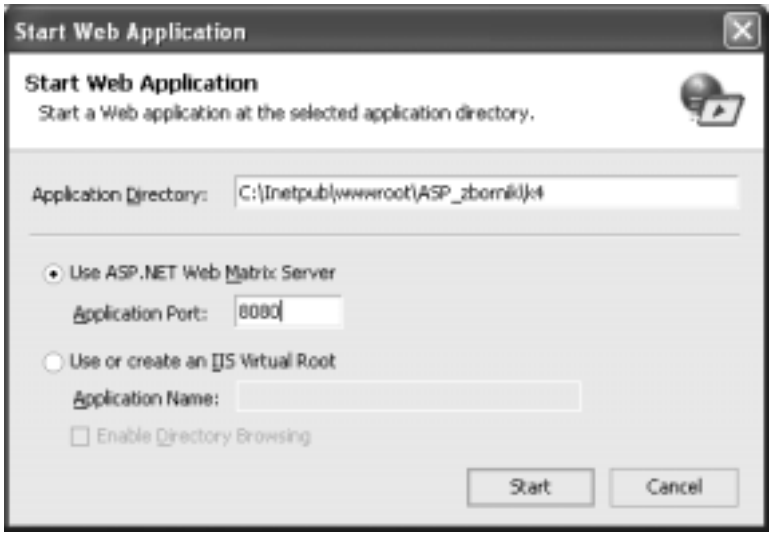
Komponenta Calendar v aplikačnom formulári mobilnej aplikácie

Po vložení komponenty kalendár sa táto komponenta pridá do kódu ASPX stránky takto:

```
<mobile:Form id="Form1" runat="server">
    <Mobile:Calendar id="Calendar1" runat="server"></Mobile:Calendar>
</mobile:Form>
```

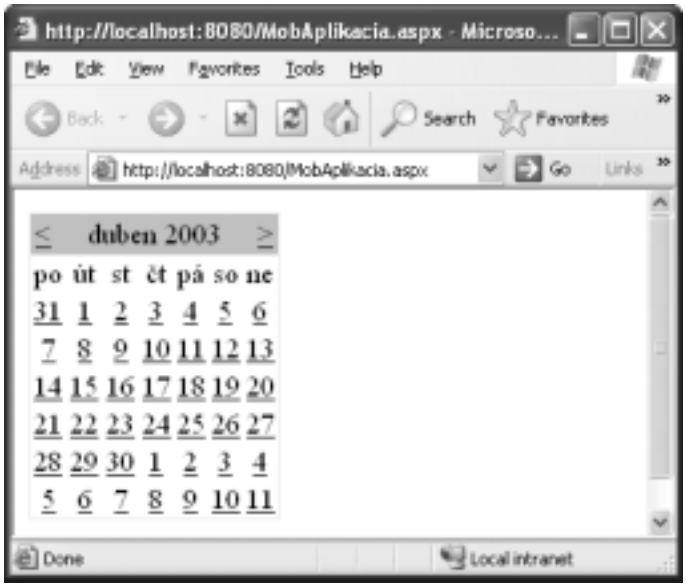
Návrh prvej fázy jednoduchej mobilnej aplikácie máme za sebou. Zostáva nám celú aplikáciu vyskúšať a pri vývoji reálnej aplikácie samozrejme dopísať príslušný kód, ktorý bude tvoriť jadro aplikačnej logiky. Na čo by mohla byť dobrá aplikácia tohoto typu, ktorá zobrazí komponentu kalendár dobrá? Nuž k čomu sa prevažne používajú mobilné zariadenia a mobilné telefóny s podporou protokolu WAP?. Predsa na personálny informačný manažment, plánovanie úloh, plánovanie času, transakcie internetbankingu... Takže náš výber komponenty kalendár nebol ani náhodný ani samoúčelný.

Spustenie a ladenie "mobilnej" aplikácie.
V tejto etape môžeme náš program spustiť. Po aktivovaní menu Web Matrixu Štart sa zobrazí dialóg pre výber web servera. Podobne ako u klasickej webovej aplikácii buď webový server (skôr listener) Microsoft ASP.NET Web Matrix Server, ktorý sa nainštaluje spolu s vývojovým prostredím Web Matrix. Aplikácie bude potom prístupná na URL adrese http://localhost:8080/MobilnaAplikacia.aspx . Druhá možnosť ponúkaná v tomto dialógu je využiť Internet Information Server (IIS).



Obr. 4 Výber webového servera pre ladenie a testovanie aplikácie

O činnosti služby Microsoft ASP.NET Web Matrix Server sme informovaní ikonou v pravej dolnej časti pracovnej obrazovky operačného systému Windows. Takto bude naša aplikácia zobrazená v okne desktopového prehliadača HTML stránok



Komponenta Calendar zobrazená na klasickej HTML stránke

Aplikáciu môžeme samozrejme vyskúšať aj na iných typoch mobilných zariadení, prípadne na ich emulátoroch, napríklad na Pocket PC, Smartphone, alebo aj na emulátore Openwave, ktorý emeulue jednoduchý mobilný telefón (Siemens S45) s podporou protokolu WAP

Komponenta Calendar zobrazená na emulátore mobilného telefónu s podporou WAP (Siemens S45)

Samozrejme aplikácie (hlavne tie zložitejšie a aplikácie, ktoré využívajú komunikáciu alebo pristupujú k hardvéru) môžeme najlepšie otestovať pomocou reálneho zariadenia napríklad triedy Pocket PC pripojeného cez ActiveSync alebo čoraz častejšie aj pomocou Wireles LAN, prípadne pomocou emulátorov platforiem Pocket PC2002 a Smartphone 2002

.

KAPITOLA 5:
Databázové aplikácie

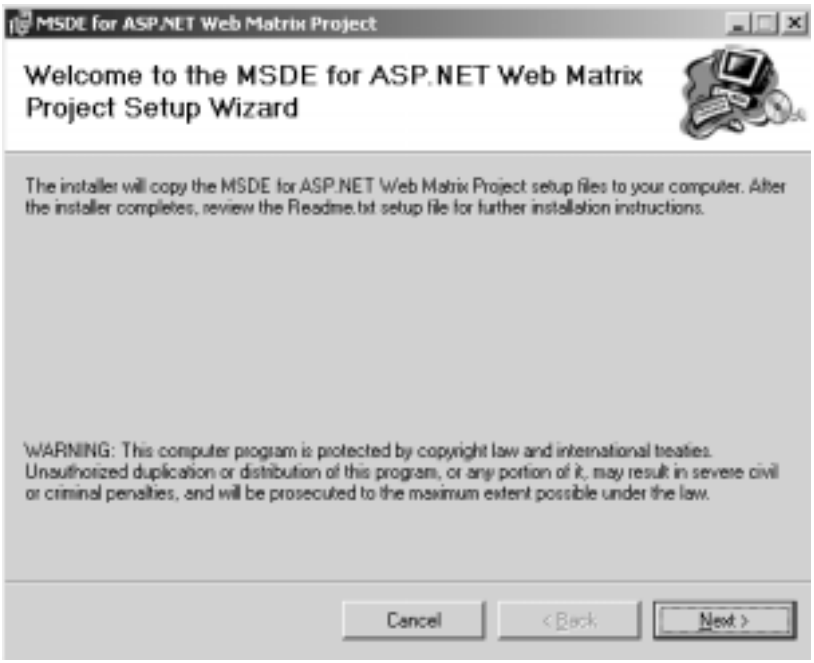
Vo sfére webových aplikácií typu hobby má zatiaľ určité dominantné postavenie voľne šíriteľný databázový server MySQL. Preto sa dá predpokladať, že veľa vývojárov hobby aplikácii, ktorí sa rozhodnú migrovať na ASP.NET majú skúsenosti práve s MySQL. Nie je dokonca nutné sa ho vzdávať, nakoľko existujú verzie MySQL pre operačný systém Windows (za nepatrný licenčný poplatok). Vždy je to lepšie riešenie ako „nešvár“ webových aplikácií, kancelárska databáza Access, prípadne jej výkonné jadro Microsoft Jet engine. Microsoft Jet engine je obsiahnutý aj vo vývojovom prostredí MS Visual Studio 6.0, kde ho môžeme voľne distribuovať s vytvorenými aplikáciami. Údaje sú uložené v súboroch s príponou MDB. Fyzické obmedzenie databázového jadra Microsoft Jet engine je hlavne v maximálnej veľkosti databázového súboru 1 GB. Oveľa väčším problémom je však zlá podpora viacpoužívateľského prístupu (problémy začínajú už pri menej ako 20 prístupoch). Pre webové aplikácie, kedy typicky predpokladáme viacpoužívateľský prístup sa teda Microsoft Jet engine zásadne neodporúča.

MSDE

Oveľa lepšie a hlavne perspektívnejšie riešenie predstavuje MSDE (Microsoft Desktop Engine) Je to voľne šíriteľné, plne funkčné jadro MS SQL Serveru primárne určené pre vývoj a testovanie databázových aplikácií a pri prevádzke databázových aplikácií na lokálnych počítačoch. Hlavné obmedzenie spočíva v možnom prístupe maximálne piatich procesov súčastne. Toto obmedzenie bolo samozrejme implementované úmyselne, aby sa MSDE nenasadzovalo tam kde nie je určené, na výkonné produkčné systémy. Toto obmedzenie sa zdá fatálne, ale nie je tomu tak. Pri dobre navrhnutej a optimalizovanej webovej aplikácii sa dajú obslúžiť stovky prístupov za minútu. Pri aplikáciách typu hobby určite nevadia ani jeho ďalšie obmedzenia (maximálna veľkosť databázových súborov je obmedzená na 2GB vo verzii 7.0, prípadne na 10 GB vo verzii 2000). Tento databázový stroj môžeme teda pokojne použiť aj pre finálnu prevádzku vyvinutej webovej aplikácie. Navyše máme v zásobe ďalšiu ohromnú výhodu, ktorou je stopercentná kompatibilita s Microsoft SQL Serverom. Ak bude naša aplikácia úspešná, môžeme prakticky okamžite vymeniť databázový stroj MSDE za plnú verziu MS SQL Servera. Náš projekt pritom zostane úplne nezmenený.

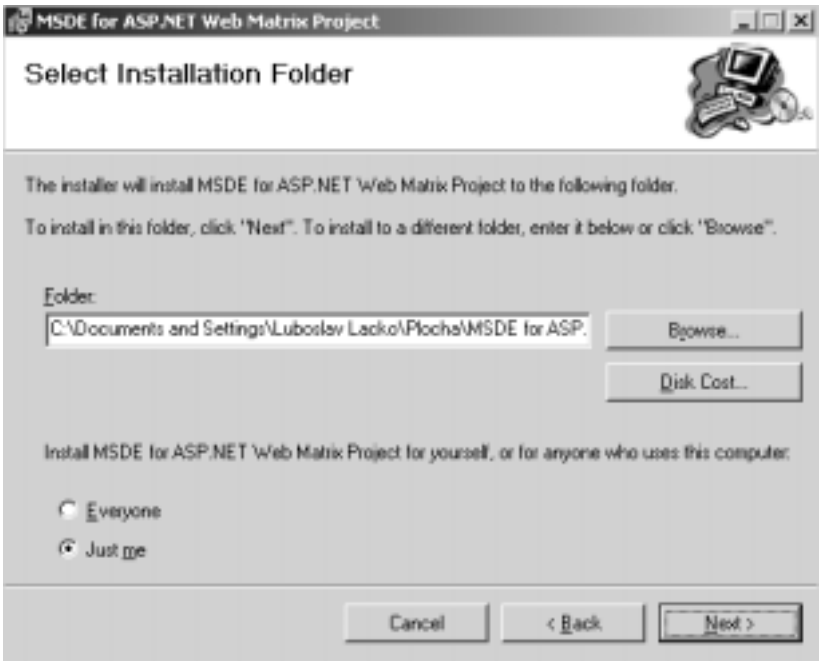
Inštalácia MSDE

Inštalačný súbor MSDE s názvom msde.msi má približne 30 megabajtov. Úvodný dialóg nás informuje, že sa jedná o verziu MSDE pre ASP.NET Web Matrix.



msde1.bmp Inštalácia MSDE

Nie je to až tak zbytočné oznámenie ako by sa mohlo na prvý pohľad zdať. MSDE nie je totiž úplne voľne šíriteľné. Licenciu pre jeho šírenie získame, ak sme predplatiteľmi MSDN (Microsoft Developer Network), prípadne ak vlastníme SQL Server Developer Edition, prípadne máme vývojové prostredie Visual Studio, alebo kancelársky balík MS Office Professional. Všetko pomerne nákladné záležitosti. MSDE však môžeme používať aj ak sme si ho stiahli spolu s voľne šíriteľným vývojovým prostredím Web Matrix.



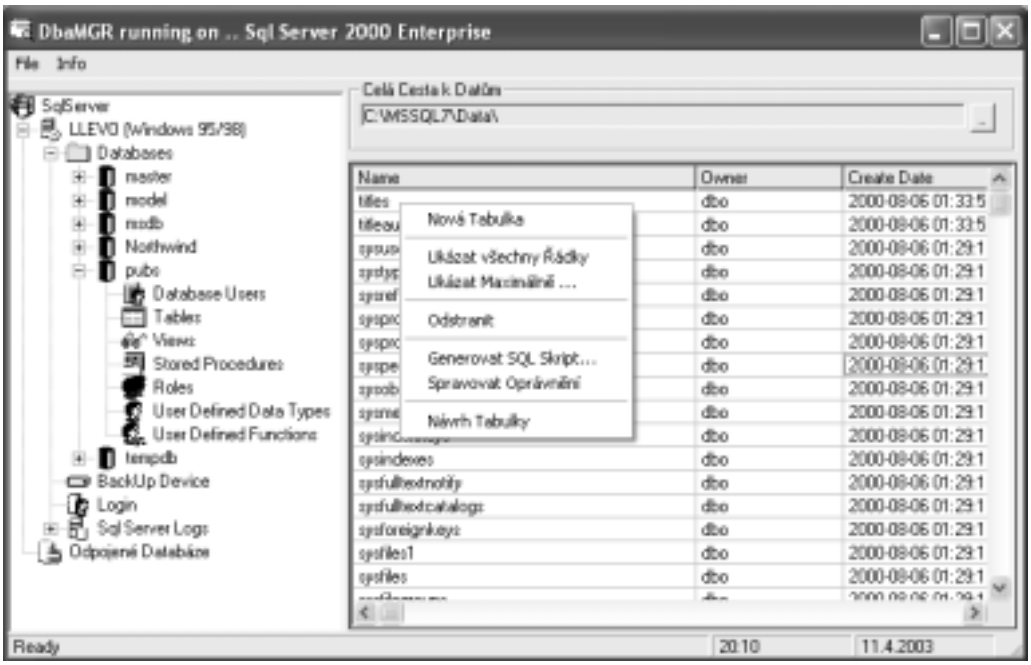
msde2.bmp Inštalácia MSDE

Následne sa môžeme rozhodnúť, či MSDE bude môcť používať každý používateľ PC na ktorom bude MSDE nainštalovaný, alebo len ten používateľ, ktorý ho nainštaloval. Napriek tomu, že na prvý pohľad sa zdá, že produkt MSDE sa po tomto kroku začne fyzicky inštalovať, nie je to celkom presné. Len sa pripravujú inštalačné súbory. Implicitne sú dokonca nainštalované do adresára, ktorý sa vytvorí na pracovnej ploche. To preto, aby sme si ho všimli a spustili tentokrát už fyzickú inštaláciu

Klientské aplikácie pre MSDE
Pre správu databázového serveru a ladenie databázovej časti aplikácií potrebujeme dva typy aplikácií. SQL príkazy môžeme pred ich zakomponovaní do ASP.NET kódu ladiť pomocou **klientskej konzolovej aplikácie**. Náplň jej činnosti je jednoduchá. Slúži pre zadávanie príkazov jazyka SQL databázovému serveru a okne tej istej aplikácie taktiež vidíme výstupy, ktoré databázový server vygeneruje ako odozvu na naše príkazy, teda napríklad výpis obsahu databázových tabuliek, potvrdenie vykonania našich príkazov, chybové hlásenia a podobne. Pre administráciu databázy, napríklad pre nastavovanie prístupových práv potrebujeme **aplikáciu pre správu databázy**. Pomocou nástroja pre správu databázy je možné nastaviť stratégiu údržby a zálohovania údajov v databáze a podobne. Pre prácu s databázovým serverom **Microsoft SQL Server 2000** (teda jeho plnou alebo trial verziou) je určená klientská konzolová aplikácia **Query Analyzer**. Pre administráciu sa používa nástroj **SQL Enterprise Manager**. Tieto nástroje môžeme samozrejme použiť aj pre prácu s MSDE, môžeme si ich nainštalovať z CD 120 dňovej trial verzie SQL Servera 2000. Toto CD bolo pribalené k mnohým knihám a k niektorým PC časopisom.

Na CD k tejto publikácii je pribalených niekoľko klientských aplikácií, dve z nich si v stručnosti predstavíme.

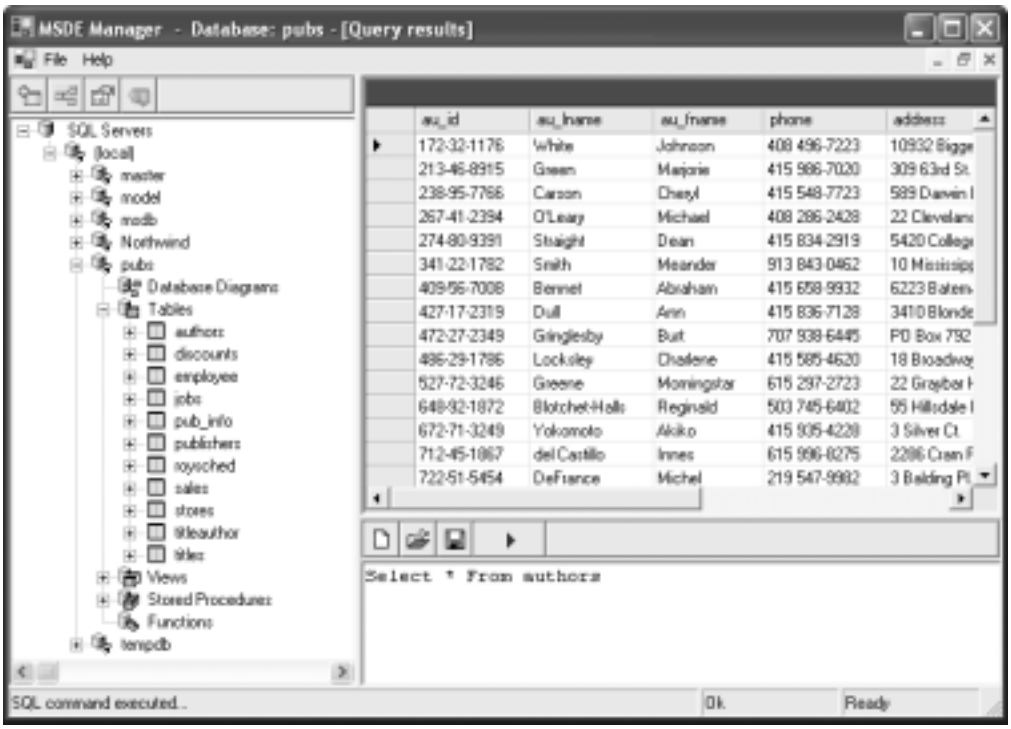
DBA Manager
Je to počeštená, pre nekomerčné aplikácie voľne šíriteľná aplikácia talianskeho autora pre administráciu databázy aj pre zobrazovanie obsahu tabuliek. Je dodávaná vrátane zdrojového kódu, ktorý je možné v prípade potreby modifikovať. (VB6)



DBA Manager

Postup pripojenia na MSDE
Klikneme pravým tlačidlom myši na položku ***SqlServer***
Vyberieme z menu položku ***Spojiť/Editovať Vlastnosti Spojení***
Nastavíme vlastnosti spojenia, hlavne meno servera.
Po nastavení parametrov aktivujeme tlačidlo ***Nadviazať spojenie***
Prípadne môžeme tieto parametre zadať pred spustením programu v súbore DBAMGR.INI

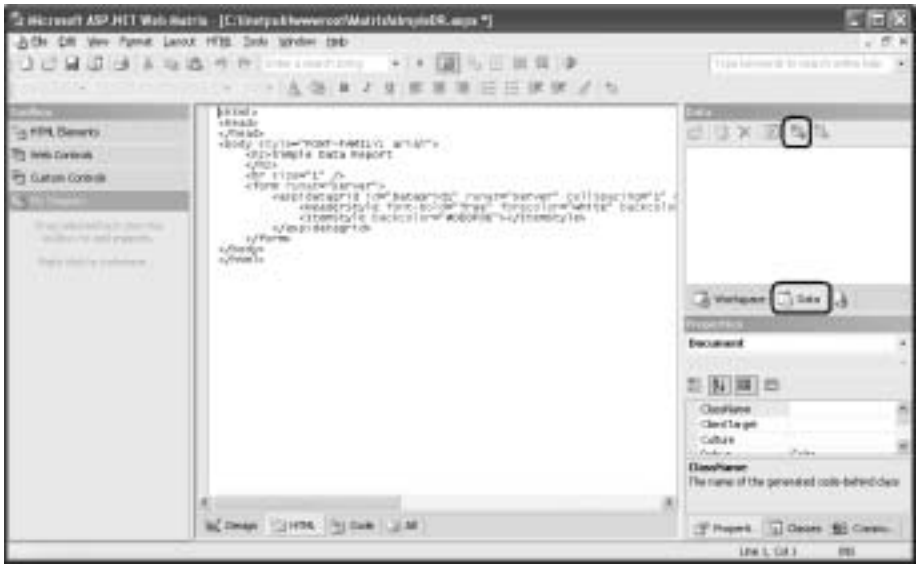
MSDE Manager
MSDE Manager je konzolová aplikácia pre zadávanie a ladenie príkazov jazyka SQL. V spodnej časti pracovnej obrazovky je priestor pre zadanie SQL príkazu, v hornej časti sa zobrazí výpis obsahu tabuľky na základe zadaného príkazu.(VB.NET)



MSDE Manager

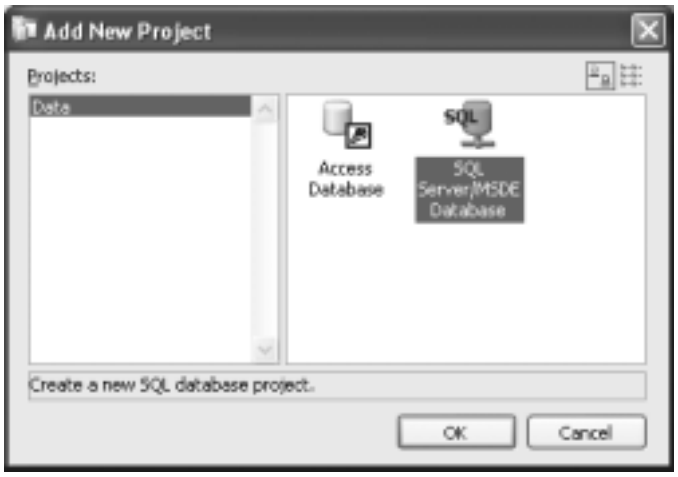
Prístup k databázam cez vývojové prostredie Web Matrix

Pomerne komfortné a komplexné možnosti pre prístup údajov v databázach nám poskytuje aj vývojové prostredie Web Matrix. Kľúčom k prístupu je okno v pravej hornej časti pracovnej plochy, ktoré prepneme na záložku **Data**. Pripojenie k databáze vytvoríme pomocou ikony v ľavej hornej časti toho okna.



Prístup k databázam cez Web Matrix

Pri vývoji databázových projektov vo vývojovom prostredí WebMatrix máme na výber dva druhy databáz. Jednak databázové súbory kancelárskej databázy Access, ktorá sa pre webové projekty nehodí, no napríklad pre intranet menšej firmy to môže byť vyhovujúce a jednak SQL Server, prípadne jeho voľne šíriteľné jadro MSDE. Typ databázy vyberieme pomocou dialógu:

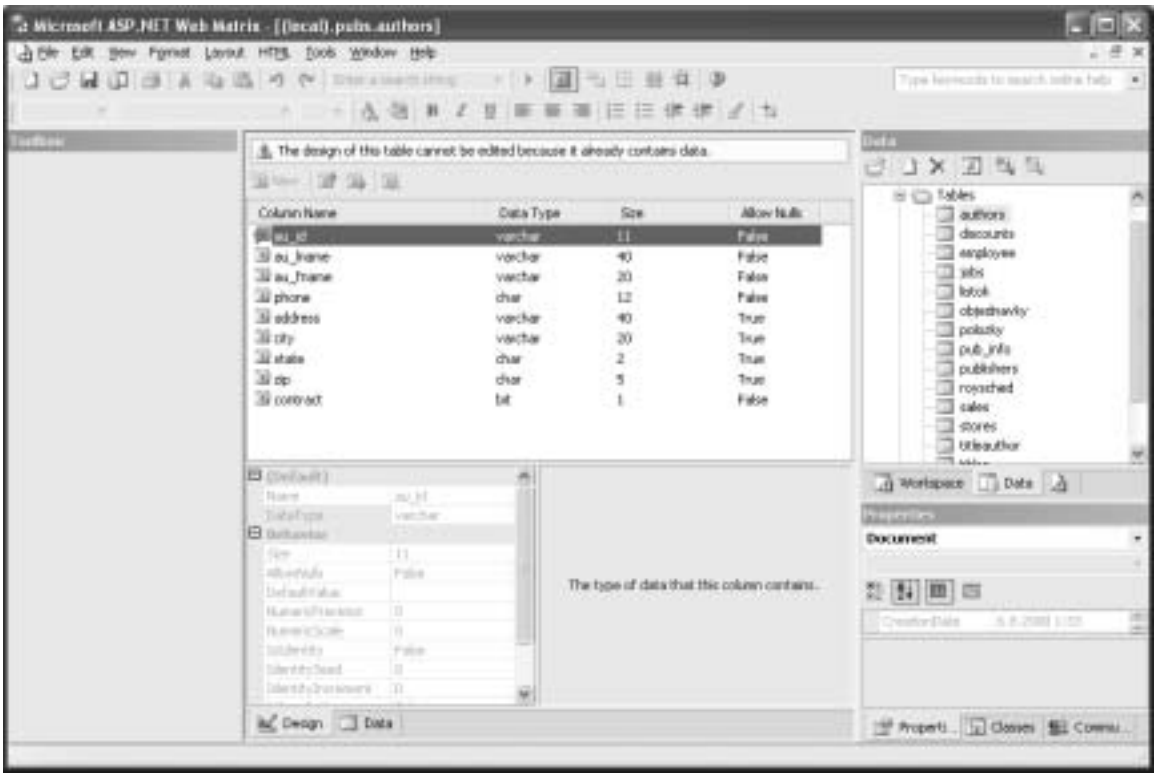


Výber typu databázy

Aktivujeme tým dialóg pre nastavenie parametrov pripojenia, kde zadáme meno databázového servera, spôsob pripojenia, prístupové parametre a meno databázy.

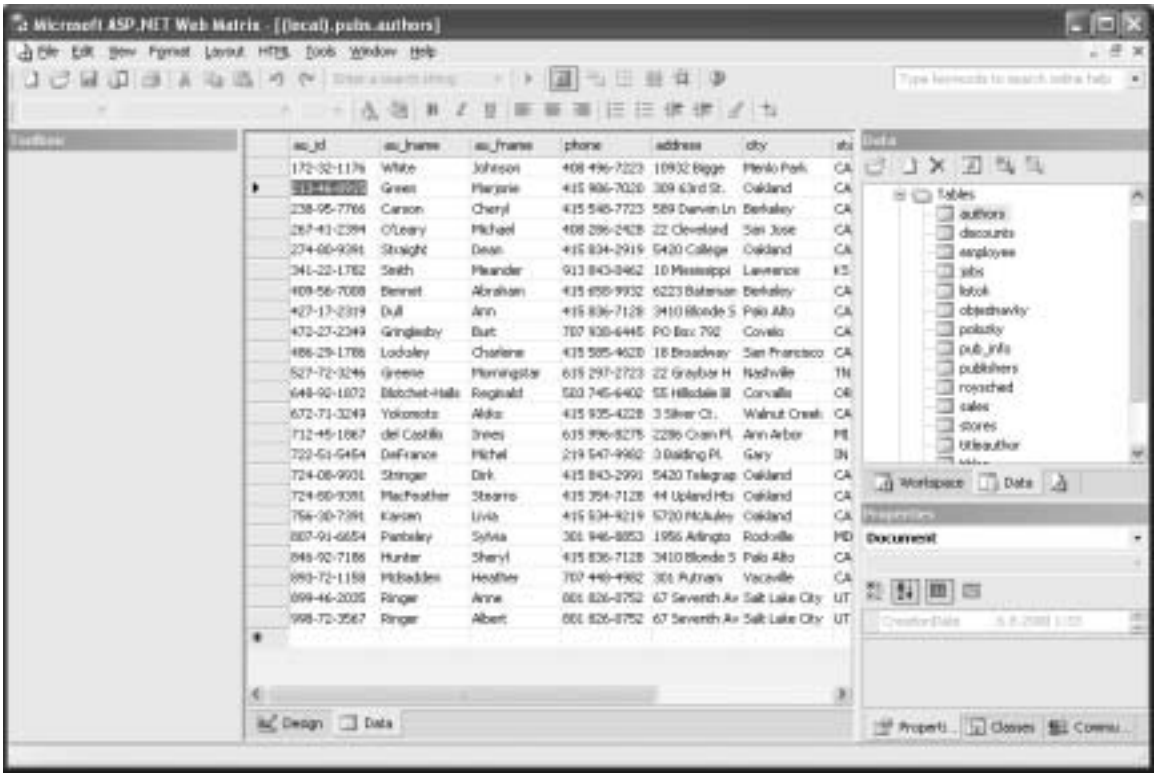
Prístup k databázam cez Web Matrix

Po pripojení sa v okne Data zobrazí zoznam tabuliek a uložených procedúr vo vybranej databáze. Všimnime si dve záložky **Design** a **Data** v dolnej časti stredného okna vývojového prostredia. V zložke Design môžeme navrhovať, prípadne meniť návrhy databázových štruktúr.



Zložka Design - návrh databázových štruktúr

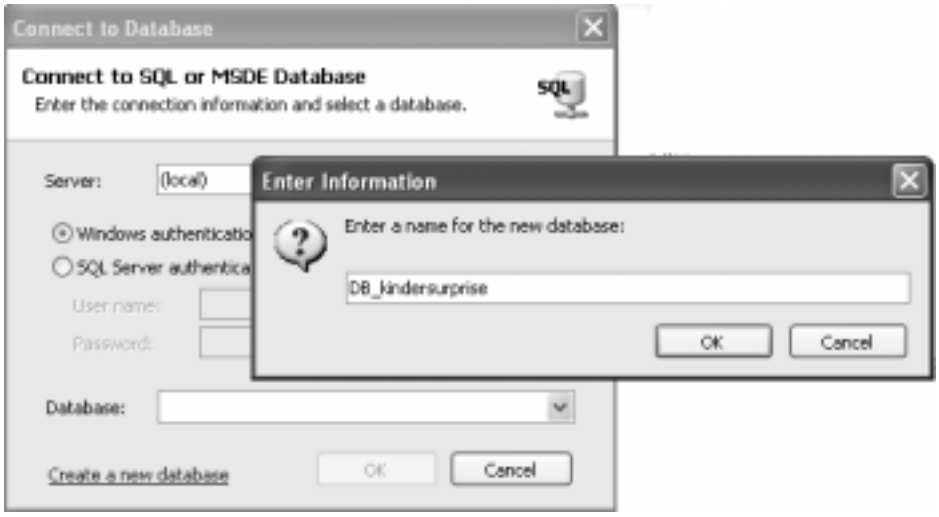
V zložke Data môžeme prehliadať, prípadne editovať obsahy jednotlivých databázových tabuliek



Zložka Data - zobrazenie a editovanie údajov v databázovej tabuľke

Vytvorenie novej databázy

V dialógu pre pripojenie sa k databáze si všimnime v dolnej časti link (nie tlačidlo) **Create a new databaze**. Pomocou tohto linku môžeme priamo v Web Matrixe vytvoriť novú databázu a navrhnúť jej štruktúru. Ako príklad jednoduchej webovej databázovej aplikácie ukážeme stránku z oblasti hobby - stránku zberateľa ručne maľovaných figúrok z vajíčok Kinder Surprise. Základom databázovej časti tejto webovej aplikácie bude pochopiteľne databáza. Vytvoríme databázu z názvom napríklad **DB_kindersurprise**.



Vytvorenie novej databázy

V okne Data sa objaví nová zložka databázy localhost.DB_kindersurprise. V jej podzložke Tables zatiaľ nie sú žiadne databázové tabuľky. Pomocou tlačidla pre vytvorenie nového objektu (druhé tlačidlo zľava, na obrázku zakrúžkované) vytvoríme novú tabuľku Figurky.

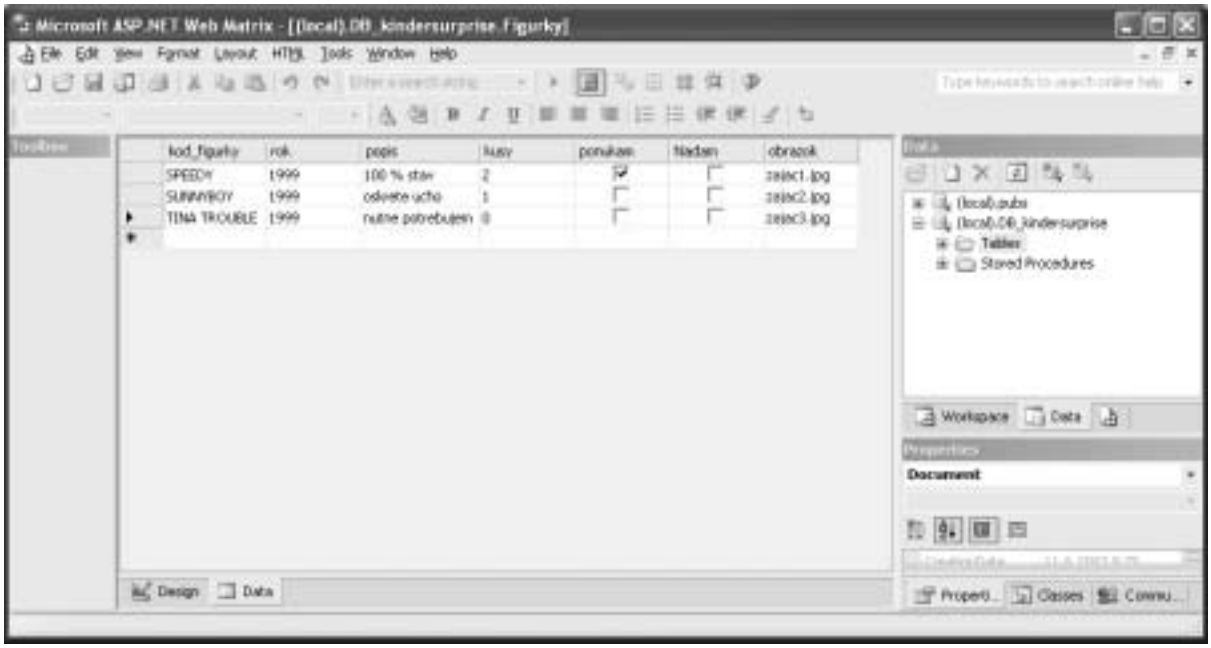
Vytváranie nových objektov v databáze

Návrh štruktúry databázovej tabuľky

Stredná časť pracovnej plochy vývojového prostredia sa prepne do režimu návrhu štruktúr databázových tabuliek. Nový stĺpec vytvoríme zakaždým pomocou tlačidla New v hornej časti obrazovky. Navrhneme jednotlivé stĺpce tabuľky z ohľadom na logiku fungovania aplikácie. Pri zberateľských stránkach bez ohľadu na to, či sa jedná o známky, odznaky, mince, figúrky a rôzne iné predmety bude mať databáza viac menej rovnakú štruktúru. Každý jednotlivý predmet bude mať v databáze svoj záznam, ktorý okrem identifikácie predmetu obsahuje jeho popis, rok kedy bol daný predmet vyrobený, potom príznaky, či zberateľ tento predmet ponúka na výmenu, prípadne ho zháňa, a prípadne informáciu o tom, koľko kusov má zberateľ vo svojej zbierke. Najlepšie sa návrh databázovej tabuľky dokumentuje vo forme príkazu CREATE TABLE pre jej vytvorenie

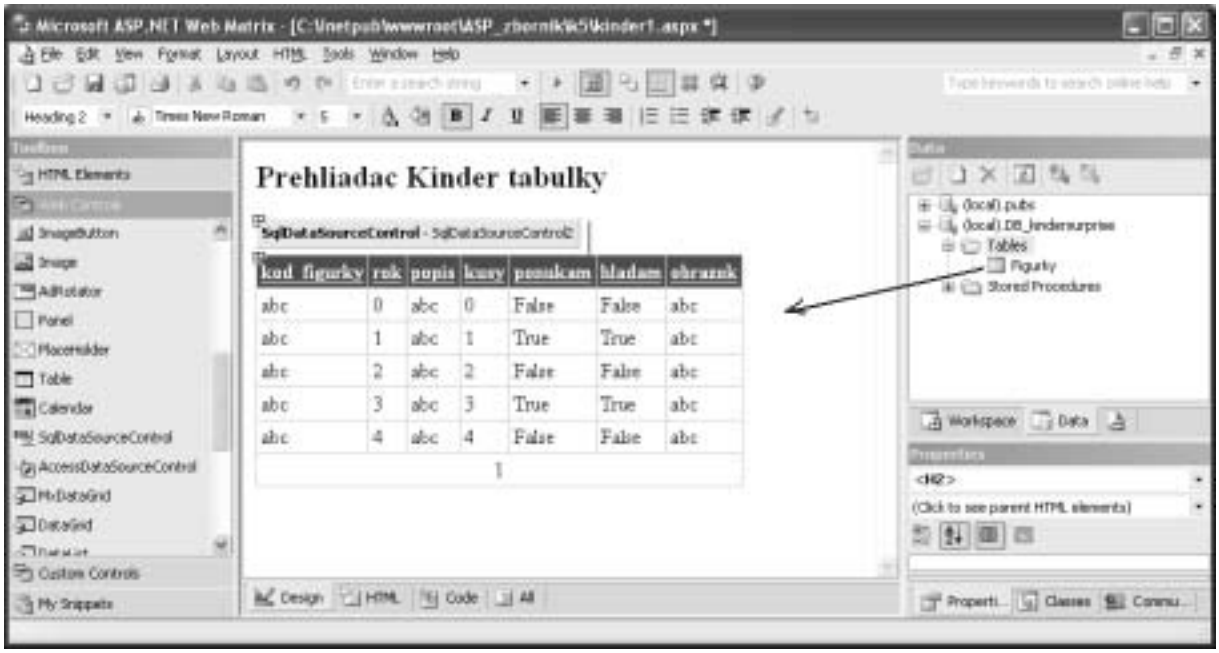
```
CREATE TABLE [Figurky] (
       [kod_figurky] [varchar] (20) COLLATE Czech_CI_AS NOT NULL ,
       [rok] [smallint] NULL ,
       [popis] [varchar] (50) COLLATE Czech_CI_AS NULL ,
       [kusy] [smallint] NULL ,
       [ponukam] [bit] NULL ,
       [hladam] [bit] NOT NULL ,
       [obrazok] [varchar] (50) COLLATE Czech_CI_AS NOT NULL ,
       CONSTRAINT [PK_Figurky_1__56] PRIMARY KEY  NONCLUSTERED
       (
              [kod_figurky]
       )  ON [PRIMARY]
) ON [PRIMARY]
GO
```

Databázová tabuľka sa totiž dá vytvoriť dvomi spôsobmi. Buď priamo napísať príkaz pre jej vytvorenie a zadať ho cez konzolu databázovému serveru, alebo použiť návrhové zobrazenie, na ktoré sme zvyknutí napríklad z kancelárskej databázy MS Access. Prvé z riešení využijú zrejme skúsenejší vývojári, začiatočníci zrejme uprednostnia návrhový dialóg. V okne Edit Table môžeme do novovytvorenej databázovej tabuľky zadať niekoľko testovacích údajov.



Vkladanie údajov do databázovej tabuľky

Teraz by sme mali pristúpiť k výkladu pojmov a teórie okolo databáz, a hlavne k predstaveniu a popisu technológie ADO.NET a datasetu. Avšak ešte predtým jedno zamyslenie. Všetky predtým vymenované pojmy sú určite lahôdkou pre vývojárov a databázových špecialistov. Ale databáza, ktorú sme navrhovali bola pre zberateľov figuriek z kinder vajíčok. Určite nás napadne, či by sa jednoduchá aplikácia, ktorá využíva databázu nedala navrhnúť aj bez veľkého teoretizovania. Ako si ukážeme neskôr určite áno a na tomto mieste ukážeme vôbec najjednoduchší spôsob návrhu prehliadača databázovej tabuľky. Pre jeho vývoj postačí jediný úkon - metódou drag and drop presunúť ikonku databázovej tabuľky z okna Data do hlavného okna vývojového prostredia .



Návrh najjednoduchšej databázovej aplikácie

Týmto jednoduchým úkonom sme na plochu aplikácie presunuli dve komponenty SqlDataSourceControl a MxDataGrid.

Pre komponentu **SqlDataSourceControl,** (nevizuálna komponenta v podobe malého šedého obdĺžnika nad databázovou tabuľkou), ktorá má na starosti pripojenie na databázu si v okne Properties môžeme všimnúť niektoré parametre, napríklad

Connection string: `server='(local)'; trusted_connection=true; database='DB_kindersurprise'`
Select command: `SELECT * FROM [Figurky]`

Zobrazenie obsahu tabuľky má na starosti komponenta **MxDataGrid**. Je napojená na komponentu SqlDataSourceControl a nastavením jej parametrov môžeme dosiahnuť zmenu zobrazenia tabuľky. Ak si pozrieme kompletný kód ktorý vývojové prostredie Web Matrix vygenerovalo po jednoduchom presune ikony databázovej tabuľky na pracovnú plochu aplikácie azda najlepšie oceníme jeho prínos.

```
<%@ Page Language="C#" %>
<%@ Register TagPrefix="wmx" Namespace="Microsoft.Matrix.Framework.Web.UI"
Assembly="Microsoft.Matrix.Framework, Version=0.6.0.0, Culture=neutral,
PublicKeyToken=6f763c9966660626" %>
<script runat="server">

    // Insert page code here
    //

</script>

<html><head></head>
<body>
    <form runat="server">
        <h2>Prehliadac Kinder tabulky </h2>

            <wmx:SqlDataSourceControl id="SqlDataSourceControl2" runat="server"
UpdateCommand=""
                  SelectCommand="SELECT * FROM [Figurky]"
                  ConnectionString="server='(local)'; trusted_connection=true;
database='DB_kindersurprise'" DeleteCommand="">
            </wmx:SqlDataSourceControl>

            <wmx:MxDataGrid id="MxDataGrid2" runat="server"
                 DataSourceControlID="SqlDataSourceControl2" BorderColor="#CCCCCC"
                 AllowSorting="True" DataMember="Figurky" AllowPaging="True" BackColor="White"
                 CellPadding="3" DataKeyField="kod_figurky" BorderWidth="1px"
BorderStyle="None">
                <PagerStyle horizontalalign="Center" forecolor="#000066" backcolor="White" ň
                   mode="NumericPages">
                </PagerStyle>
                <FooterStyle forecolor="#000066" backcolor="White"></FooterStyle>
                <SelectedItemStyle font-bold="True" forecolor="White" backcolor="#669999">
                </SelectedItemStyle>
                <ItemStyle forecolor="#000066"></ItemStyle>
                <HeaderStyle font-bold="True" forecolor="White"
backcolor="#006699"></HeaderStyle>
            </wmx:MxDataGrid>
            <!-- Insert content here -->
        </h2>
    </form>
</body>
</html>
```

Príslušného zberateľa bude asi viac než tento kód najviac zaujímať, či vôbec a ako sa takto jednoducho navrhnutá stránka zobrazí v prehliadači HTML stránok u klienta. Databázoví špecialisti samozrejme už tušia, ako by to asi mohlo dopadnúť...

Server Error in '/' Application.

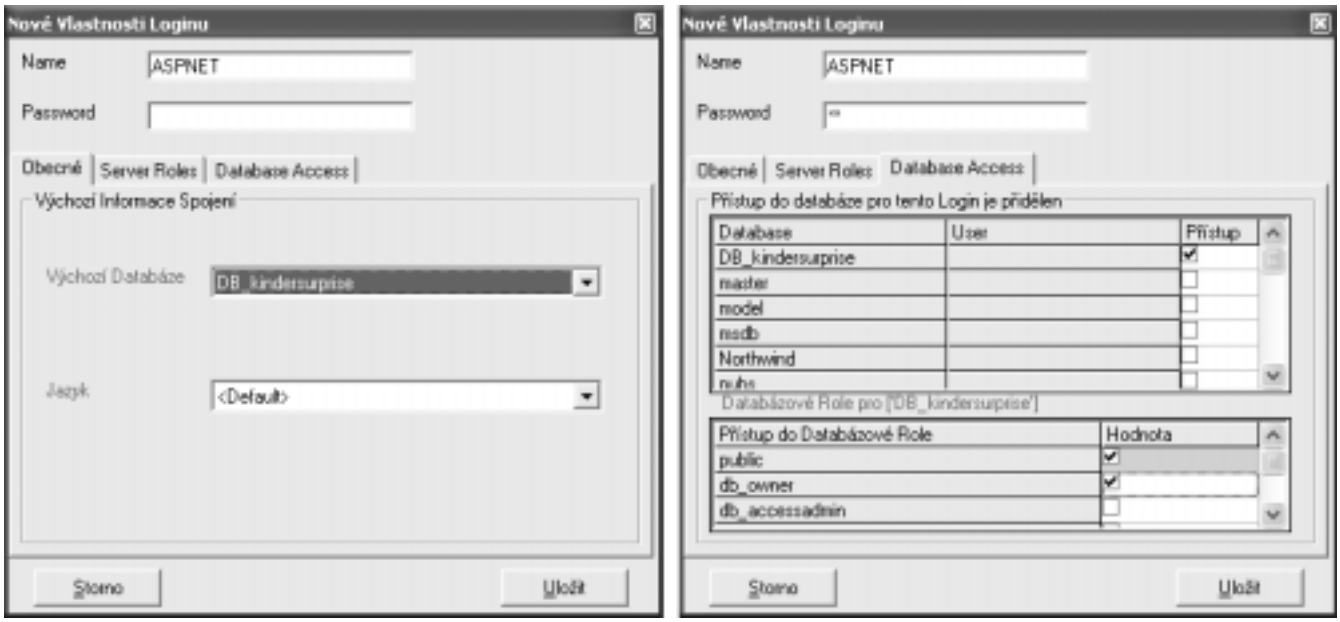
Login failed for user 'LLEVO\ASPNET'.
Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Data.SqlClient.SqlException: Login failed for user 'LLEVO\ASPNET'.

Source Error:
An unhandled exception was generated during the execution of the current web request. Information regarding the origin and location of the exception can be identified using the exception stack trace below.

Dôležitá je z toho celého len jediná veta: **Login failed for user 'LLEVO\ASPNET'**, ktorá dostatočne vysvetľuje príčinu nášho neúspechu. Nemáme samozrejme nastavené prístupové práva pre prístup k databáze pre používateľa ASPNET, takže databázový server prístup k údajom nepovolil.

Minimum databázového administrátora
Pre vytváranie nového používateľa môžeme spravidla použiť aplikáciu DbaManager, postup je na obrázku,



Vytvorenie nového používateľa v aplikácii DbaManager

no pre tohto používateľa, kde je názov domény a používateľa oddelený obráteným lomítkom sa použiť nedá, nakoľko do editačného okna pre meno sa nedá obrátené lomítko zadať. Aj pri zadávaní cez položku Database users sa vyskytne problém, tentokrát lomítko zadať dokážeme ale problém je z dĺžkou názvu.

Na „veľkom“ SQL Serveri si poradíme hravo a vytvoríme nového používateľa pomocou administrátorskej aplikácie SQL Server Enterprise manager. Ak ju nemáme, musíme nového používateľa vytvoriť pomocou príkazov jazyka SQL. Pre SQL Server a pre MSDE detto je situácia ešte zamotanejšia. Vo všeobecných učebniciach jazyka SQL nájdeme spravidla pre vytvorenie nového používateľa syntax príkazu CREATE USER. No u platformy SQL Server 2000 musíme pri administrácii prístupových práv používateľov používať iný postup. Predovšetkým SQL Server je jednoplatformový databázový server a ako taký je úzko zviazaný s operačným systémom Windows (NT/2000/XP/Windows 2003 Server...). Môžeme teda použiť dva spôsoby zabezpečenia
– Integrované zabezpečenie Windows NT (2000/XP/Windows 2003 Server)
– Zmiešané zabezpečenie Windows NT a SQL Servera

SQL Server 2000 má implementované vytváranie a manipuláciu s používateľskými účtami pomocou uložených procedúr. A aby situácia bola dostatočne zamotaná, pre prideľovanie prístupových privilégií musíme mať sami dostatočné oprávnenie, to znamená, že musíme byť členom role **sysadmin** alebo **securityadmin**.

Vytvorenie nového používateľa
Nového používateľa, alebo presnejšie povedané nový používateľský účet vytvoríme pomocou uloženej procedúry SP_ADDLOGIN. Parametre uloženej procedúry sú:

```
sp_addlogin [ @loginame = ] 'meno'
     [ , [ @passwd = ] 'heslo' ]
     [ , [ @defdb = ] 'databáza' ]
     [ , [ @deflanguage = ] 'jazyk' ]
     [ , [ @sid = ] sid ]
     [ , [ @encryptopt = ] 'šifrovacie_parametre' ]
```

Ak nezadáme parameter databáza, prihlási sa novovytvorený používateľ do databázy MASTER, čo nám málokedy vyhovuje.:

```
EXEC sp_addlogin 'ASPNET'
```

Toto je najjednoduchšia syntaktická varianta. Asi najčastejšie sa definuje nový používateľ v tvare:

```
EXEC sp_addlogin 'ASPNET ', 'DB_kindersurprise', 'pubs'
```

kedy definujeme jednak počiatočné heslo a databázu

Zmena hesla
Heslo používateľa je možné zmeniť pomocou procedúry SP_PASSWORD:

```
sp_password [ [ @old = ] 'stare_heslo' , ]
     { [ @new =] 'nove_heslo' }
     [ , [ @loginame = ] 'meno' ]
```

Ak nezadáme parameter meno, zmení sa heslo aktuálne prihláseného používateľa

Odobratie používateľa
Aj pre odobratie používateľa musíme použiť uloženú procedúru, v tomto prípade SP_DROPLOGIN

```
sp_droplogin [ @loginame = ] 'používateľské meno
```

Prístup ku konkrétnej databáze
Nový používateľský účet vytvárame samozrejme za účelom prístupu k údajom v niektorej databáze pod správou SQL Servera. :

```
sp_grantdbaccess [@loginame =] 'meno'
     [,[@name_in_db =] 'meno_v_databáze' [OUTPUT]]
```

v našom prípade môžeme používateľovi `ASPNET` umožniť prístup k aktuálnej databáze príkazom:

```
EXEC sp_grantdbaccess 'LLHOME\Administrator', ' ASPNET '
```

prípadne môžeme najskôr prepnúť na požadovanú databázu.

```
USE pubs
EXEC sp_grantdbaccess 'LLHOME\Administrator', ' ASPNET '
```

Pre prideľovanie konkrétnych práv pre prístup k objektom databázy používame príkaz GRANT. Privilégia môžeme prideľovať jednak pre vytváranie objektov, jednak pre prístup k objektom v databáze. Syntax príkazu GRANT pre udeľovanie privilégií pre vytváranie objektov

```
GRANT { ALL | ukon [ ,...n ] }
  TO nazov_uctu [ ,...n ]
```

Konkrétne môžeme prideľovať privilégia k vykonaniu nasledujúcich príkazov pre vytváranie objektov: CREATE DATABASE, CREATE DEFAULT, CREATE FUNCTION, CREATE PROCEDURE, CREATE RULE, CREATE TABLE, CREATE VIEW, BACKUP DATABASE, BACKUP LOG

Syntax príkazu GRANT pre prístup k objektom v databáze je

```
GRANT
     { ALL [ PRIVILEGES ] | oprávnenie [ ,...n ] }
     {
          [ ( stlpec [ ,...n ] ) ] ON { tabulka | pohlad }
          | ON { tabulka | pohlad } [ ( stlpec [ ,...n ] ) ]
          | ON { ulozena_procedura | rozsirena_procedure }
          | ON { uzivatelsky_definovana_funkcia }
     }
TO nazov_uctu [ ,...n ]
[ WITH GRANT OPTION ]
[ AS { group | role } ]
```

Napríklad nášmu novovytvorenému používateľovi ASPNET môžeme prideliť privilégia pre vytváranie tabuliek a pohľadov.

```
GRANT create table, create view  TO ASPNET;
```

Príkazom **REVOKE** môžeme predtým pridelené privilégia používateľovi odobrať.

```
REVOKE create table  FROM ASPNET;
```

Role

Role u platformy SQL Server 2000 môžeme rozdeliť do dvoch skupín. Na serverové role, ktoré umožňujú svojim členom administrátorské úkony a databázové role, ktoré sa vzťahujú k jednotlivým databázam.

Role je možné vytvoriť volaním systémovej uloženej procedúry SP_ADDROLE:

```
sp_addrole [ @rolename = ] 'rola'
     [ , [ @ownername = ] 'vlastnik' ]
```

napríklad

```
EXEC sp_addrole 'Analytik'
```

Aby to malo praktický význam, každej roli musíme prideliť nejaké privilégiá pomocou príkazu GRANT, napríklad

```
GRANT create table, create view  TO Analytik;
```

Po definovaní jednotlivých rolí môžeme do nich pridať používateľov pomocou volania systémovej procedúry SP_ADDROLEMEMBER

```
sp_addrolemember [ @rolename = ] 'role' ,
     [ @membername = ] 'nazov_uctu'
```

napríklad nášho používateľa `ASPNET` zaradíme medzi analytikov príkazom:

```
EXEC sp_addrolemember 'Analytik', ' ASPNET'
```

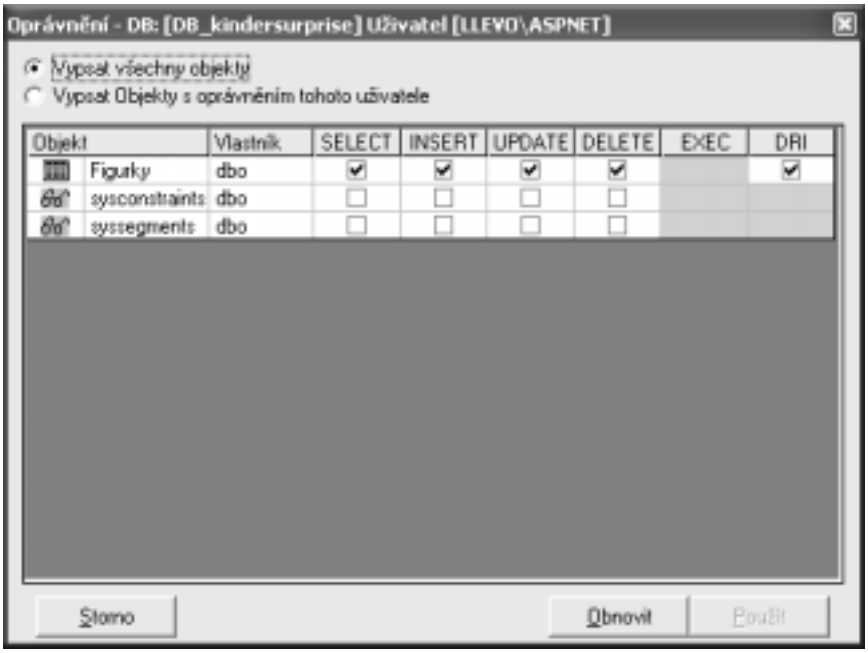
Riešením pre vytvorenie nového používateľa pre IIS z názvom v tvare doména\ASPNET napríklad 'LLEVO\ASPNET' v MSDE je utilita OSQL, ktorú spustíme z príkazovej konzoly operačného systému. Interaktívnu nápovedu k tejto utilite získame príkazom `OSQL -?`.

```
C:\>osql -?
usage: osql                [-U login id]          [-P password]
   [-S server]             [-H hostname]          [-E trusted connection]
   [-d use database name] [-l login timeout]     [-t query timeout]
   [-h headers]            [-s colseparator]      [-w columnwidth]
   [-a packetsize]         [-e echo input]        [-I Enable Quoted Identifiers]
   [-L list servers]       [-c cmdend]            [-D ODBC DSN name]
   [-q "cmdline query"]    [-Q "cmdline query" and exit]
   [-n remove numbering]   [-m errorlevel]
   [-r msgs to stderr]     [-V severitylevel]
   [-i inputfile]          [-o outputfile]
   [-p print statistics]   [-b On error batch abort]
```

Nového používateľa vytvoríme príkazom

```
osql -E -S localhost -Q "sp_grantlogin 'LLEVO\ASPNET1'"
```

Po vytvorení používateľa nastavíme napríklad pomocou aplikácie DbaManager prístupové parametre pre konkrétnu databázovú tabuľku, v našom prípade pre tabuľku Figurky. Jedná sa o operácie SELECT, INSERT, UPDATE a DELETE.



Nastavenie privilégií pre konkrétnu tabuľku

Zobrazenie databázovej tabuľky u klienta

.Vráťme sa o tri strany dozadu a pozrime si SQL príkaz pre výber údajov z databázovej tabuľky,

Select command: `SELECT * FROM [Figurky]`

ktorý je nastavený ako vlastnosť komponenty SqlDataSourceControl. Ak tento príkaz zmeníme, napríklad ak nás zaujímajú len figurky, ktoré zberateľ ponúka, teda tie, kde je parameter ponukam nastaveny na hodnotu TRUE (testujeme to hodnotu 1, FALSE ako hodnotu 0)

Select command: `SELECT * FROM [Figurky] WHERE ponukam =1`

potom sa nám zobrazia len záznamy figúriek, ktoré sú v ponuke

Iný výber údajov z databázovej tabuľky

Na tomto mieste uvedieme hlavne pre databázových začiatočníkov stručný prehľad jazyka SQL.

Stručné minimum jazyka SQL
Nasledujúci prehľad základných príkazov jazyka SQL si nekladie za cieľ výklad syntaxe, poslúži skôr na získanie prehľadu možností tohoto príkazového jazyka.

Typický SQL príkaz pre výpis všetkých údajov z databázovej tabuľky má tvar:

```
SELECT * FROM zakaznici
```

V úvode tohoto odseku sme naznačili, že to s výkladom syntaxe nebudeme preháňať.
ale v tomto prípade urobíme výnimku. Príkaz SELECT si to skutočne zaslúži.

SELECT [**DISTINCT**] *položky* **FROM** *meno_tabulky* [**WHERE** *podmienka_výberu*] [**GROUP BY** *položky,* [**HAVING** *podmienka_agregácie*]] [**ORDER BY** *položky*]

Pomocou klauzule **DISTINCT** dokážeme zariadiť, že sa nám duplicitné riadky objavia vo výpise len raz. Kľúčovým slovom **FROM** vyšpecifikujeme tabuľku, z ktorej potrebujeme vybrať údaje. Ak potrebujeme vybrať len určitú skupinu zákazníkov, alebo dokonca jednotlivca, musíme to prostredníctvom podmienky v SQL príkaze jasne špecifikovať. Na jej vyjadrenie slúži klauzula **WHERE**. V našom prípade to bude napríklad

```
SELECT * FROM zakaznici WHERE vek > 30
alebo
SELECT * FROM zakaznici VHERE rodne_cislo = '651122/1234'
```

V prvom prípade dostaneme zoznam zákazníkov nad 30 rokov, v druhom prípade podmienke vyhovuje jeden jediný zákazník zo zadaným rodným číslom. Ak chceme výpis zákazníkov nad 30 rokov vylepšiť tým že ich zoradíme od najstaršieho po najmladšieho, použijeme klauzulu **ORDER BY**.

```
SELECT * FROM zakaznici WHERE vek > 30 ORDER BY vek DESCENDING
```

Dosiaľ popisovaná skupina príkazov slúži len na výber údajov z databázy. Nemôžeme zatiaľ ani mazať, ani vkladať údaje.

Pre vkladanie údajov slúži príkaz **INSERT**

```
INSERT INTO zakaznici VALUES ('Ignac Knihomol', '591111/1234',
  'Ustredny archiv', 'Bratislava' )
```

Neaktuálne údaje môžeme prepísať novými pomocou príkazu UPDATE.

```
UPDATE zakazníci SET pracovisko = 'Urad vlady'
WHERE RodneCislo = '591111/1234'
```

Vymazať údaje môže oprávnená osoba pomocou príkazu **DELETE FROM**. Zákazníka, ktorého údaje potrebujeme z databázy vymazať, špecifikujeme podmienkou v klauzuli **WHERE** podľa jedinečného kľúča, v našom prípade rodného čísla:

```
DELETE FROM zakaznici WHERE RodneCislo = '591111/1234'
```

Okrem dotazov typu SELECT môžeme používať aj takzvané AGREGAČNÉ DOTAZY, ktoré pomocou matematických, alebo štatistických príkazov spracujú hodnoty z celých stĺpcov. V SQL sú implementované napríklad tieto funkcie:

SUM() - súčet hodnôt v stĺpci,
MIN() - minimálna hodnota ,
MAX() - maximálna hodnota,
COUNT() - počet numerických hodnôt v stĺpci,
AVG() - aritmetický priemer numerických hodnôt v stĺpci.

Okrem manipulácie s údajmi môžeme pomocou SQL príkazov pracovať s celou databázou. Dokážeme vytvoriť tabuľky, indexy, dokonca je možné definovať aj používateľov vrátane ich oprávnení.

Na vytvorenie tabuľky slúži príkaz **CREATE TABLE**:

```
CREATE TABLE meno_tabulky
(
   meno_stlpca typ [integritne_obmedzenia],
    ...,
)
```

Napríklad tabuľku vodičov vytvoríme takto:

```
CREATE TABLE vodici
(
  vodic  VARCHAR(30)  NOT NULL,
  vozidlo_farba VARCHAR(15),
  vozidlo_typ VARCHAR(20)
)
```

Veľmi podobný príkaz **ALTER TABLE** slúži na vykonanie zmien v predtým navrhnutej tabuľke:

```
ALTER TABLE meno_tabulky
(
   ADD meno_stlpca typ [integritne_obmedzenia],
   MODIFY ...
   DROP ...
)
```

Klauzula **ADD** pridá stĺpec do tabuľky, **MODIFY** zmení definíciu stĺpca a pomocou klauzule **DROP** stĺpec odoberieme. Veľmi opatrní musíme byť pri použití klauzuly pre zmenu typu MODIFY. Databázový stroj musí byť schopný vykonať automatickú typovú konverziu. Nie je možná napríklad konverzia z typu VARCHAR na INTEGER.

Dobre navrhnutá tabuľka okrem stĺpcov obsahuje veľmi často aj indexy. Na ich vytvorenie slúži príkaz **CREATE INDEX**:

```
CREATE INDEX meno_indexu ON meno_tabulky (atribut [ASC DESC] [,...])
```

Napríklad:

```
CREATE INDEX ivodica ON vodici (vodic)
```

Prvú jednoduchú databázovú aplikáciu a dve stručné minimá ohľadne administrácie SQL Servera (MSDE) a jazyka SQL máme za sebou, teraz sa budeme venovať trochu pokročilejšej teórii, pomocou ktorej vysvetlíme ako to všetko v princípe funguje.

ADO.NET

Pre vývoj databázových aplikácií máme niekoľko možností, pričom najnovšou, najvýkonnejšou a najkomfortnejšou z nich by mala byť práve ADO.NET. To isté samozrejme Microsoft tvrdil aj o techológiach ODBC, DAO, RDC, ADO a OLEDB, samozrejme vždy v dobe, kedy tieto technológie boli novinkami.

ODBC (Open Database Connectivity) výhodou tohto asi najuniverzálnejšieho rozhrania bola a stále je interoperabilita s prakticky všetkými databázovými systémami. Pre prístup k údajom sa používa jazyk SQL. Univerzálnosť je však vo väčšine prípadov vykúpená nižšou rýchlosťou.
DAO (Data Access Objects) toto rozhranie slúžilo pre prístup k údajom v databázach cez JET/ISAM, využíva technológie (ActiveX, OLE Automation)
RDO (Remote Data Objects) objekty pre vzdialený prístup k údajom.
OLE DB prístup k údajom v relačných databázach založený na technológii COM. Prístup nie je obmedzený možnosťami jazyka SQL. OLE DB prístup môže používať aj ODBC ovládače. Umožňuje nízkoúrovňový prístup, napríklad pomocou kódu v jazyku C++.
ADO (ActiveX Data Objects) využíva pre prístup k údajom v databázach ActiveX komponenty pomocou jednoduchého objektovo orientovaného rozhrania

ADO.NET

Podobne ako technológia ASP.NET je vylepšenou nástupníckou technológiou ASP stránok, aj ADO.NET je ďalším evolučným krokom technológie ADO. Je to súbor tried, štruktúr rozhraní a typov pre prístup k údajom uložených v relačných databázach ve spojení s technológiou **.NET Framework**. Azda najviac vyzdvihovanými vlastnosťami ADO.NET sú
– možnosť odpojenej práce s údajmi, vzdialene použiteľná cache
– relačný programový model
– bohatý objektový model
– maximálny výkon
– práca s heterogénnymi údajmi
– nahráva a ukladá údaje ako XML
– nahráva a ukladá relačné schémy ako XSD

V predchádzajúcich rozhraniach sa prístup k údajom v databáze realizoval pomocou objektu zvaného Recordset. V ADO.NET tento objekt nenájdeme, pre prístup k údajom sa používajú dva druhy objektov

DataReader sa používa pre jednoduché čítanie údajov (read-forward-only)

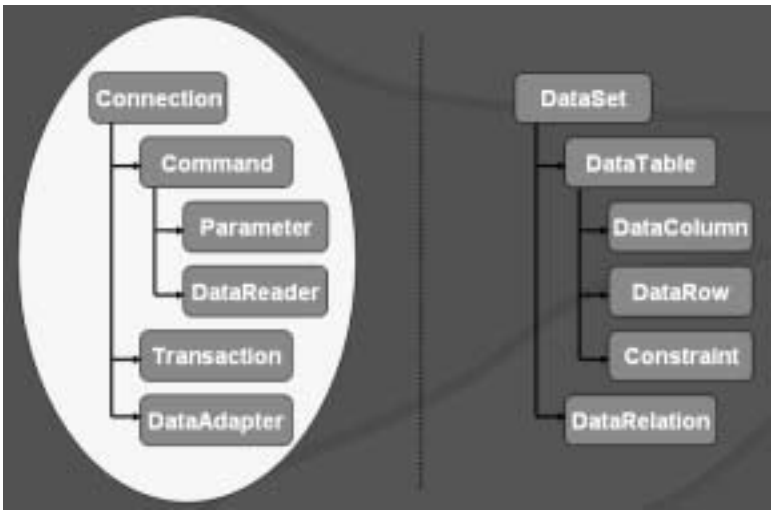
DataSet sa používa pre čítanie a modifikáciu údajov a to aj v prípade ak nie sme momentálne pripojení k databáze, napríklad ak sme mimo dosahu Wireless LAN a podobne.

ADO.NET - sprostredkovatelia verzus konzumenti
Triedy objektového modelu ADO.NET môžeme rozdeliť na dve skupiny:
– **sprostredkovatelia údajov** (Data Providers alebo Managed Providres)
– **konzumenti údajov,** napríklad formuláre windows, formuláre webových aplikácií a podobne.

A medzi nimi je situovaný objekt DataSet. Zjednodušene môžeme vyjadriť tento vzťah nasledovne

Sprostredkovatelia údajov — DataSet — Konzumenti údajov

Pozrime sa najprv na ľavú stranu tohoto vzťahu:



Sprostredkovatelia údajov

pričom začneme súčasťami sprostredkovateľov údajov

Connection (objekt pripojenia) slúži na fyzické pripojenie k zdroju údajov. Pripojenie môžeme otvoriť, zatvoriť, realizovať transakcie, prípadne zmeniť databázu. Uvedieme príklad pre vytvorenie, otvorenie a zatvorenie pripojenia:

```
Dim strConn As String
strConn = "Provider=SQLOLEDB;Data Source=(local)" & _
                  "Initial Catalog=Northwind;Trusted_Connection=Yes;"
Dim cn As New OleDbConnection(strConn)
cn.Open()
...
cn.Close()
```

Command (objekt príkazu) slúži pre prácu s príkazmi databázového jazyka SQL napríklad:

```
Dim strConn, strSQL As String
strConn = "Provider=SQLOLEDB;Data Source=(local);..."
Dim cn As New OleDbConnection(strConn)
cn.Open()
strSQL = "SELECT OrderID, OrderDate FROM Orders " & _
                 "WHERE CustomerID = ?"
Dim cmd As New OleDbCommand(strSQL, cn)
cmd.Parameters.Add("@CustomerID", OleDbType.WChar, 5)
cmd.Parameters("@CustomerID").Value = "ALFKI"
Dim rdr As OleDbDataReader = cmd.ExecuteReader()
...
rdr.Close()
cn.Close()
```

Príkazy jazyka SQL môžeme rozdeliť na dve skupiny, na výkonné (INSERT, DELETE, UPDATE..), ktoré vykonajú nejakú akciu, napríklad pridajú, alebo vymažú záznam v databáze ale nevracajú žiadnu hodnotu a dopytovacie (SELECT), ktoré vracajú jeden alebo viac záznamov. Pre výkonné príkazy slúži metóda **ExecuteNonQuery()** napríklad:

```
Dim cn As New OleDbConnection(strConn)
cn.Open()
strSQL = "DELETE FROM Products WHERE ProductID = ?"
Dim cmd As New OleDbCommand(strSQL, cn)
cmd.Parameters.Add("@ProductID", OleDbType.Integer)
cmd.Parameters("@ProductID").Value = 7
Dim intRowsAffected As Integer = cmd.ExecuteNonQuery()
```

Pre dopytovacie príkazy, ktoré vrátia len jeden záznam, teda skalár slúži metóda **ExecuteScalar()**

```
Dim cn As New OleDbConnection(strConn)
cn.Open()
strSQL = "SELECT COUNT(OrderID) FROM Orders WHERE CustomerID = ?"
Dim cmd As New OleDbCommand(strSQL, cn)
cmd.Parameters.Add("@CustomerID", OleDbType.WChar, 5)
cmd.Parameters("@CustomerID").Value = "ALFKI"
Dim intNumOrders As Integer = cmd.ExecuteScalar()
```

DataReader (objekt pre čítanie údajov) slúži pre prácu s údajmi, no len pre ich dopredné čítanie. Toto obmedzenie je však vyvážené vysokou rýchlosťou prístupu k údajom. Ukážeme si príklad čítania údajov pomocou objektu DataReader:

```
Dim strConn, strSQL As String
Dim cn As New OleDbConnection(strConn)
Dim cmd As New OleDbCommand(strSQL, cn)
cn.Open()
Dim rdr As OleDbDataReader = cmd.ExecuteReader()
Do While rdr.Read()
    Console.WriteLine(rdr("OrderID"))
    Console.WriteLine(rdr(1))
    Console.WriteLine(rdr.GetDateTime(2))
Loop
rdr.Close()
cn.Close()
```

Transaction (objekt pre riadenie transakcií) slúži pre prácu s údajmi pomocou transakcií. Databázové servery nám prostredníctvom transakcií umožnia zachovávať v každom okamihu konzistenciu údajov v databázových tabuľkách a nedeliteľnosť vykonávaných zmien na princípe „Buď všetko, alebo nič“. Klasická transakcia je ak klient A prevádza finančnú čiastku na účet klienta B, musí sa spomínaná čiastka najskôr odpočítať z účtu klienta A a následne pripočítať na účet klienta B. Je to triviálna operácia, ale len v tom prípade ak všetko prebehne bez problémov. Predstavme si však situáciu, že klient B svoj účet medzičasom zrušil, alebo klient A sa pomýlil pri zadávaní čísla účtu klienta B. V takomto prípade sa predmetná finančná čiastka odpočíta z účtu klienta A, ale pre spomínané príčiny nie je možné uložiť tieto peniaze na účet klienta B. Našťastie sa nič nestane, pretože systém riadenia databázy transakciu stornuje a peniaze sa bezpečne vrátia na účet klienta A. Takáto transakcia buď prebehne celá, to znamená, že peniaze sa z účtu odosielateľa odpočítajú a pripočítajú sa na účet príjemcu, alebo sa transakcia zruší a peniaze nebudú z účtu odosielateľa odpočítané. Priebeh transakcie riadime v jazyku SQL príkazmi **COMMIT**, **SAVEPOINT** a **ROLLBACK**. Pomocou objektu Transaction vykonáme jednoduchú transakciu pomocou kódu:

```
Dim cn As New OleDbConnection(strConn)
cn.Open()
Dim txn As OleDbTransaction = cn.BeginTransaction()
Dim cmd As New OleDbCommand(strSQL, cn, txn)
```

```
intRowsAffected = cmd.ExecuteNonQuery()
If intRowsAffected = 1 Then             'Uspesne ukoncenie transakcie!
    txn.Commit()
Else                                     'Chyba!
    txn.Rollback()
End If
cn.Close()
```

DataAdapter vytvára adaptér (prepojenie) medzi objektom pripojenia a sadou údajov. Pre manipuláciu s údajmi je možné používať príkazy **SelectCommand, InsertCommand, DeleteCommand** a **UpdateCommand,** ktoré významovo zodpovedajú príkazom jazyka SQL SELECT, INSERT, DELETE a UPDATE. Príklad kódu:

```
Dim catDA As SqlDataAdapter = New SqlDataAdapter("SELECT CategoryID, CategoryName FROM
Categories", nwindConn)
catDA.UpdateCommand = New SqlCommand("UPDATE Categories SET CategoryName = @CategoryName WHERE
CategoryID = @CategoryID", nwindConn)

catDA.UpdateCommand.Parameters.Add("@CategoryName", SqlDbType.NVarChar, 15, "CategoryName")

Dim workParm As SqlParameter = catDA.UpdateCommand.Parameters.Add("@CategoryID", SqlDbType.Int)
workParm.SourceColumn = "CategoryID"
workParm.SourceVersion = DataRowVersion.Original

Dim catDS As DataSet = New DataSet
catDA.Fill(catDS, "Categories")

Dim cRow As DataRow = catDS.Tables("Categories").Rows(0)
cRow("CategoryName") = "New Category"

catDA.Update(catDS)
```

DataSet

Je základná najkomplexnejšia trieda v rozhraní ADO.Net. Z hľadiska uloženia údajov a manipulácie s nimi je to v podstate relačná databáza v pamäti. Údajmi sa plní buď z databázy pomocou objektu DataAdapter, prípadne pracuje s údajmi vo formáte XML. Dôležité je pochopiť, že DataSet je ako taký úplne odpojený od dátového zdroja, alebo inými slovami povedané, on sám sa nikdy nepripojí priamo do databázy. DataSet vlastne ani nevie z akého druhu dátového zdroja pochádzajú, či z XML, Oracle, alebo z SQL Servera. Takáto univerzálnosť je vynikajúca z hľadiska škálovateľnosti a prípadnej migrácie aplikácie.

DataSet môže laicky povedané zapúzdrovať niekoľko nehomogénnych databázových tabuliek, presnejšie DataSet pozostáva z viacerých objektov typu DataTable. Tieto objekty majú podobné vlastnosti ako tabuľky v relačnom databázovom serveri. (riadky, stĺpce, integritné obmedzenia, relačné vzťahy medzi tabuľkami cez cudzie kľúče..).

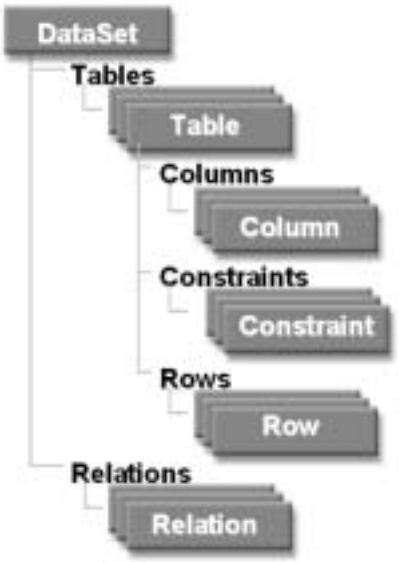


Schéma objektu DataSet

Komplexne môžeme naznačiť prácu s objektom DataSet pomocou nasledujúceho fragmentu kódu:

```
string sConnString = "Persist Security Info=False;" +
                     "User ID=sa;Initial Catalog=Northwind;" +
                     "Data Source=MYSERVER";
SqlConnection conn = new SqlConnection(sConnString);
conn.Open();
string sQueryString = "SELECT CompanyName FROM Customers";
SqlDataAdapter myDSAdapter = new SqlDataAdapter();
DataSet myDataSet = new DataSet();
myDSAdapter.SelectCommand = new SqlCommand(sQueryString, conn);
myDSAdapter.Fill(myDataSet);
conn.Close();
```

DataTable - aj tento objekt môžeme rozmeniť „na drobné“, v našom prípade na objekty Columns, Rows a Constraints. Ukážeme si najskôr kód pre uloženie výsledkov dopytu cez DataAdapter

```
Dim strConn, strSQL As String
Dim da As New OleDbDataAdapter(strSQL, strConn)
Dim tbl As New DataTable()
da.Fill(tbl)
```

a taktiež kód pre naplnenie údajov

```
Dim strConn, strSQL As String
Dim da As New OleDbDataAdapter(strSQL, strConn)
Dim tbl As New DataTable()
tbl.Columns.Add("CustomerID", GetType(String))
tbl.Columns.Add("CompanyName", GetType(String))
…
da.Fill(tbl)
```

Ak chceme vypísať jednotlivé záznamy použijeme kód typu:

```
Dim row As DataRow
For Each row In tbl.Rows
    Console.WriteLine(row("CustomerID") + " - " + row("CompanyName"))
Next row
```

Každý objekt **DataColumn** ma dve povinné vlastnosti názov a dátový typ , a pripadne ďalšie vlastnosti napríklad či povoľuje null , obmedzenia rozsahu údajov u reťazcov a podobne. Pre ilustráciu ukážeme kód pre pridanie nového stĺpca do tabuľky

```
DataColumn c =
new DataColumn("cislo",typeof(int));
Tabulka.Columns.Add(c);
```

DataRelation umožňuje definovať relačné vzťahy medzi objektami v databáze. Relácie medzi tabuľkami popisujú vzťahy medzi objektmi reálneho sveta, ktoré tieto tabuľky predstavujú. Pri návrhu databázových tabuliek, na ktoré naväzuje aplikačná logika môžeme definovať niekoľko druhov vzťahov. Vzhľadom na kardinalitu budeme entity nazývať prvá a druhá. Medzi nimi bude vzťah (relácia)

- 1 : 1 - prvej entite, napríklad záznamu v databázovej tabuľke zodpovedá maximálne jedna druhá entita, teda záznam z inej databázovej tabuľky
- 1 : N - prvej entite zodpovedá viac druhých entít. Ale naopak druhej entite zodpovedá maximálne jedna prvá entita.
- M : N - prvej entite zodpovedá viac druhých entít. A taktiež aj naopak, druhej entite zodpovedá viac prvých entít.

Ale to sme urobili len krátky odskok do teórie návrhu databáz. Ukážeme si radšej praktický príklad pre naplnenie objektu DataSet údajmi, definovanie relácie medzi údajmi a prístup k takto relačne zviazaným údajom.

```
Dim daCustomers As New SqlDataAdapter(strSQLCustomers, strConnSQLServer)
Dim daOrders As New OracleDataAdapter(strSQLOrders, strConnOracle)
Dim ds As New DataSet()
daCustomers.Fill(ds, "Customers")
daOrders.Fill(ds, "Orders")

ds.Relations.Add("CustomersOrders", ds.Tables("Customers").Columns("CustomerID"), _
                                    ds.Tables("Orders").Columns("CustomerID"))

Dim rowCustomer, rowOrder As DataRow
For Each rowCustomer In ds.Tables("Customers").Rows
    Console.WriteLine(rowCustomer("CompanyName"))
    For Each rowOrder In rowCustomer.GetChildRows("CustomersOrders")
        Console.WriteLine(vbTab & rowOrder("OrderID") & " - " & rowOrder("OrderDate"))
    Next rowOrder
Next rowCustomer
```

Constraint (integritné obmedzenia). Aby sme zabránili zadávaniu nesprávnych hodnôt do databázových tabuliek, je niekedy potrebné zaviesť pre niektoré stĺpce určité obmedzenia. Bolo by nezmyslom stanoviť niektorý stĺpec ako primárny kľúč, napríklad rodné číslo, keby polovica záznamov mala v tomto stĺpci hodnotu NULL a podobne. Obmedzenia sa môžu vzťahovať ku konkrétnemu stĺpcu, alebo k celej tabuľke. Najčastejšie sa používajú obmedzenia NOT NULL, PRIMARY KEY, UNIQUE, FOREIGN KEY, a CHECK

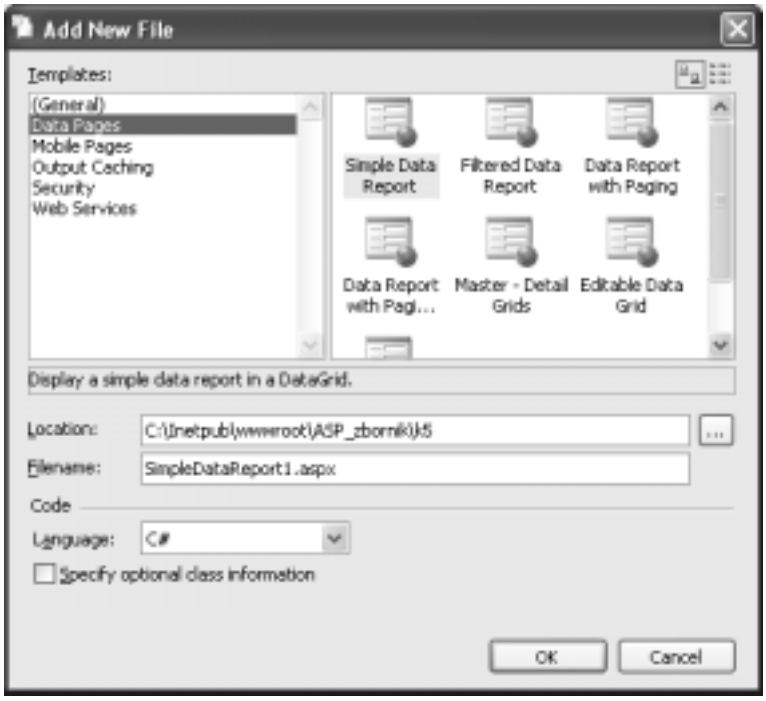
Vývoj databázových aplikácií v prostredí Web Matrix

Prvú databázovú aplikáciu vo Web Matrixe, vytvorenú presunom ikony databázovej tabuľky na plochu aplikačného formulára máme za sebou, no Web Matrix poskytuje pre vývoj databázových aplikácií oveľa širšie možnosti.
Pre založenie nového projektu využijeme záložku **Data Pages** v dialógu pre pridanie nového projektu alebo súboru. Môžeme si vybrať z niekoľkých druhov návrhu budúcej aplikácie, napríklad jednoduchý databázový report, filtrovaný databázový report, databázový report so stránkovaním, alebo databázový report typu Master -Detail editovateľný DataGrid a podobne.

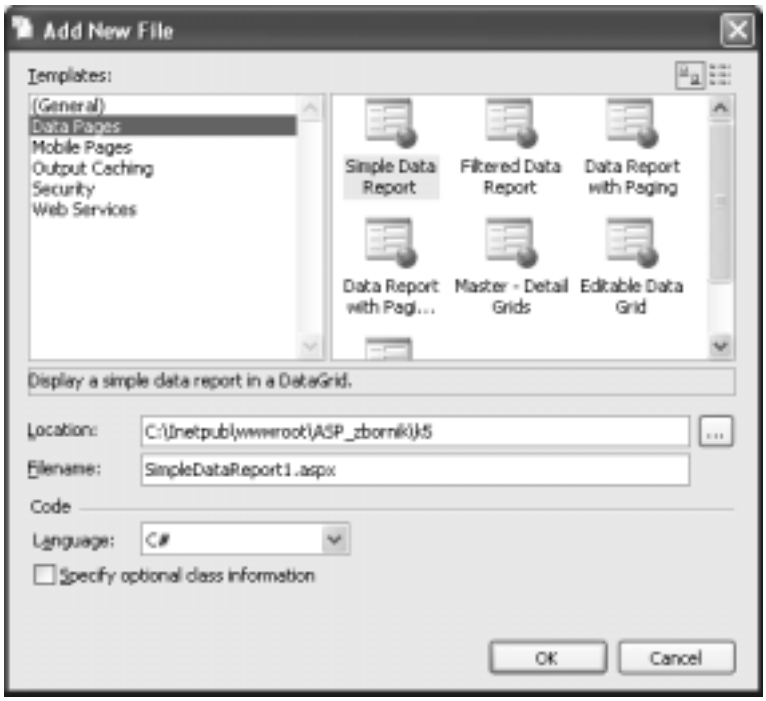
ponuka druhov databázových aplikácii

Jednoduchý výpis údajov

Jednotlivé typy databázových súborov vytvárame v dialógu pre vytvorenie nového súboru v zložke **Data Pages**. Ako prvú vytvoríme aplikáciu typu Simple Data Report. Súbor s aplikáciu nazveme pre názornosť SimpleDataReport.aspx.



Dialóg pre vytváranie databázových aplikácií



Aplikácia typu Simple Data Report

Je to veľmi podobná v princípe vlastne rovnaká aplikácia ako náš prvý pokus. Otázka znie, ako napojiť objekt DataGrid na zdroj údajov.

V záložke Code si môžeme pre zaujímavosť pozrieť príklad kódu pre pripojenie sa k zdroju údajov, ktorý by sme mali upraviť pre naše potreby

```
void Page_Load(object sender, EventArgs e) {

    // TODO: Update the ConnectionString and CommandText values for your application
    string ConnectionString = "server=(local);database=pubs;trusted_connection=true";
    string CommandText = "select au_lname as [Last Name], au_fname as [First Name], Address,
City, State from Authors";

    SqlConnection myConnection = new SqlConnection(ConnectionString);
    SqlCommand myCommand = new SqlCommand(CommandText, myConnection);

    myConnection.Open();

    DataGrid1.DataSource = myCommand.ExecuteReader(CommandBehavior.CloseConnection);
    DataGrid1.DataBind();
}
```

Existuje aj oveľa jednoduchšie riešenie. Použijeme malý trik, záložku Code si vôbec nevšímame a vytvoríme prepojenie presunom ikony databázovej tabuľky Figurky z okna Data do okna aplikácie. Spolu s objektom SqlDataSourceControl, kôli ktorému tento úkon robíme sa prenesie aj objekt MxDataGrid, ktorý nepotrebujeme, takže ho vymažeme. Následne nastavíme u objektu DataGrid vlastnosť Data Source na náš objekt SqlDataSourceControl. Tým je jadro aplikácie hotové.

Samozrejme môžeme to urobiť aj klasicky „podľa návodu“. V záložke **Code** modifikujeme hlavne pripojovací reťazec, pole výberu a SQL dopyt pre výber stĺpcov.

```
data source=LLEVO; initial catalog=DB_kindersurprise;
integrated security=SSPI; persist security info=False;
workstation id=LLEVO; packet size=4096
```

aj SQL dopyt pre výber údajov

```
SELECT * FROM figurky
```

Filtrovaný výpis údajov

Nie vždy potrebujeme vypísať všetky údaje v databázovej tabuľke. Potrebujeme výber nejakým spôsobom obmedziť. Pre tento príklad sa naša jednoduchá tabuľka z troma záznamami absolútne nehodí. Aj samotná šablóna aplikácie ktorú vygeneruje vývojové prostredie je navrhnutá pre cvičnú databázu PUBS, ktorá sa nainštaluje spolu s databázovým serverom (SQL Serverom aj MSDE).
Cvičná databáza PUBS je určená jednak na naše prvé pokusy s jazykom SQL, nájdeme tu aj niekoľko cvičných tabuliek, ktoré spolu tvoria databázu autorov, titulov, vydavateľstiev a ich zamestnancov a predajcov. Nás bude konkrétne zaujímať tabuľka **authors** , ktorá je navrhnutá nasledovne:

au_id	**id**
au_lname	**varchar**(40)
au_fname	**varchar** (20)
phone	**char** (12)
address	**varchar** (40)
city	**varchar** (20)
state	**char** (2)
zip	**char** (5)
contract	**bit**

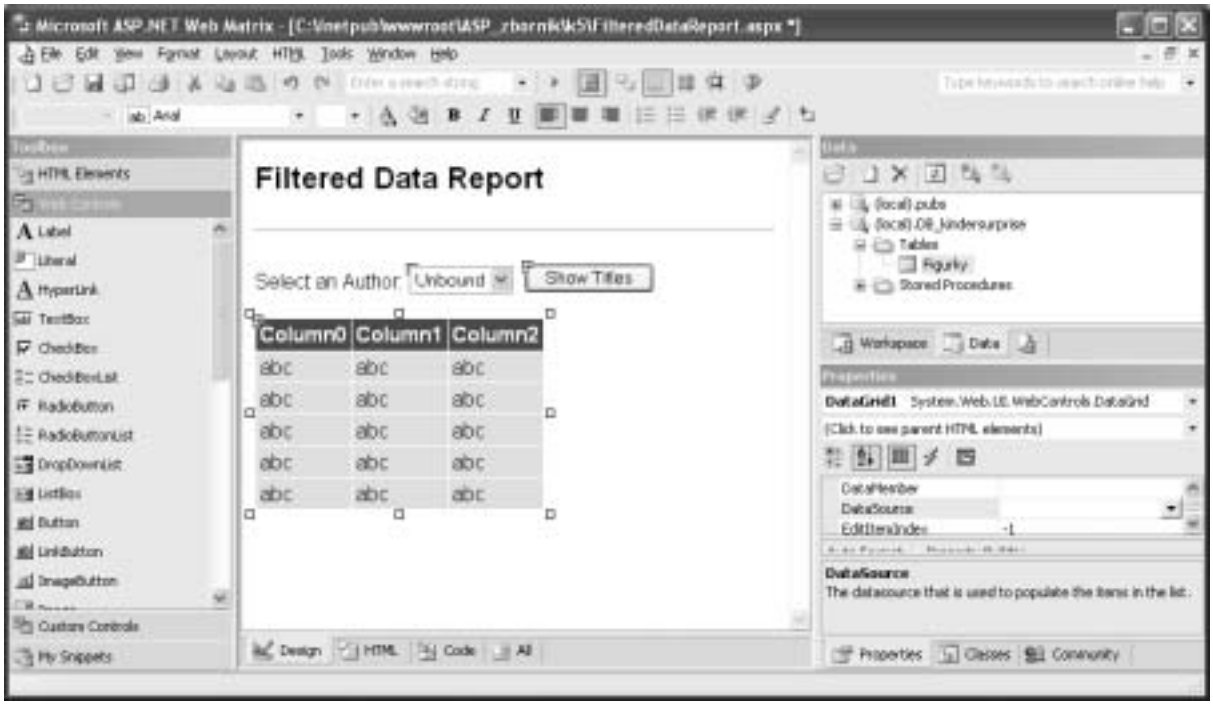
Pri návrhu tabuľky authors boli do príkazu pre jej vytvorenie zakomponované aj niektoré integriné obmedzenia. Tabuľka bola vytvorená príkazom:

```
CREATE TABLE authors
(
       au_id id NOT NULL ,
       au_lname varchar (40)  NOT NULL ,
       au_fname varchar (20)  NOT NULL ,
       phone char (12)  NOT NULL,
       address varchar (40),
       city varchar (20),
       state char (2),
       zip char (5),
       contract bit NOT NULL,
         CONSTRAINT UPKCL_auidind PRIMARY KEY  CLUSTERED
       (
              au_id
       )  ON PRIMARY ,
        CHECK (au_id like '[0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9][0-9][0-9]'),
        CHECK (zip like '[0-9][0-9][0-9][0-9][0-9]')
) ON PRIMARY
```

a obsahuje 23 záznamov

au_id	au_lname	au_fname	phone	address	city	state	zip	contract
172-32-1176	White	Johnson	408 496-7223	10932 Bigge Rd.	Menlo Park	CA	94025	1
213-46-8915	Green	Marjorie	415 986-7020	309 63rd St. #411	Oakland	CA	94618	1
238-95-7766	Carson	Cheryl	415 548-7723	589 Darwin Ln.	Berkeley	CA	94705	1
...								

Databázu s ktorou budeme v príklade pracovať sme stručne predstavili, vráťme sa k aplikácii typu Simple Data Report. Súbor s aplikáciu nazveme pre názornosť SimpleDataReport.aspx.



Aplikácia typu Filtered Data Report

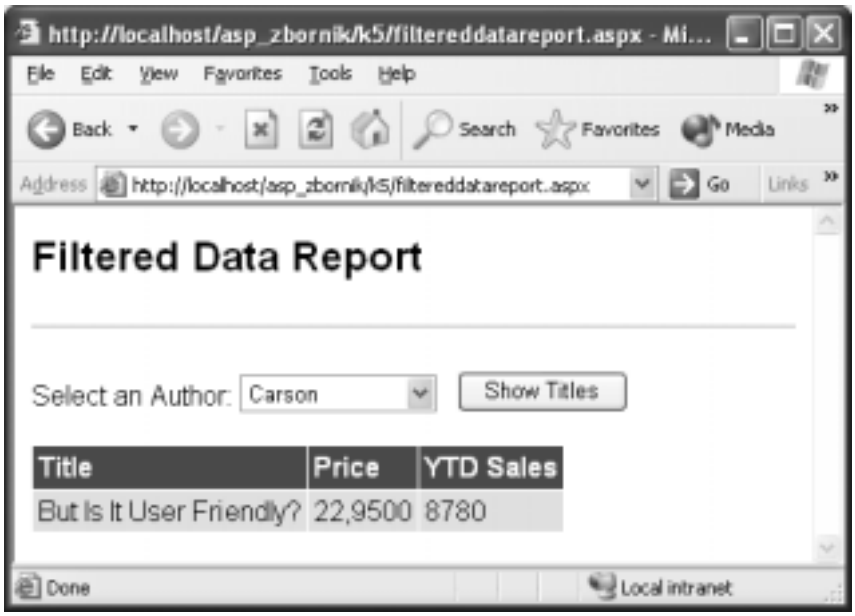
Keďže vzorový návrh formulára je určený pre rovnakú databázu s akou budeme pracovať aj my, ponecháme formulár aj jeho vlastnosti v okne properties bezo zmien. Aj kód v prípade spustenia na lokálnom počítači môže zostať bezo zmien. Ak používame iný server v kóde zmeníme len jediné slovo **(local)**, za názov databázového servera v pripojovacom reťazci a to na dvoch miestach, v procedúre `Page_Load` a v procedúre a v procedúre `ApplyFilter_Click`. Zmena je vo výpise kódu vyznačená hrubým fontom. Pre úsporu miesta uvádzame len kód v programovacom jazyku Visual Basic. Programátori v C# si to určite ľahko nájdu.

```
Sub Page_Load(Sender As Object, E As EventArgs)

    If Not Page.IsPostBack Then

        ' Databind the filter dropdown on the first request only
        ' (viewstate will restore these values on subsequent postbacks).

        ' TODO: update the ConnectionString and CommandText values for your application
        Dim ConnectionString As String = "server=(local);database=pubs;trusted_connection=true"
        Dim CommandText As String = "select distinct au_lname from Authors"

        Dim myConnection As New SqlConnection(ConnectionString)
        Dim myCommand As New SqlCommand(CommandText, myConnection)

        ' TODO: Update the DataTextField value
        DropDownList1.DataTextField = "au_lname"
        myConnection.Open()

        DropDownList1.DataSource = myCommand.ExecuteReader(CommandBehavior.CloseConnection)
        DropDownList1.DataBind()

        ' insert an "All" item at the beginning of the list
        DropDownList1.Items.Insert(0, "-- All Authors --")

    End If

End Sub


Sub ApplyFilter_Click(Sender As Object, E As EventArgs)

    ' TODO: update the ConnectionString value for your application
    Dim ConnectionString As String = "server=(local);database=pubs;trusted_connection=true"
    Dim CommandText As String

    ' get the filter value from the DropDownList
    Dim filterValue As String = DropDownList1.SelectedItem.Text.Replace("'", "''")

    ' TODO: update the CommandText value for your application
    If filterValue = "-- All Authors --" Then
        CommandText = "select distinct title as Title, price as Price, ytd_sales as [YTD Sales]
from titleview"
    Else
        CommandText = "select title as Title, price as Price, ytd_sales as [YTD Sales] from
titleview where au_lname = '" & filterValue & "'"
    End If

    Dim myConnection As New SqlConnection(ConnectionString)
    Dim myCommand As New SqlCommand(CommandText, myConnection)

    myConnection.Open()
```

```
    DataGrid1.DataSource = myCommand.ExecuteReader(CommandBehavior.CloseConnection)
    DataGrid1.DataBind()

End Sub
```

Po uložení aplikácie môžeme overiť jej funkčnosť.

Aplikácia typu Filtered Data Report - výber autora



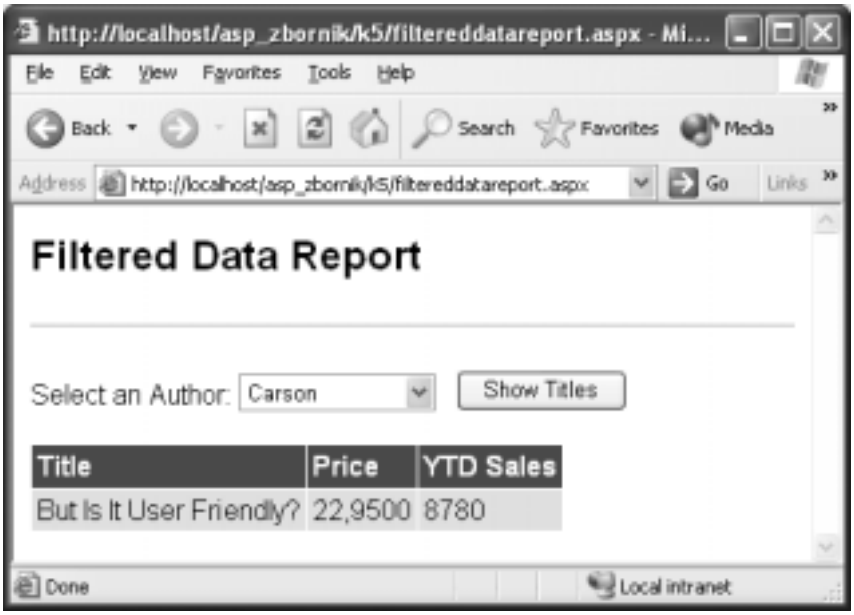
Aplikácia typu Filtered Data Report - výpis jeho titulov

Samozrejme časť formulára pomocou ktorej zostavujeme podmienku pre výber požadovaných údajov v konkrétnej aplikácii prekonštruujeme podľa požiadaviek. Spravidla pre definovanie kombinovanej podmienky budeme potrebovať viac ovládacích prvkov

Stránkovaný výpis údajov

Jednou z ciest pre zobrazovanie údajov z veľkej tabuľky je pomocou nejakého pravidla - podmienky obmedziť počet údajov a tieto následne zobraziť na HTML stránke klienta. Ani tento postup však niekedy nevedie k cieľu. Obvykle sa

dá vybrať požadovaný rozumný počet údajov, ale uvedomme si jednu vec, formulár pre vytvorenie podmienky dávame vlastne k dispozícii klientovi a ten často v dobrom úmysle vedený heslom "čím viac tým lepšie" vyberie čo najviac údajov. Po polhodinovom naťahovaní stránky v ktorej sa pomocou dvoj milimetrovej ikonky pre skrolovanie aj tak nedá orientovať pochopí svoj omyl. Niekedy však skutočne potrebujeme obsiahnuť zobraziť a napríklad postupne skontrolovať viac údajov. Vtedy je výhodné použiť stránkovaný výpis.
Návrhový formulár vygenerovaný vývojovým prostredím nie je príliš zaujímavý, všimnime si len v pravom dolnom rohu DataGridu čísla stránok (zakrúžkovali sme ich v obrázku výslednej stránky u klienta).



Aplikácia typu Data Report with Paging

V kóde je opäť potrebné pri prevádzkovaní na inom serveri zmeniť len názov databázového servera v pripojovacom reťazci, tentokrát len na jednom mieste, v procedúre `BindGrid`.

```
Sub Page_Load(Sender As Object, E As EventArgs)

    If Not Page.IsPostBack Then
        ' Databind the data grid on the first request only
        ' (on postback, rebind only in paging command)
        BindGrid()
    End If
End Sub

Sub DataGrid_Page(Sender As Object, e As DataGridPageChangedEventArgs)
    DataGrid1.CurrentPageIndex = e.NewPageIndex
    BindGrid()
End Sub

Sub BindGrid()
    ' TODO: update the ConnectionString and CommandText values for your application
 Dim ConnectionString As String = "server= (local);database=pubs;trusted_connection=true"
    Dim CommandText As String = "select au_lname, au_fname, address, city, state from Authors
order by au_lname"

    Dim myConnection As New SqlConnection(ConnectionString)
    Dim myCommand As New SqlDataAdapter(CommandText, myConnection)
    Dim ds As New DataSet()
    myCommand.Fill(ds)
    DataGrid1.DataSource = ds
    DataGrid1.DataBind()
End Sub
```

Výpis údajov typu Master Detail

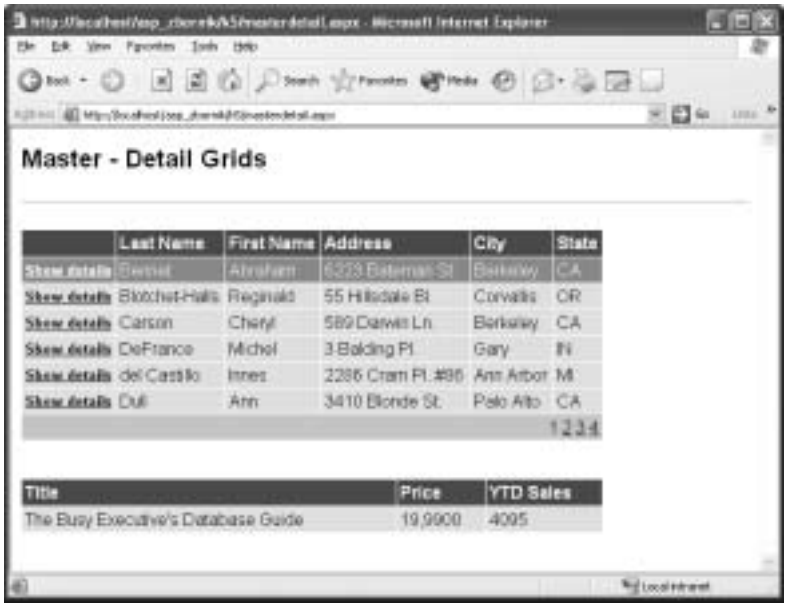
S dokumentami typu master - detail sa bežne stretávame v papierovej informatike. Príkladom takéhoto vzťahu môžu byť napríklad skladové karty, alebo výpožičné lístky kníh, kedy v záhlaví tlačiva máme informácie o položke, alebo o knihe a pod týmito informáciami je tabuľka pohybov na skladovej karte alebo výpožičiek. V našom príklade k tabuľke autorov priradíme aj tabuľku kníh, ktoré títo autori napísali. S príkladom opäť na lokálnom PC nemusíme robiť nič, inak upravíme reťazec pre pripojenie sa k databázovému serveru. V zdrojovom kóde si všimnime SQL príkaz SELECT pre nadriadenú tabuľku autorov a podriadenú tabuľku titulov

```
select au_lname as [Last Name], au_fname as [First Name], Address, City, State from Authors
order by [Last Name]
```

a podriadenú tabuľku titulov

```
select title as Title, price as Price, ytd_sales as [YTD Sales] from titleview where au_lname =
'" & filterValue & "'"
```

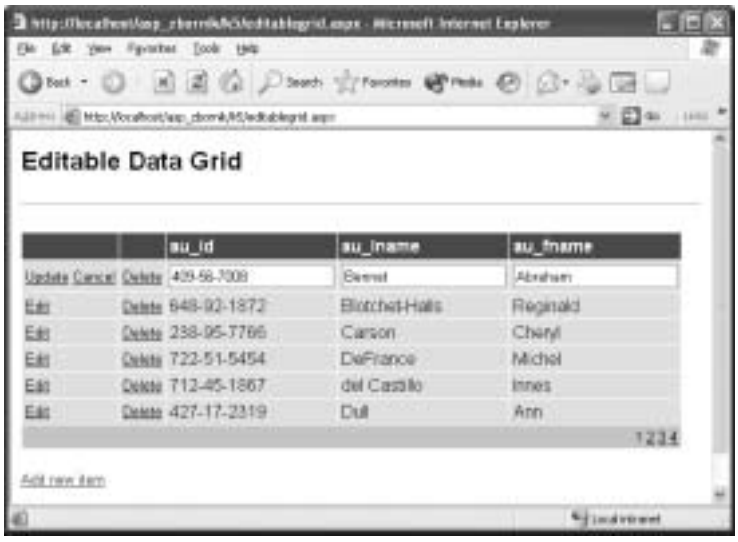
Najlepšie tento vzťah pochopíme ak si aplikáciu vyskúšame a budeme klikať v jednotlivých záznamoch na link Show details.



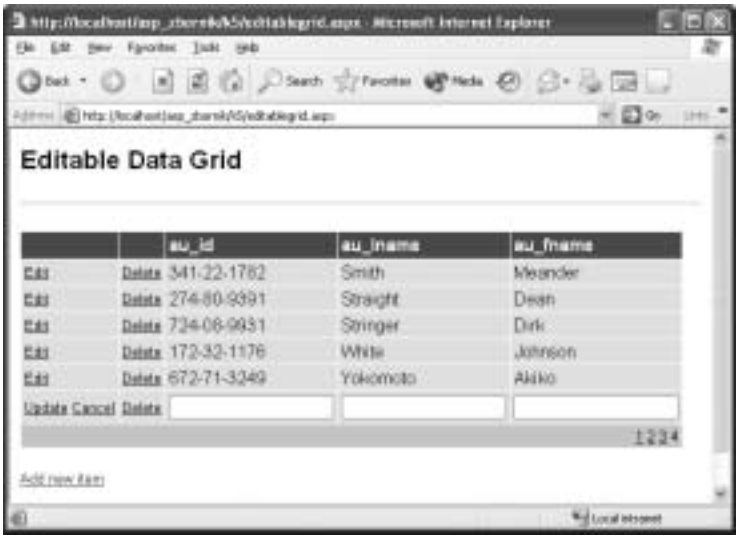
Aplikácia Master Detail

Editovateľný výpis údajov z databázovej tabuľky

Do kolekcie funkcionality práce s údajmi v databázových tabuľkách v ASP.NET nám chýba ešte editovanie údajov a pridávanie nových údajov. Vedľa každého záznamu sú linky na jeho editovanie, prípadne vymazanie. V dolnej časti stránky je link na pridanie nového záznamu.



Editovanie údajov



Pridanie nového záznamu

Pre tvorbu aplikácií tohto typu je potrebné dobre si preštudovať kód tejto vzorovej aplikácie. Všimnime si hlavne ošetrenie vzniku potenciálnych chýb pri zápise údajov v procedúre DataGrid_Update

```
Sub DataGrid_Update(Sender As Object, E As DataGridCommandEventArgs)

  ...
    Try
         myConnection.Open()
         UpdateCommand.ExecuteNonQuery()

    Catch ex as Exception
         Message.Text = ex.ToString()

    Finally
         myConnection.Close()

    End Try

    ...

End Sub
```

Pokúsme sa vložiť napríklad prázdny záznam, ktorý pochopiteľne neprejde cez rôzne obmedzenia, ktoré boli definované pri návrhu databázovej tabuľky. Výsledkom je chybové hlásenie

System.Data.SqlClient.SqlException: INSERT statement conflicted with COLUMN CHECK constraint 'CK__authors__au_id__77BFCB91'.
The conflict occurred in database 'pubs', table 'authors', column 'au_id'.
The statement has been terminated. at System.Data.SqlClient.SqlCommand.ExecuteReader(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream) at System.Data.SqlClient.SqlCommand.ExecuteNonQuery() at ASP.EditableGrid_aspx.DataGrid_Update(Object Sender, DataGridCommandEventArgs E)

Samozrejme v reálnej aplikácii by sme to ošetrili kultúrnejšie, toto je len cvičný príklad.

WebMatrix nám svojou ponukou šablón najčastejšie používaných typov formulárov a reportov značne uľahčí situáciu pri vývoji webových aplikácií tohoto typu. Aplikáciu pomocou ktorej vytvoríme prehliadač údajov z dvoch databázových tabuliek, medzi ktorými je väzba typu master - detail napríklad pomocou WebMatrixu vytvoríme za niekoľko minút. Spomínané šablóny aplikácií majú aj tú výhodu, že sú naviazané na konkrétnu databázu a prakticky ihneď po vytvorení fungujú. To nám umožňuje rozsiahle experimentovanie napríklad aj metódou pokus - omyl. Po každom významnejšom zásahu do kódu aplikácie si môžeme overiť jeho dôsledky.

KAPITOLA 6:

ASP.NET Starter kits

Málokedy začíname s vývojom aplikácií ako sa ľudovo povie „na zelenej lúke“. Spravidla využijeme časti kódu, prípadne celé bloky, alebo komponenty z predtým vyvíjaných projektov. Dochádza jednak k zrýchleniu vývoja, čo môže priniesť nielen zníženie nákladov, ale hlavne konkurenčnú výhodu. Na druhej strane dochádza aj k zvýšeniu spoľahlivosti nakoľko využívame samozrejme len komponenty, ktoré sa v reálnej prevádzke predchádzajúcej aplikácie osvedčili. Okrem svojich vlastných komponentov vývojári používajú samozrejme aj produkty od rôznych firiem.

Projekt ASP.NET Starter Kits je koncipovaný ako množina príkladov alebo prototypových aplikácií pre ASP.NET. Jednotlivé časti projektu sa dajú získať z priloženého CD prípadne z webu z adresy www.asp.net. Projekt je rozdelený do niekoľkých častí

- Time Tracker Starter Kit
- Reports Starter Kit
- Community Starter Kit
- Commerce Starter Kit
- Portal Starter Kit

Všeobecné požiadavky na inštaláciu:
Jednotlivé časti ASP.NET Starter kitu vyžadujú aby na serveri, alebo vývojárskom počítači kde sa budú inštalovať boli nainštalované nasledovné softvérové komponenty:

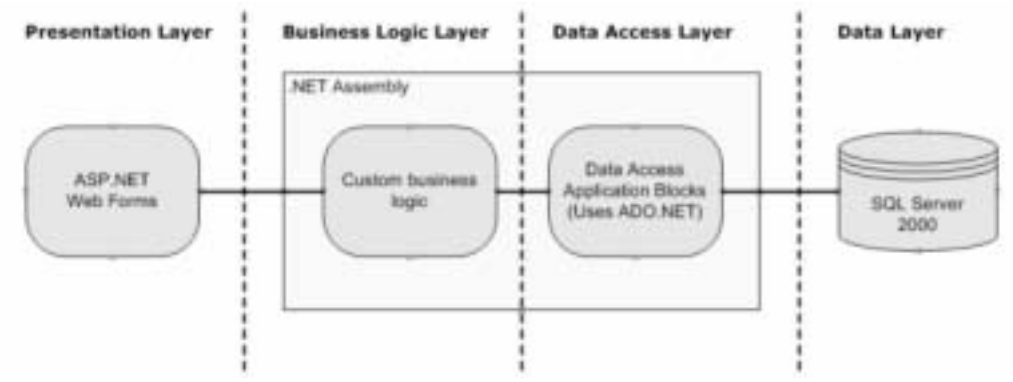
Windows 2000 or Windows XP (Server alebo Professional)
platforma Microsoft.NET Framevork vo verzii 1.0, alebo 1.1, pričom je doporučovaná nová verzia 1.1
SQL Server 2000 alebo Microsoft Data Engine (MSDE) 2000

Time Tracker Starter Kit a Portal Starter Kit navyše vyžadujú mať nainštalovaný Microsoft® Mobile Internet Toolkit, samozrejme len v prípade, ak k webu budeme pristupovať z mobilných zariadení a v prípade ak pracujeme s Visual Studiom 2002. Visual Studio .NET 2003 a .NET Framework majú podporu pre tvorbu webových mobilných aplikácií už zahrnutú.

ASP.NET Starter kits sú dodávané so zdrojovými soubormi jednak pre Visual Studio.NET, a taktiež aj pre WebMatrix. Starter kit určený pre Visual Studio.NET obsahuje príponu VS, kit určený pre WebMatrix obsahuje príponu SDK. Zdrojové kódy sú k dispozícii v jazykoch VB.NET, C# a J#.

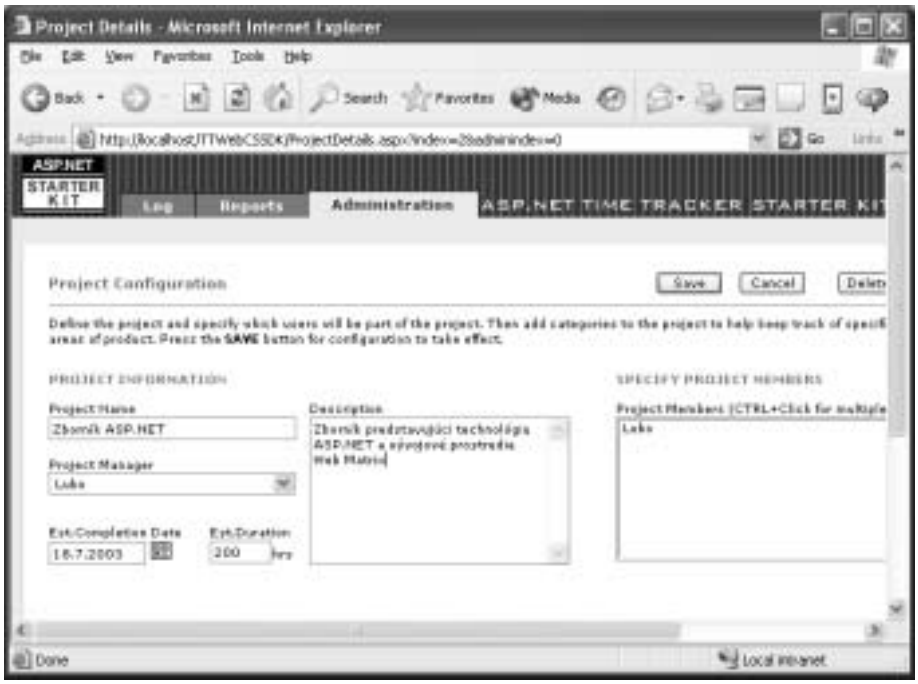
Time Tracker Starter Kit

Tento projekt môže poslúžiť pri aplikáciách pre plánovanie času a riadenie časového plánu úloh. Time Tracker Starter Kit pracuje jednak z projektom ako celkom a jednak s dielčími úlohami, z ktorých sa projekt skladá. Aplikácia umožňuje monitorovať priebeh projektu, prípadne dielčích úloh až po ich ukončenie. Môžeme vyhodnocovať „spotrebu“ času pre každý deň, vytvárať reporty, pomocou ktorých je možné dokladovať nadriadeným stav projektu a spotrebu času na jednotlivé dielčie úlohy. Tieto informácie je možné vygenerovať aj v grafickej podobe. Projekt Time Tracker Starter Kit samozrejme môžeme použiť ako celok, teda v takej podobe ako bol vytvorený, čo bude zrejme zriedkavejší prípad, alebo môžeme z neho využiť len tie časti a komponenty, ktoré potrebujeme, prípadne ich môžeme upraviť podľa svojich požiadaviek. Architektúra balíka je navrhnutá ako trojvrstvová

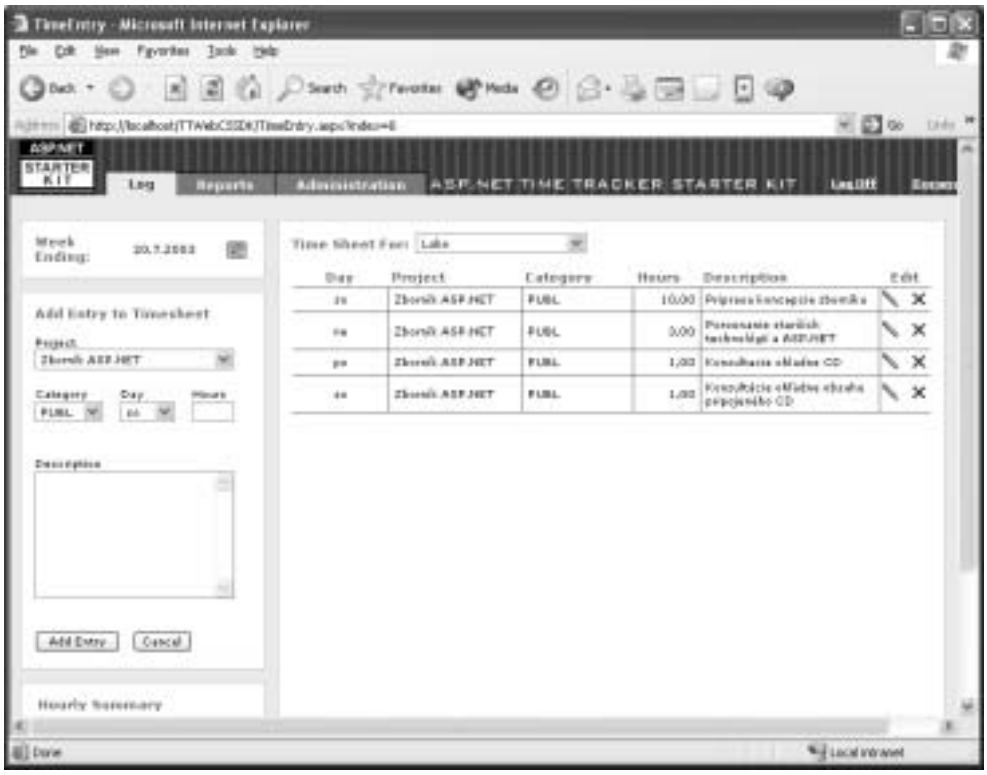


ASP.NET Starter Kit Time Tracker architektúra

Informácie o projekte sú veľmi citlivé, v niektorých prípadoch majú priam strategický význam, preto prístup do systému je riešený pomocou Windows autentifikácie.

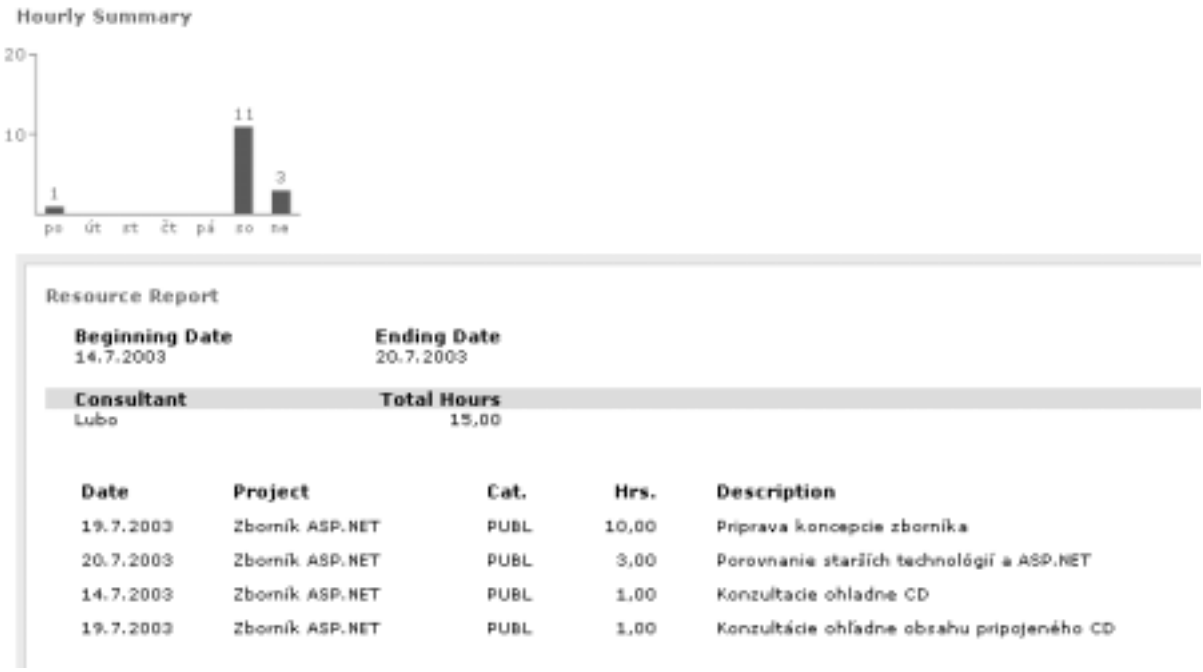


Vytvorenie nového projektu



ASP.NET Starter Kit Time Tracker

Samozrejme najdôležitejšie sú vnútorné vrstvy, ktoré tvoria kostru aplikácie. U prezentačnej vrstvy je zas dôležitá grafická úprava a zrozumiteľnosť. Údaje môžeme prezentovať vo forme reportov a grafov



Zobrazovanie údajov vo forme reportov a grafov

Inštalácia

ASP.NET Starter Kit Time Tracker sa inštaluje zo súboru asp.net time tracker (cssdk) installer v1.0.msi, krorý má niečo cez megabajt.

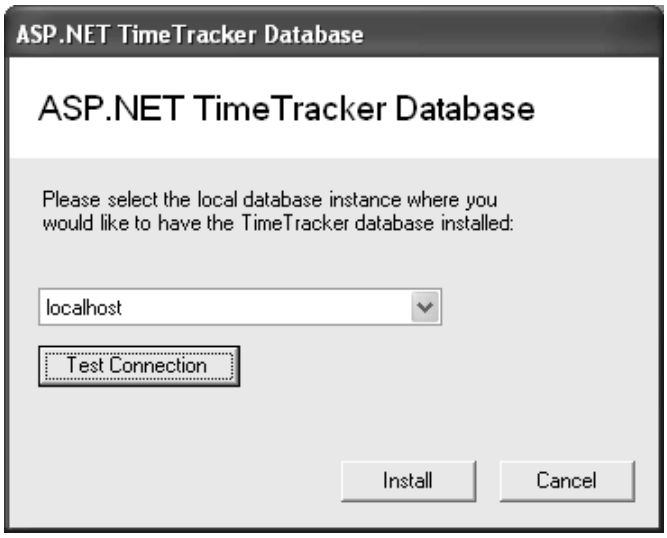
ASP.NET Starter Kit Time Tracker Inštalácia

Poznámky k inštalácii:

Budú vytvorené virtuálne adresáre ***TTWebCSSDK*** a ***TTMobileCSSDK***
Pre inštaláciu typu „Local“ bude vytvorená databáza ***TimeTracker***
Pre inštaláciu typu „Remote“ bude vytvorený pripojovací reťazec na databázu a bude potrebné spustiť SQL skript pre vytvorenie a naplnenie objektov v databáze.

Použitie:

Prístup k ASP.NET Starter Kit Time Tracker je cez URL adresy
http://localhost/TTWebCSSDK/ a http://localhost/TTMobileCSSDK/
K zdrojovým kódom môžeme pristupovať napríklad cez menu
Start > Programs > ASP.NET Starter Kits alebo priamo do adresára
c:\Program Files\ASP.NET Starter Kits\ASP.NET TimeTracker (CSSDK)\



Konektivita na databázu počas inštalácie

Databáza

Po nainštalovaní Starter Kitu sa vytvorí nová databáza TimeTracker. Štruktúra databázy je jasná z diagramu.

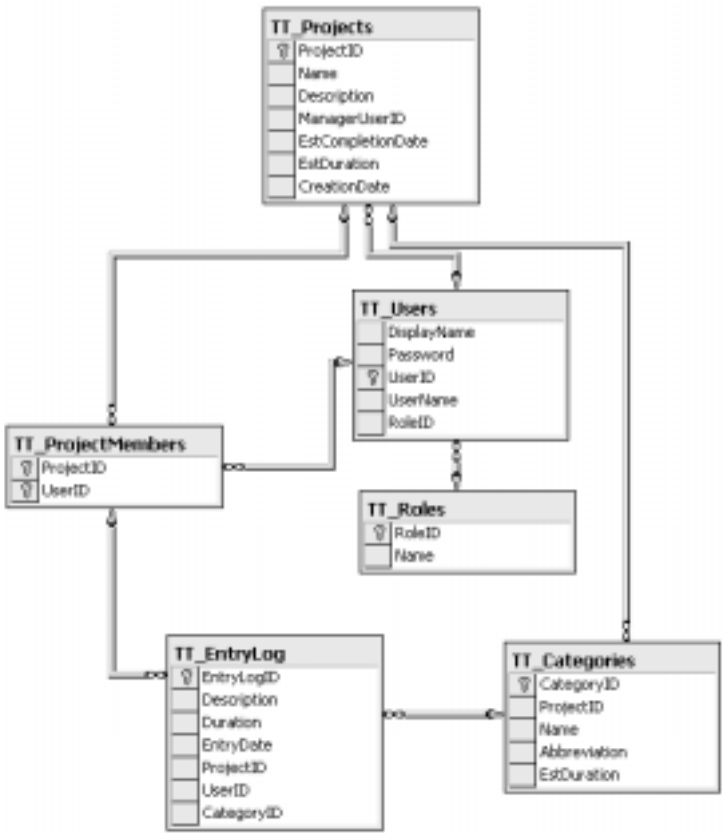


Diagram databázy TimeTracker

Pre prácu so Starter Kitom je potrebné aspoň v hrubých rysoch túto databázu poznať, preto ukážeme jej návrhovú štruktúru (pre jednoduchosť bez constraintov)

Tabuľka Projects

```
CREATE TABLE TT_Projects
(
       ProjectID int IDENTITY (1, 1) NOT NULL ,
       Name nvarchar (50),
       Description nvarchar (1024),
       ManagerUserID int NULL,
       EstCompletionDate datetime NULL ,
       EstDuration decimal(10, 2) NULL ,
       CreationDate datetime NOT NULL DEFAULT (getdate()),
)
```

možno lepšiu predstavu o tom, čo tabuľka obsahuje získame ak si necháme vypísať jej obsah (výpis sme zvislým rezom rozdelili na dve časti)

```
ProjectID    Name             Description
-----------  ---------------  ------------------------------------------------------
1            Sample Project   This is a sample project created during installation

ManagerUserID EstCompletionDate         EstDuration  CreationDate
------------- ------------------------ ------------  ----------------------
1              2003-05-18 08:05:03.947  500.00        2003-03-18 08:05:03.947
```

Tabuľka TT_ProjectMembers

```
CREATE TABLE TT_ProjectMembers
(
       ProjectID int NOT NULL,
       UserID] int NOT NULL
)
```

```
Tabuľka po inštalácii obsahuje jeden záznam

ProjectID   UserID
----------- -----------
1            1
```

Tabuľka TT_Users

```
CREATE TABLE TT_Users
(
       DisplayName nvarchar (50) NULL,
       Password nvarchar (50) NULL,
       UserID int IDENTITY (1, 1) NOT NULL,
       UserName nvarchar (50) NOT NULL ,
       RoleID int NOT NULL
)
```

Tabuľka TT_Roles

```
CREATE TABLE TT_Roles
(
       RoleID int NOT NULL,
       Name nvarchar (50)
)
```

Tabuľka TT_EntryLog

```
CREATE TABLE TT_EntryLog
(
       EntryLogID int IDENTITY (1, 1) NOT NULL,
       Description nvarchar (255) COLLATE NOT NULL,
       Duration decimal(10, 2) NOT NULL,
       EntryDate smalldatetime NOT NULL,
       ProjectID int NOT NULL,
       UserID int NOT NULL,
       CategoryID int NOT NULL,
)
```

Znovu pomôže v získaní presnejšej predstavy ukážka údajov z tejto tabuľky

```
EntryLogID  Description           Duration   EntryDate            ProjectID   UserID  CategoryID
----------- -------------------- ---------- -------------------- ----------- ------- ----------
1           Site Development      5.00       2003-03-19 08:05:00  1            1       1
2           Architecture Design   6.00       2003-03-20 08:05:00  1            1       1
3           Code Review           7.00       2003-03-21 08:05:00  1            1       1
```

Tabuľka TT_Categories

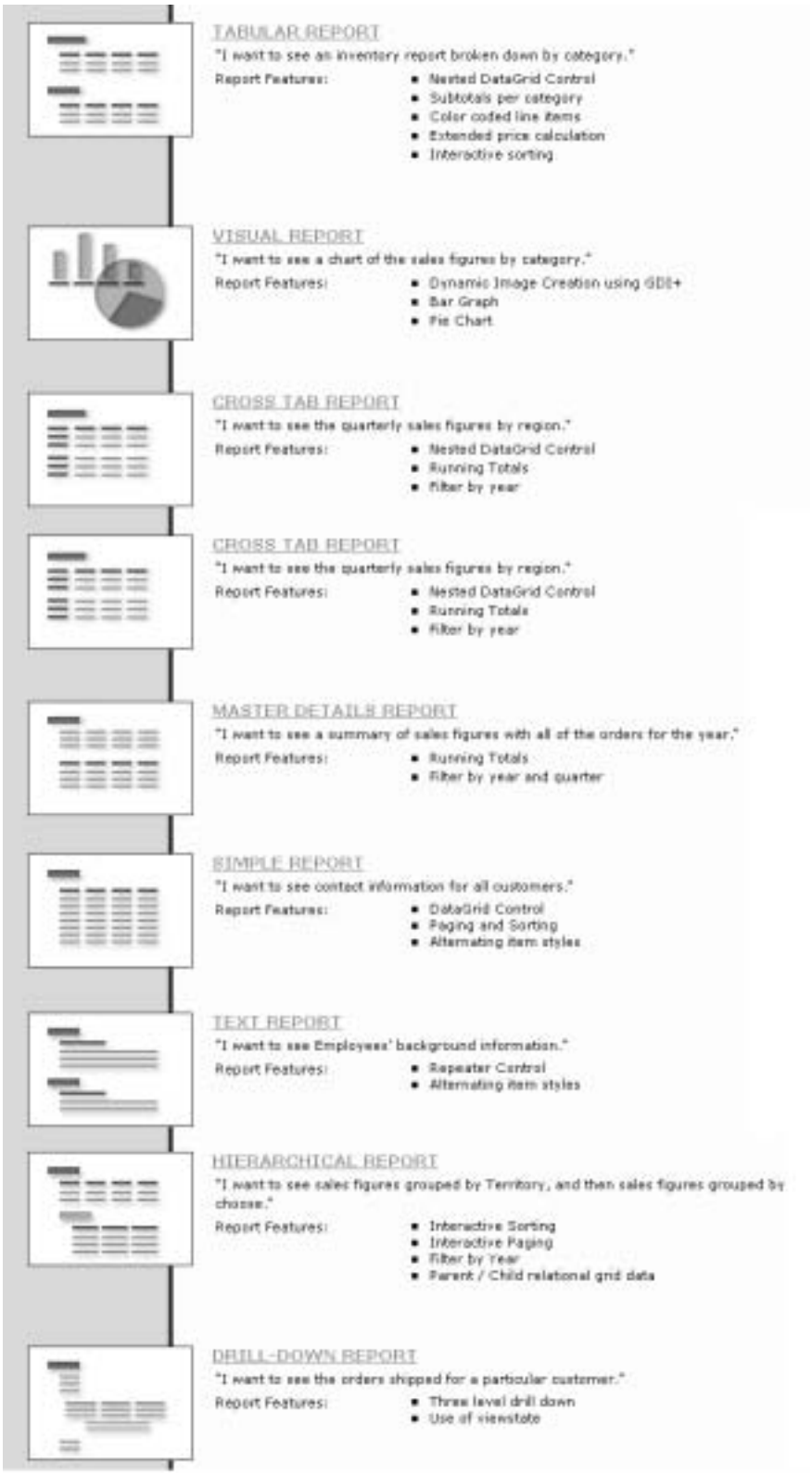
```
CREATE TABLE TT_Categories
(
       CategoryID int IDENTITY (1, 1) NOT NULL ,
       ProjectID int NOT NULL ,
       Name nvarchar (50),
       Abbreviation nvarchar (4),
       EstDuration decimal(10, 2) NULL ,
)
```

Príklad údajov

```
CategoryID  ProjectID   Name                      Abbreviation  EstDuration
----------- ----------- ------------------------- ------------  ------------
1            1          Development               DEV           250.00
2            1          Design                    DSN           100.00
3            1          Testing and Deployment    TEST          150.00
```

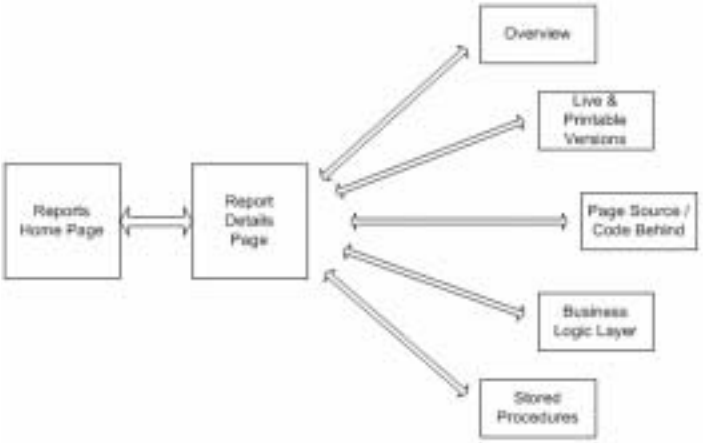
Reports Starter Kit

Veľmi často sa stretávame s úlohou vypísať do kontextu webovej stránky rôzne údaje prevažne z databáz, či už vo forme tabuliek, grafov, reportov a podobne. Pre tento účel slúži komponenta ASP.NET Starter Kit Reports. Spravidla ako podklad pre vytvorenie reportu slúžia údaje uložené v databázových tabuľkách.



ASP.NET Starter Kit Reports - typy reportov

Pri návrhu aplikácie využívajúcej pre zobrazovanie údajov reporty môže poslúžiť tento diagram



Popis fungovania aplikácie využívajúcej reporty- flow diagram

Poznámky k inštalácii:

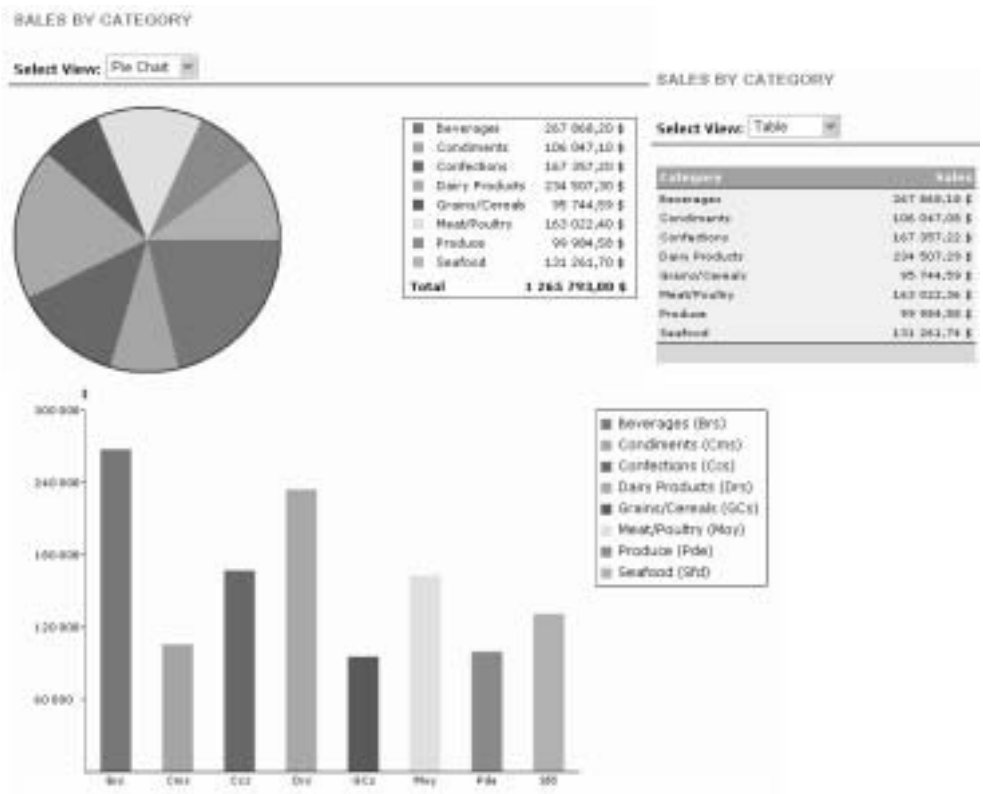
Pri inštalácii bude vytvorený virtuálny adresár ***ReportsCSSDK***
Pre inštaláciu typu „Local“ bude vytvorená databáza ***Reports***
Pre inštaláciu typu „Remote“ bude vytvorený pripojovací reťazec na databázu a bude potrebné spustiť SQL skript pre vytvorenie a naplnenie objektov v databáze.

Použitie:

Prístup k aplikácii je cez URL adresu http://localhost/ReportsCSSDK/
K zdrojovým kódom môžeme pristupovať napríklad cez menu
Start > Programs > ASP.NET Starter Kits alebo priamo do adresára
c:\Program Files\ASP.NET Starter Kits\ASP.NET Reports (CSSDK)\

Databáza

S databázou Reports sa v tomto prípade zoznamovať nemusíme, táto databáza je prakticky kópiou známej cvičnej databázy Northwind, ktorá sa nainštaluje s SQL Serverov. Report Starter Kit túto databázu sám nevyužíva, je na ňu naviazaný preto, lebo cvičná aplikácia vytvára reporty nad údajmi z tejto databázy. V praxi by sme sa samozrejme pripojili na našu vlastnú databázu. Môžeme generovať reporty vo forme rôznych tabuliek vrátane hierarchických a tabuliek typu master - detail, grafov a podobne.



Rôzne formy grafickej a tabuľkovej prezentácie údajov

Community Starter Kit
Tento balík je výhodné použiť pre vytvorenie webového sídla nejakej komunity, napríklad komunity vývojárov, zberateľov, používateľov nejakých zariadení a podobne. Je to aplikácia určená napríklad pre webhostingové servery a podobne

Community Starter Kit

Inštalácia

Počas inštalácie bude vytvorený virtuálny adresár CommunityStarterKitCSSDK
a bude vytvorená databáza CommunityStarterKit

Použitie:

Prístup k aplikácii je cez URL adresu http://localhost/CommunityStarterKitCSSDK/Default.aspx
sídlo sa administruje po prístupe na adresu http://localhost/CommunityStarterKitCSSDK/Admin
Prístupové parametre sú po inštalácii nastavené nasledovne:
username: ISPAdmin
password: ISPPassword

K zdrojovým kódom môžeme pristupovať cez menu
Start > Programs > ASP.NET Starter Kits
alebo priamo do adresára
c:\Program Files\ASP.NET Starter Kits\ASP.NET Community Starter Kit (CSSDK)\

Po úspešnej inštalácii je potrebné aplikáciu nakonfigurovať, teda inými slovami povedané, ušiť na mieru príslušnej komunity. Po prvom prístupe na URL adresu

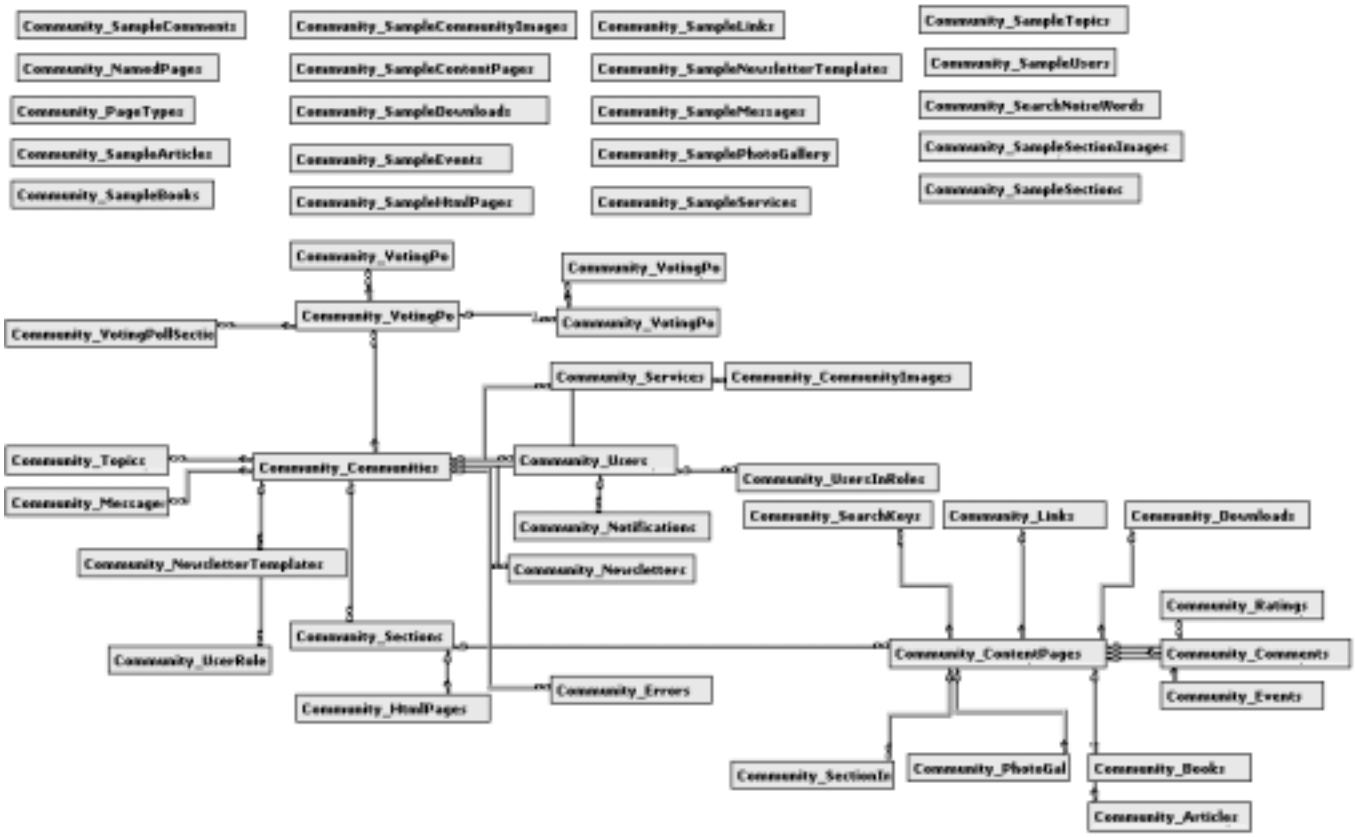
http://localhost/CommunityStarterKit/Default.aspx
sa objaví len lakonický oznam:

To setup this community, please configure a community for localhost by clicking **here**.
Community ID: 1

Po kliknutí na odkaz prejdeme na adresu administrátorského prístupu:

http://localhost/CommunityStarterKit/ISPAdmin/IspLogin.aspx

Konfigurácia a administrácia projektu Community nie je zrovna jednoduchou záležitosťou, no aj tak predstavuje dobrý kompromis medzi univerzálnosťou a zložitosťou. Podrobný popis je mimo možností rozsahu tejto publikácie, veď len databáza obsahuje 50 tabuliek (viď schéma)

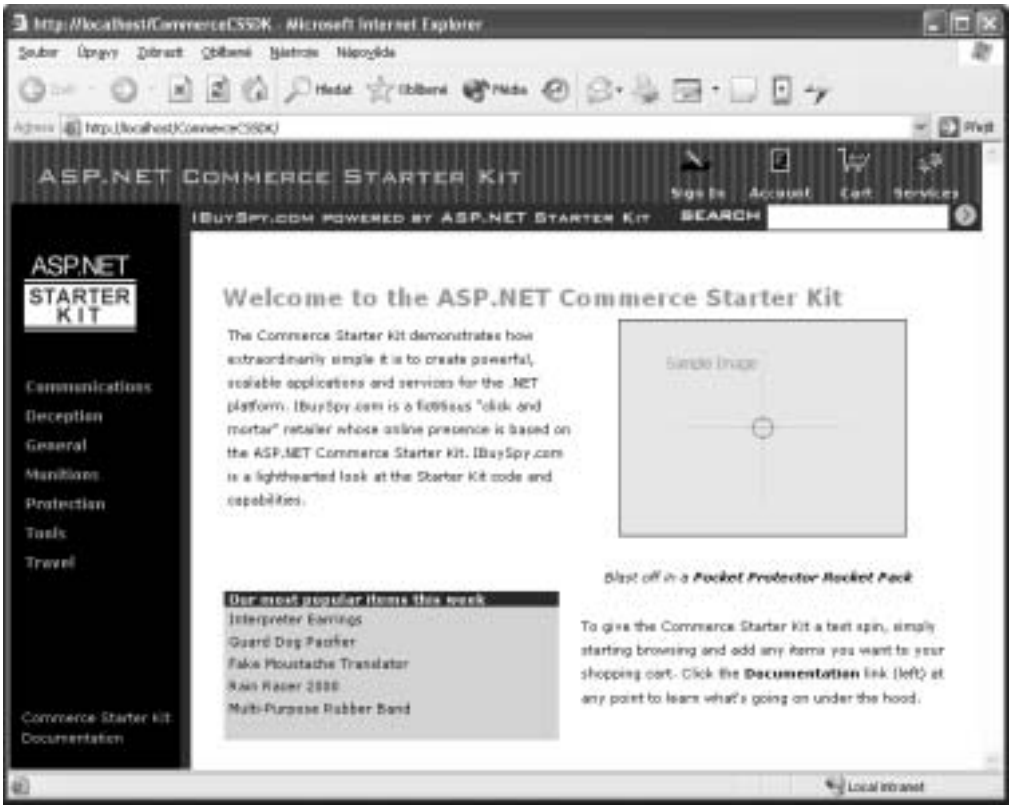


Štruktúra databázových tabuliek

Pri administrácii a konfigurácii je potrebné postupovať podľa dokumentácie, ktorá sa nainštaluje spolu s Community Starter Kitom

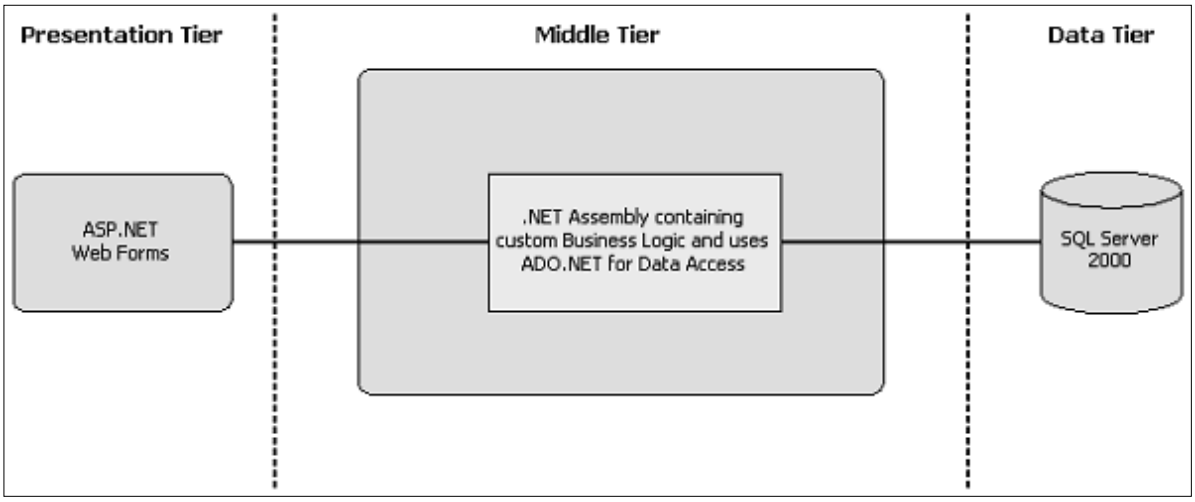
Commerce Starter Kit

Azda nikoho netreba presviedčať o tom, že Internet by sa zaručene nerozvinul do tej podoby ako ho poznáme dnes bez komerčných aplikácií. Nielen potrebný výkonný hardvér je veľmi drahý, ale aj kvalifikovaní administrátori a vývojári webových projektov sa podieľajú na pomerne vysokých nákladoch. Tieto náklady je možné hradiť len z komerčných aplikácií, prípadne z inzercie, preto je dôležitá aj aplikácia typu „internetový obchod“, ktorá umožňuje predávajúcemu prostredníctvom webových katalógov ponúkať svoj tovar a kupujúcim zasa nakupovať a samozrejme nejakou formou za tovar platiť. Pre tento účel bola vyvinutá aj pomerne jednoduchá aplikácia Commerce Starter Kit.



Commerce Starter Kit

Aj v tomto prípade je použitá trojvrstvová architektúra podľa schémy



Trojvrstvová architektúra

Inštalácia

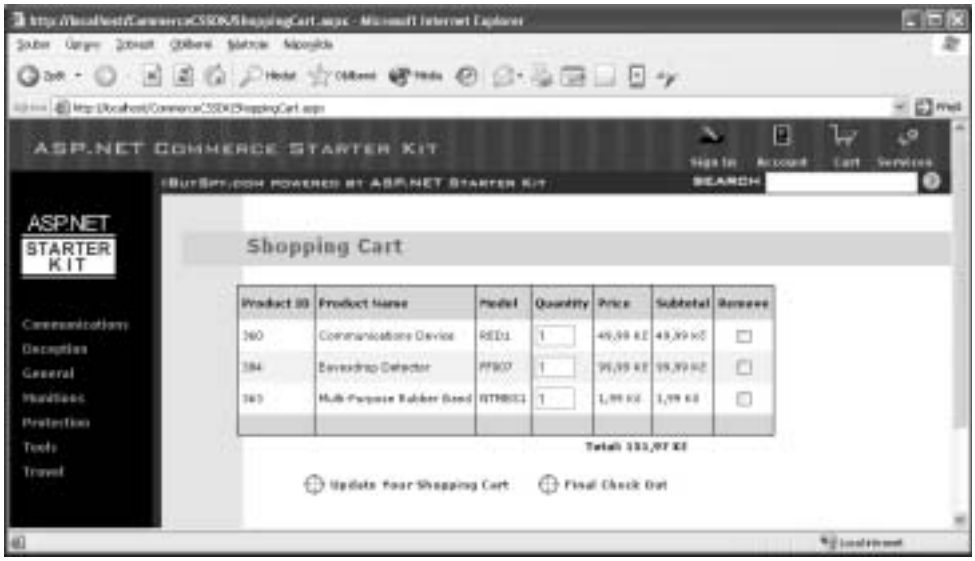
ASP.NET Starter Kit Commerce sa inštaluje zo súboru asp.net commerce (cssdk) installer v1.0.msi, pričom jeho veľkosť je okolo pol megabajtu

Poznámky k inštalácii:

Bude vytvorený virtuálny adresár CommerceCSSDK
Pre inštaláciu typu „Local“ bude vytvorená databáza Commerce
Pre inštaláciu typu „Remote“ bude vytvorený pripojovací reťazec na databázu a bude potrebné spustiť SQL skript pre vytvorenie a naplnenie objektov v databáze.

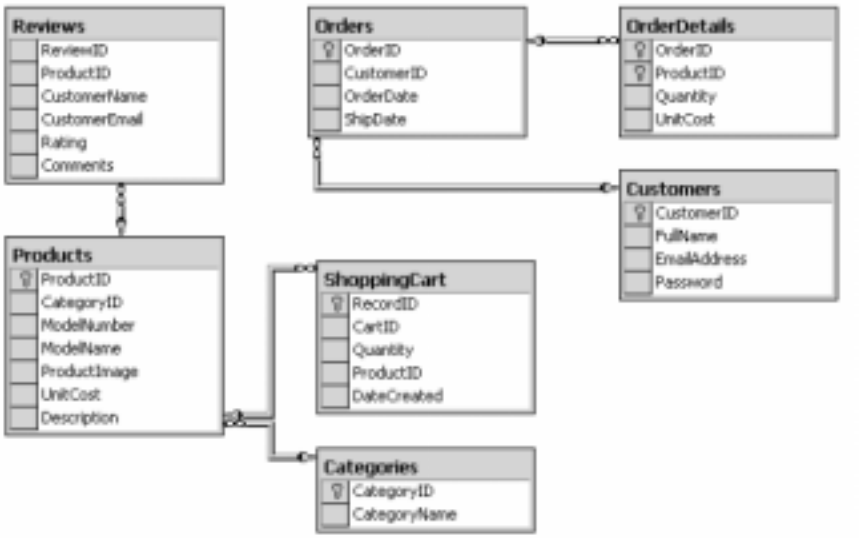
Použitie:

Prístup k Commerce Starter Kit je cez URL adresu
http://localhost/CommerceCSSDK/
K zdrojovým kódom môžeme pristupovať napríklad cez menu
Start > Programs > ASP.NET Starter Kits alebo priamo do adresára
c:\Program Files\ASP.NET Starter Kits\ASP.NET Commerce (CSSDK)\



Commerce Starter Kit – obsah nákupného košíka

Databáza



Commerce Starter Kit – databázové tabuľky

Pretože aplikácie tohoto typu sú veľmi často používané, veď na internete sa obchoduje prakticky s čímkoľvek, predstavíme jednotlivé databázové tabuľky.

Tabuľka **Products**

```
CREATE TABLE Products

       ProductID int IDENTITY (1, 1) NOT NULL ,
       CategoryID int NOT NULL ,
       ModelNumber nvarchar (50),
       ModelName nvarchar (50),
       ProductImage nvarchar (50),
       UnitCost money NOT NULL ,
       Description nvarchar (3800),

)
```

```
ProductID CategoryID ModelNumber ModelName           UnitCost   Description
--------- ---------- ----------- ----------------    ---------- -------------------------
355       16         RU007       Rain Racer 2000     1499.9900  Looks like an ordinary bumbershoot,
                                                                  but don't be fooled! Simply place
356       20         STKY1       Edible Tape          3.9900    The latest in personal survival gear,
                                                                  the STKY1 looks like a roll of
357       16          P38        Escape Vehicle(Air)  2.9900    In a jam, need a quick escape?
                                                                  Just whip out a sheet of our patented
358       19          NOZ119     Extracting Tool     199.0000   High-tech miniaturized extracting tool.
                                                                  Excellent for extricating for
```

Tabuľka **Categories**

```
CREATE TABLE Categories
(
       CategoryID int IDENTITY (1, 1) NOT NULL ,
       CategoryName nvarchar (50),

)
```

```
CategoryID  CategoryName
----------- -----------------
14           Communications
15           Deception
16           Travel
17           Protection
18           Munitions
19           Tools
20           General
```

Tabuľka **Reviews**

```
CREATE TABLE Reviews
(
       ReviewID int IDENTITY (1, 1) NOT NULL ,
       ProductID int NOT NULL ,
       CustomerName nvarchar (50),
       CustomerEmail nvarchar (50),
       Rating int NOT NULL ,
       Comments nvarchar (3850),
)
```

```
ReviewID  ProductID CustomerName    CustomerEmail    Rating  Comments
--------- --------- --------------- ---------------- ------- --------------------------------
21        404       Sarah Goodpenny sg@ibuyspy.com   5       Really smashing!  Don't know
                                                              how I'd get by without them!
22        378       James Bondwell  jb@ibuyspy.com   3       Well made, but only moderately
                                                              effective.  Ouch!
```

Tabuľka **Customers**

```
CREATE TABLE Customers
(
       CustomerID int IDENTITY (1, 1) NOT NULL ,
       FullName nvarchar (50),
       EmailAddress nvarchar (50),
       Password nvarchar (50),
)
```

```
CustomerID  FullName             EmailAddress         Password
----------- ------------------- ------------------- -------------
1            James Bondwell      jb@ibuyspy.com      IBS_007
2            Sarah Goodpenny     sg@ibuyspy.com      IBS_001
3            Gordon Que           gq@ibuyspy.com     IBS_000
19           Guest Account       guest               guest
16           Test Account        d                   d
```

Tabuľka **Orders**

```
CREATE TABLE Orders
(
       OrderID int IDENTITY (1, 1) NOT NULL ,
       CustomerID int NOT NULL ,
       OrderDate datetime NOT NULL ...DEFAULT (getdate(),
       ShipDate datetime NOT NULL ...DEFAULT (getdate(),
)
```

```
OrderID     CustomerID  OrderDate    ShipDate
----------- ----------- ------------ ------------
99           19          2000-07-06   2000-07-07
93           16          2000-07-03   2000-07-04
101          16          2000-07-10   2000-07-11
103          16          2000-07-10   2000-07-10
96           19          2000-07-03   2000-07-03
104          19          2000-07-10   2000-07-11
```

Tabuľka **OrderDetails**

```
CREATE TABLE OrderDetails
(
       OrderID int NOT NULL ,
       ProductID int NOT NULL ,
       Quantity int NOT NULL ,
       UnitCost money NOT NULL ,

)
```

```
OrderID      ProductID    Quantity     UnitCost
-----------  -----------  -----------  ---------------------
99            404          2            459.9900
93            363          1            1.9900
101           378          2            14.9900
102           372          1            129.9900
96            378          1            14.9900
103           363          1            1.9900
104           355          1            1499.9900
104           378          1            14.9900
```

Tabuľka **ShoppingCart**

```
CREATE TABLE ShoppingCart
(
       RecordID int IDENTITY (1, 1) NOT NULL ,
       CartID nvarchar (50),
       Quantity int NOT NULL ... DEFAULT (1),
       ProductID int NOT NULL ,
       DateCreated datetime NOT NULL ... DEFAULT (getdate()),
)
```

```
RecordID     CartID                                Quantity     ProductID    DateCreated
-----------  ------------------------------------  -----------  -----------  -------------------
1             6c856fe2-0602-4e0f-a507-dc200c5bde43  1             360          2003-03-27 07:23:11
2             6c856fe2-0602-4e0f-a507-dc200c5bde43  1             363          2003-03-27 07:23:21
3             6c856fe2-0602-4e0f-a507-dc200c5bde43  1             384          2003-03-27 07:23:24
```

Portal Starter Kit

Publikačný portál sa pre jeho pre jeho univerzálnosť a množstvo modifikácií považuje za takmer „kultovú“ webovú aplikáciu. Existuje množstvo riešení, či už na platforme platforme ASP stránok a MS SQL Servera 2000, alebo riešení na platforme PHP a MySQL, teda vhodné hlavne pre LINUX ale aj Windows. Preto publikačný a webový portál nesmie chýbať ani v balíku ASP.NET Starter Kit

Využitie publikačného portálu je oveľa širšie, než by sa na prvý pohľad zdalo. Pozrime si stránky niektorých inštitúcií, stránky zberateľov a podobne a čo zistíme. Vyskytujú sa tu rôzne texty, príspevky, informácie o novinkách... obvykle zoradené do rôznych tematických skupín a kategórii. Všetko webové projekty na princípe publikačného portálu..

Portal Starter Kit

Inštalácia

Bude vytvorený virtuálny adresár PortalCSSDK
Pre inštaláciu typu „Local“ bude vytvorená databáza Portal
Pre inštaláciu typu „Remote“ bude vytvorený pripojovací reťazec na databázu a bude potrebné spustiť SQL skript pre vytvorenie a naplnenie objektov v databáze.

Použitie:

Prístup k Portal Starter Kit je cez URL adresu
http://localhost/PortalCSSDK/
K zdrojovým kódom môžeme pristupovať napríklad cez menu
Start > Programs > ASP.NET Starter Kits alebo priamo do adresára
c:\Program Files\ASP.NET Starter Kits\ASP.NET Commerce (CSSDK)\

Z hľadiska využitia zobrazovacej plochy v okne prehliadača HTML stránok, je možné obrazovku publikačného portálu rozdeliť na 7 častí

1. **Autentifikačný modul:** je zobrazený automaticky, ak sa používateľ portálu ešte neautentifikoval.
2. **Modul odkazov na spriatelené stránky:** Implicitne je naplnený odkazmi na stránky s tematikou ASP.NET
3. **HTML modul:** Zobrazenie hypertextového obsahu, napríklad vybraného článku, príspevku a podobne. Ak k portálu pristupujeme z mobilného zariadenia, zobrazí sa len textový obsah.
4. **Ponukový modul:** obsahuje aktuálne ponuky, informácie o novinkách a prehľady
5. **Modul pre pripomínanie udalostí:** obsahuje upomienky a oznamy ohľadne akcií, ktoré sa pripravujú, pričom je zobrazená informácia o čase a mieste konania udalosti a jej stručný popis.
6. **Pomocný HTML modul:** spravidla obsahuje obrázok a stručnú textovú informáciu
7. **XML modul:** Výsledok zobrazený v tomto okne je výsledkom XSL/T transformácie obsahu XML súboru.

Rozdelenie obrazovky publikačného portálu

Celý obsah portálu je uložený v databázových tabuľkách. Pre lepšie zorientovanie, nakoľko publikačný portál je veľmi často nasadzovanou aplikáciou si jednotlivé tabuľky stručne predstavíme vo forme zjednodušených príkazov pre ich vytvorenie (bez constraintov)

Databázové tabuľky sa viac či menej budú týkať niektorej zo siedmych dosiaľ vymenovaných častí pracovnej obrazovky. Prvé tri tabuľky (**`Users`**, **`Roles`** a **`UserRoles`**) sa vzťahujú k administrácii používateľov a k ich prístupovým právam, teda k modulu číslo 1

1. Autentifikačný modul.

```
CREATE TABLE Users
(
       UserID int IDENTITY (1, 1) NOT NULL ,
       Name nvarchar (50),
       Password nvarchar (20),
       Email nvarchar (100)
)
```

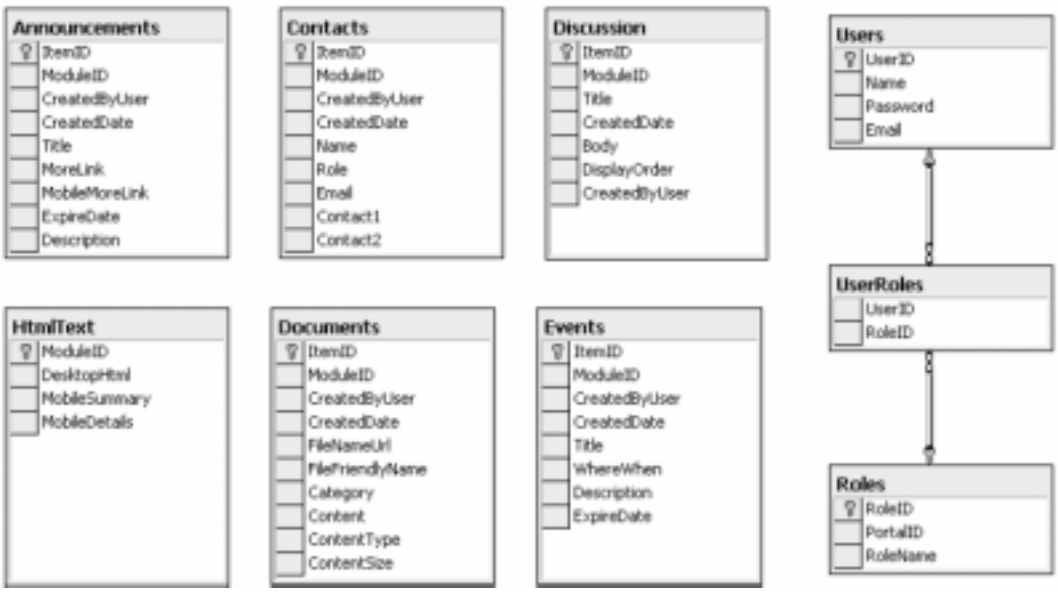
```
UserID      Name                  Password             Email
----------- --------------------- -------------------- ----------------------
1            Guest                 guest                guest
2            Luboslav Lacko        lubo                 lubolacko@post.sk
```

```
CREATE TABLE Roles
(
       RoleID int IDENTITY (0, 1) NOT NULL ,
       PortalID int NOT NULL ,
       RoleName nvarchar (50)
)
```

```
RoleID      PortalID    RoleName
----------- ----------- ------------
0            0           Admins
```

```
CREATE TABLE UserRoles
(
       UserID int NOT NULL ,
       RoleID int NOT NULL

)
```

```
UserID      RoleID
----------- -----------
1            0
```



Štruktúra databázy publikačného portálu

2. Modul odkazov na spriatelené stránky

K tomuto modulu sa vzťahuje tabuľka **Links**.

```
CREATE TABLE Links
(
       ItemID int IDENTITY (0, 1) NOT NULL ,
       ModuleID int NOT NULL ,
       CreatedByUser nvarchar (100),
       CreatedDate datetime NULL ,
       Title nvarchar (100),
       Url nvarchar (250),
       MobileUrl nvarchar (250),
       ViewOrder int NULL ,
       Description nvarchar (2000)
)
```

ItemID	ModuleID	CreatedByUser	Title	Url
0	0	NULL	NULL	NULL
1	1	JennaJ@ibuyspy.com	ASP.NET Site	http://www.asp.net
2	1	JennaJ@ibuyspy.com	GotDotNet.com	http://www.gotdotnet.com
3	1	JennaJ@ibuyspy.com	ASP.NET on MSDN	http://msdn.microsoft.com/net/aspnet
4	1	JennaJ@ibuyspy.com	QuickStart Samples	http://www.gotdotnet.com/quickstart/

Údaje uložené v tabuľke **Links** sa zobrazia na HTML stránke vo forme tabuľky nasledovne

Quick Launch
ASP.NET Site
GotDotNet.com
ASP.NET on MSDN
QuickStart Samples

3. HTML modul

Príspevky v okne HTML modulu sú uložené v databázovej tabuľke **HtmlText**.

```
CREATE TABLE HtmlText
(
       ModuleID int NOT NULL ,
       DesktopHtml ntext,
       MobileSummary ntext,
       MobileDetails ntext
)
```

ModuleID	DesktopHtml	MobileSummary	MobileDetails
2	<table cellSpacing="0" cellPaddi...	Welcome to the <b>Portal Starter ...	This site can s
5	<span class="Normal"> The QL...	The QLT2112 <a href="http://www...	The QLT2112 &lt
7	<span class="Normal"> <im...		
11	<table cellspacing="0" cellpaddi...		
17	<span class="Normal"> &l...		
22	<table cellspacing=0 cellpadding=0 border=...	The <b>ASP.NET Portal Starter Kit...	

Obsah tabuľky HtmlText

Keďže v tabuľke HtmlText sú uložené jednotlivé príspevky v hypertextovej podobe, veľa toho v jednoduchom výpise neuvidíme, ukážeme časť HTML kódu obsahu jedného okna. Znaky `&lt` a `&gt` predstavujú znaky < a >

```
&lt;table cellSpacing=&quot;0&quot; cellPadding=&quot;5&quot; border=&quot;0&quot;&gt;
    &lt;tr valign=&quot;top&quot;&gt;
        &lt;td&gt;
            &lt;a target=&quot;_blank&quot; href=&quot;http://www.ibuyspy.com&quot;&gt;
                &lt; Welcome to the &lt;b&gt;Portal Starter Kit&lt;/b&gt;, the Intranet Home
for corporate employees or as your personal portal...
...
```

Údaje uložené v tabuľke **`HtmlText`** sa zobrazia v hypertextovej podobe

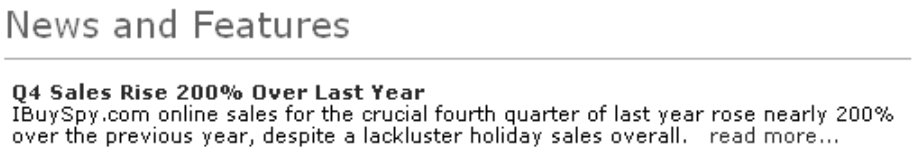
Modul HTML

4. Ponukový modul

Tento modul je založený na tabuľke **`Announcements`**.

```
CREATE TABLE Announcements
(
       ItemID int IDENTITY (0, 1) NOT NULL ,
       ModuleID int NOT NULL ,
       CreatedByUser nvarchar (100),
       CreatedDate datetime NULL ,
       Title nvarchar (150),
       MoreLink nvarchar (150),
       MobileMoreLink nvarchar (150),
       ExpireDate datetime NULL ,
       Description nvarchar (2000),
)

(vo výpise sú len dva stĺpce)

Title
-------------------------------------------
Q4 Sales Rise 200% Over Last Year

Description
-----------------------------------------------------
 IBuySpy.com online sales for the crucial fourth quarter
 of last year rose nearly 200% over the previous year,
 despite a lackluster holiday sales overall.
```

Na HTML stránke v okne klienta sa zobrazí



Ponukový modul

5. Modul pre pripomínanie udalostí

Tento modul je založený na tabuľke **`Events`**.

```
CREATE TABLE Events
(
       ItemID int IDENTITY (0, 1) NOT NULL ,
       ModuleID int NOT NULL ,
       CreatedByUser nvarchar,
       CreatedDate datetime NULL ,
       Title nvarchar (150),
       WhereWhen nvarchar (150),
       Description nvarchar (2000),
       ExpireDate datetime NULL
)
```

(výpis je rozdelený na dve časti zvislým rezom)

```
ItemID ModuleID CreatedByUser         Title
------ -------- -------------------- --------------------
0       0        NULL                 NULL
1       4        JennaJ@ibuyspy.com   Spy-o-Rama



2       4        JennaJ@ibuyspy.com   Dark Ops Sock Hop

WhereWhen                           Description
----------------------------------- --------------------------------------------------
NULL                                 NULL
This Saturday, usual secret
time and place...    It's back!     The premier regional swap meet for
                                    spy paraphernalia of every description.  Shop
                                    early for some amazing bargins.
Saturday, 8pm to ?, Dark Ops Cafe   Back by popular demand!  Practice your surveillance
                                    of the opposite sex, and dance some too.
                                    Great opportunity for a brush pass!
```

na HTML stránke bude ponukový modul zobrazený takto:

Modul pre pripomínanie udalostí

Databáza ďalej obsahuje tabuľku kontaktov na jednotlivých členov tímu, teda redaktorov portálu, tabuľku názvov dokumentov, napríklad súborov textového editora MS Word, ktoré je možné prezerať alebo stiahnuť a tabuľka pre ukladanie diskusných príspevkov k jednotlivým článkom

Podklady pre modul číslo **6. Pomocný HTML modul** ako vyplýva už z jeho názvu sú uložené v HTML súbore a podklady pre modul číslo **7. XML modul** sú uložené vo formáte XML a výsledok zobrazený v príslušnom okne je výsledkom XSL/T transformácie XML súboru

Administrácia a redakcia portálu
Pre úkony spojené s prevádzkou časopisu, budú členovia redakčného kolektívu pristupovať cez autorizačnú a autentifikačnú stránku redakcie. Administráciu portálu môžu vykonávať len používatelia, ktorí majú pridelenú rolu 0 Admin. Implicitne je táto rola pridelená používateľovi guest, ktorý sa vytvorí pri inštalácii Portal Starter Kitu.

```
UserID       Name                  Password             Email
-----------  --------------------- -------------------- ----------------------
1             Guest                 guest                guest
```

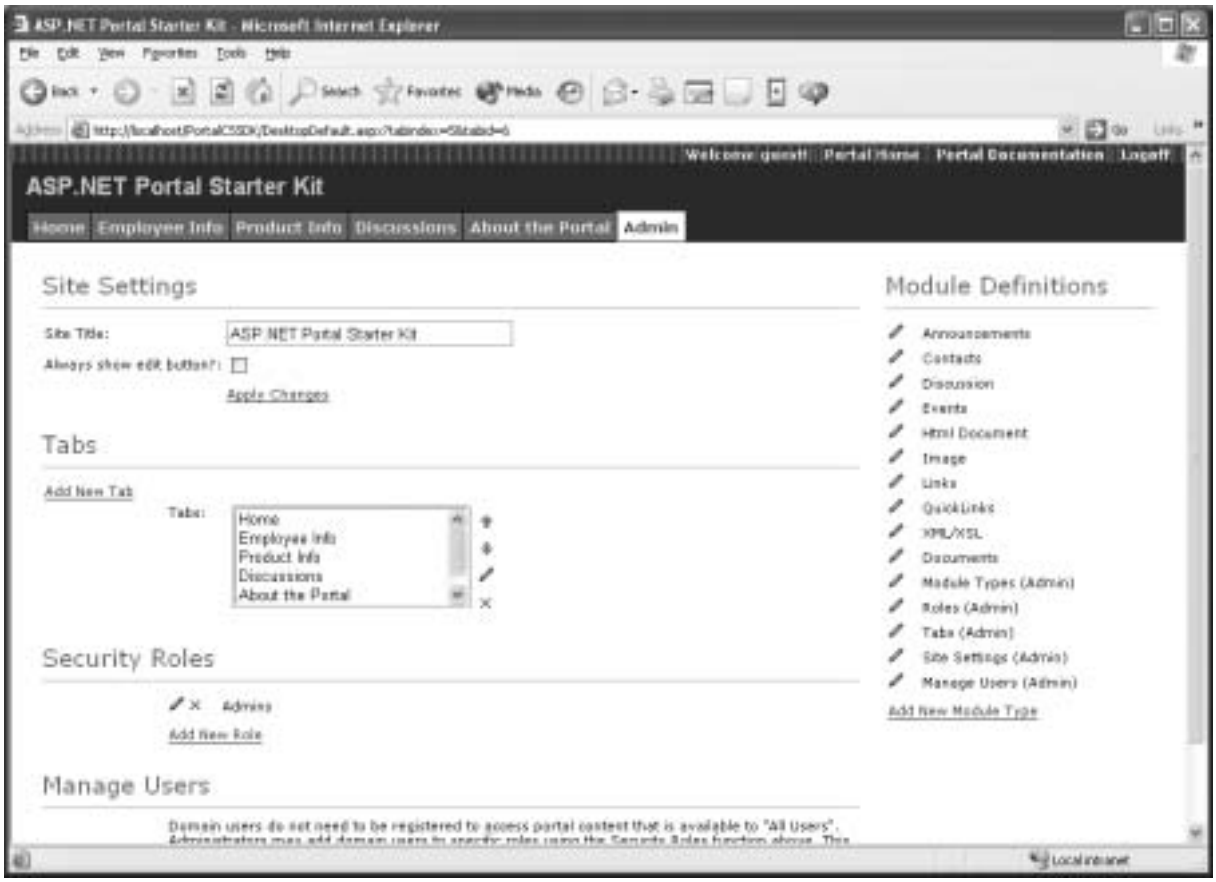
Pre prvé pokusy s administráciou portálu sa teda môžeme prihlásiť nasledovne:

Email: guest
Password: guest

Zobrazí sa nám mierne modifikovaná stránka portálu s pridanými odkazmi na editáciu a vkladanie článkov odkazov a podobne. Po aktivácii odkazu sa dostaneme na stránky zadávacieho formulára, kde môžeme napríklad zadávať a editovať článok.

Administrácia - vkladanie prípadne editovanie článku

Pri administrácii portálu môžeme meniť nielen obsah ale aj formu zobrazenia. Môžeme určovať, ktoré z modulov sa zobrazia a ktore nie, môžeme v určitom rozsahu ovplyvniť spôsob zobrazenia, môžeme meniť zdrojové HTML a XML súbory pre moduly číslo 6 a 7.



Modul HTML

Komponenty

Okrem komplexných riešení môžeme pri vývoji ASP.NET aplikácií do týchto aplikácií vhodne zakomponovať aj rôzne komponenty, ktoré môžeme získať z priloženého CD, prípadne z webu. Napríklad z adresy http://www.asp.net môžeme zdarma získať a použiť komponenty: **TreeView**, **Toolbar**, **TabStrip** a **Multipage**. Na inej adrese www.netdirect.cz/aspnet môžeme taktiež zdarma získať a použiť komponenty, **Anketa**, **Datagrid**, **Menu**, **WYSIWYG Editor** a **Upload Manager**.

KAPITOLA 7:
Webové služby

Skôr než vedieť presne a exaktne definovať webovú službu, je užitočnejšie vedieť na čo sa dajú webové služby použiť a ako sa dajú vytvárať. Napriek komplexnosti riešení, ktoré webové služby umožňujú, ide z hľadiska vývojára (aspoň v začiatkoch) o veľmi jednoduchú problematiku. Všeobecný princíp je asi takýto. Naša spoločnosť má prívlastok informačná a preto ľudia aj firmy sa delia do dvoch kategórií. Na tých čo informácie majú a sú ochotní (väčšinou za nejakú protihodnotu, alebo z propagačných a marketingových dôvodov) ich poskytnúť, a tých, ktorí o tieto informácie majú záujem. Vysvetlíme to na jednoduchom príklade. Na Slovensku vychádza veľa internetových denníkov, a nachádza sa tu aj mnoho portálov, na ktoré denne pristupujú klienti. Všetci prevádzkovatelia takýchto denníkov a portálov zverejňujú denne správy o počasí. Získavajú ich rôzne a vkladajú na svoje stránky. Tieto informácie však majú jednotný zdroj, buď sa jedná o hydrometeorologický ústav, alebo nejakú inú firmu, ktorá sa zaoberá predpoveďou počasia. Ak by takáto firma poskytla za rozumných obchodných podmienok údaje o predpovedi počasia, prevádzkovatelia internetových denníkov a portálov by ju určite radi využívali. Ide len o to, ako na to? Momentálny stav je zrejme taký, že poskytovateľ predpovede poskytne túto v nejakom formáte, napríklad textovom, alebo binárnom a webmastri to po prípadnej konverzii povkladajú na svoje stránky, každý v tom svojom formáte a grafickej úprave. Podobne sa to deje aj s kurzami valút u jednotlivých bánk, programom televíznych staníc... Kvalitatívne novým riešením je použitie webovej služby, kedy poskytovateľ poskytne svoje údaje vo formáte XML. Technológia XML je totiž otvorená a platformovo nezávislá. XML súbor obsahuje okrem samotných údajov aj ich štruktúru.. Zjednodušene by sa dalo povedať, že tento formát pre výmenu údajov je rovnako dobre čitateľný pre počítače aj pre ľudí, aj keď ideálny stav je taký, kedy si stroje medzi sebou vymenia údaje a používateľ o tom ani nevie. Údaje môžeme ľahko načítať a identifikovať pomocou pomerne jednoduchej rutiny v ľubovoľnom programovacom jazyku, alebo po definovaní formátu zobrazenia priamo publikovať na webovej stránke. XML dokument sa skladá z elementov. Každý element obsahuje počiatočnú a koncovú značku (tag) Obidva tagy obsahujú názov elementu, koncový tag obsahuje navyše pred názvom elementu lomítko. Typický príklad XML elementu môže byť napríklad:

<teplota>**36**</teplota>

Každý element môže v sebe obsahovať ďalšie takzvané vnorené elementy.

```
<pocasie>
   <vietor>severozapadny</vietor>
   <teplota>36</teplota>
</ pocasie >
```

Toľko stačilo ako teoretický úvod a poďme vytvoriť prvú webovú službu vo vývojovom prostredí Web Matrix. V menu pre vytvorenie súboru v zložke WebServices môžeme vytvárať štyri typy súborov

Typy súborov v zložke Web Services

Najjednoduchšia webová služba pre .NET Framework

Pre vytvorenie webovej služby môžeme použiť aj voľne šíriteľné vývojové prostredie WebMatrix. Ak v tomto vývojovom prostredí vytvoríme novú webovú službu, zistíme, že celá zdrojovka ukážky webovej služby, ktorej úlohou je sčítať dva čísla má len 10 riadkov.

Vytvorenie projektu typu webová služba vo vývojovom prostredí WebMatrix

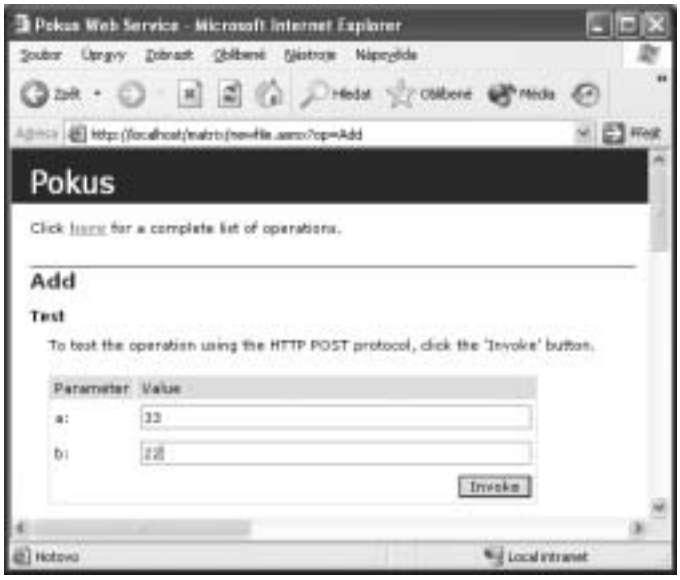
Príklad v programovacom jazyku Visual Basic .NET

```
<%@ WebService language="VB" class="Pokus" %>

Imports System
Imports System.Web.Services
Imports System.Xml.Serialization

Public Class Pokus

    <WebMethod> Public Function Add(a As Integer, b As Integer) As Integer
        Return a + b
    End Function

End Class
```

Príklad v programovacom jazyku C#

```
<%@ WebService language="C#" class="Pokus" %>
using System;
using System.Web.Services;
using System.Xml.Serialization;

public class Pokus {

    [WebMethod]
    public int Add(int a, int b) {
        return a + b;
    }
}
```

Túto jednoduchú webovú službu, ktorú vytvoril sprievodca vytvorením aplikácie môžeme otestovať pomocou prehliadača webových stránok, v našom prípade na adrese http://localhost/matrix/newfile.asmx



Testovanie webovej služby

Po zatlačení tlačidla Invoke nám webová služba vráti výsledok, číslo 8, samozrejme vo formáte XML

```
<?xml version="1.0" encoding="utf-8" ?>
  <int xmlns="http://tempuri.org/">55</int>
```

Príklad webovej služby - pôžičkový kalkulátor

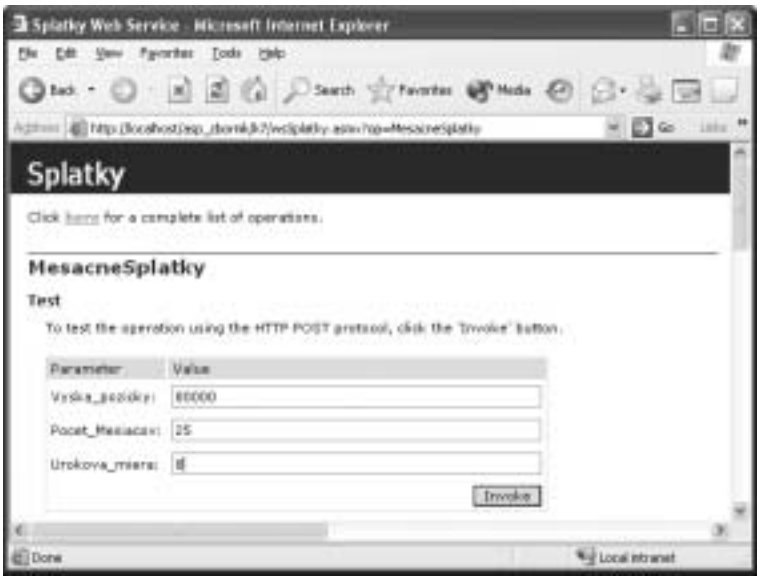
V tomto príklade vytvoríme jednoduchú webovú službu **wsSplatky**, ktorá spočíta výšku mesačnej splátky pri zadaní výšky pôžičky, úrokovej miery a počtu mesiacov na ktoré je pôžička poskytnutá. Založíme nový projekt typu WebServices, nazveme ho wsSplatky.asmx a umiestnime ho do adresára napríklad *C:\Inetpub\wwwroot\ASP_zbornik\k7*. Pre jednoduchosť použijeme ako programovací jazyk Visual Basic .NET nakoľko má implementovanú funkciu Pmt pre výpočet úroku.

Vývojové prostredie nám vygeneruje kostru aplikácie, kde doplníme kód tela webovej službu (hrubým fontom). Aby sme mohli využívať funkcie Pmt pre výpočet úroku a FormatCurrency pre formátovanie „obeživa“ musíme importovať namespace **`Microsoft.VisualBasic.Strings`** a **`Microsoft.VisualBasic.Strings`**.

Kompletný kód webovej služby bude:

```
<%@ WebService language="VB" class="Splatky" %>
Imports System
Imports System.Web.Services
Imports System.Xml.Serialization
Imports Microsoft.VisualBasic.Strings
Imports Microsoft.VisualBasic.Financial


Public Class Splatky
<WebMethod()>Public Function MesacneSplatky (ByVal Vyska_pozicky As    Double, ByVal
Pocet_Mesiacov As Integer, ByVal Urokova_miera As Double) As String
        Dim dblSplatka As Double
        dblSplatka = -Pmt((Urokova_miera / 1200), Pocet_Mesiacov, Vyska_pozicky)
        Return FormatCurrency(dblSplatka)
  End Function
End Class
```

Webovú službu môžeme jednoducho otestovať pomocou prehliadača webových stránok, v našom prípade na adrese http://localhost/asp_zbornik/k7/wsSplatky.asmx



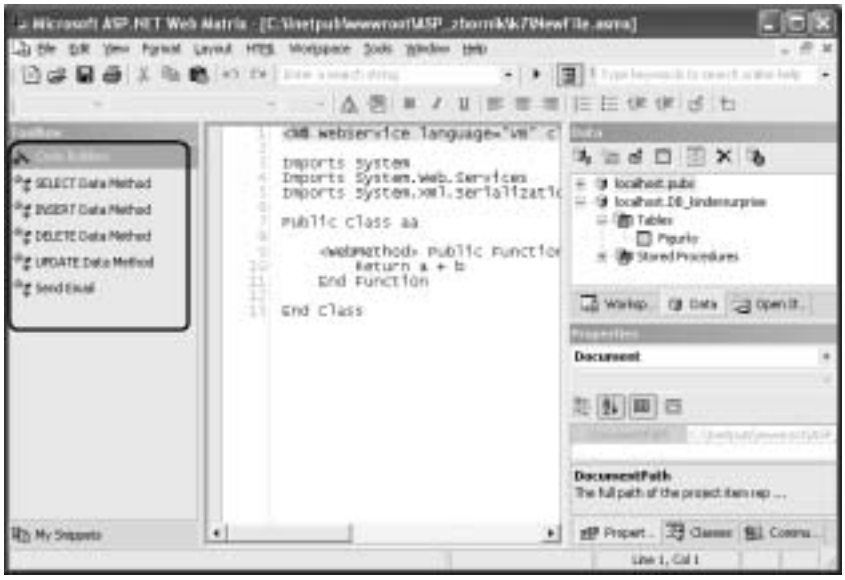
Testovanie webovej služby Pôžičkový kalkulátor

Po zatlačení tlačidla Invoke nám webová služba vráti výsledok, samozrejme vo formáte XML

```
<?xml version="1.0" encoding="utf-8" ?>
  <string xmlns="http://tempuri.org/">3 484,70 Kč</string>
```

Webová služba využívajúca databázu

Pre vytvorenie webovej služby, ktorá bude poskytované údaje čerpať z databázy by sme do tela webovej služby museli napísať niekoľko riadkov pre pripojenie sa k databáze a výber požadovaných údajov. Vývojové prostredie Web Matrix nám však podá pomocnú ruku aj pri tvorbe aplikácie tohto typu. Všimnime si pri založení nového súboru typu webová služba zostane v Toolboxe záložka CodeBuilders. Táto obsahuje šablóny kódov pre najpoužívanejšie operácie s údajmi v databáze, teda SELECT, INSERT, DELETE a UPDATE a šablónu kódu pre posielanie emailu.

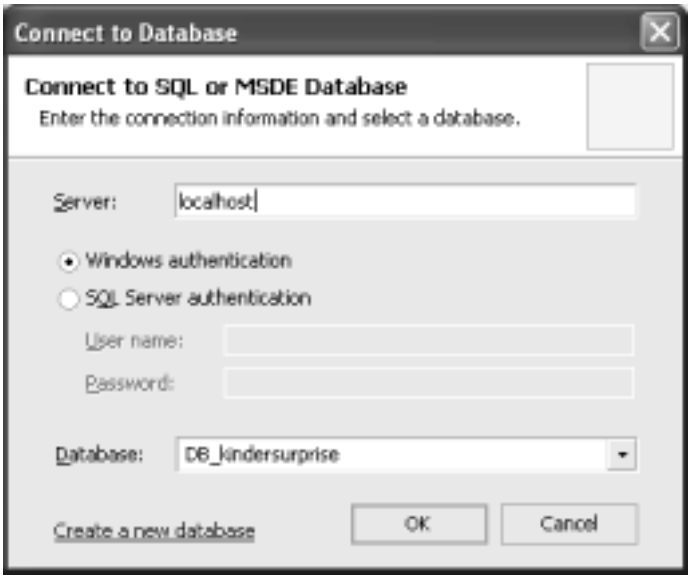


Toolbox - záložka Code Builders

Použitie týchto šablón kódu je jednoduché a intuitívne. V našom prípade použijeme databázu DB_kindersurprise z piatej kapitoly. Cieľom je vytvoriť webovú službu, ktorej ako parameter zadáme názov figúrky a služba nám vráti informácie o danej entite, ktorú má zberateľ, alebo zberateľský klub k dispozícii na predaj alebo na výmenu. Začneme podobne ako v predchádzajúcich príkladoch, vytvoríme nový súbor webovej služby.
Mohli by sme prepísať kostru funkcie, ktorú premenujeme na `DajInfoFigurky`. Táto funkcia dostane ako parameter názov figúrky, teda reťazec a vráti celý záznam o figúrke, teda DataSet. No ako uvidíme neskôr, postačí vymazať pôvodnú funkciu a využiť možnosti, ktoré nám poskytuje vývojové prostredie.

```
<%@ WebService language="VB" class="Kinderka" %>

Imports System
Imports System.Web.Services
Imports System.Xml.Serialization

Public Class Kinderka

    <WebMethod>

End Class
```

Do tela triedy, za kód <WebMethod> presunieme ikonu SELECT Data Method zo záložky Code Builders. Všetko potrebné pre vytvorenie kódu pre pripojenie sa k databáze vykoná za nás vývojové prostredie.



Pripojenie sa na databázu

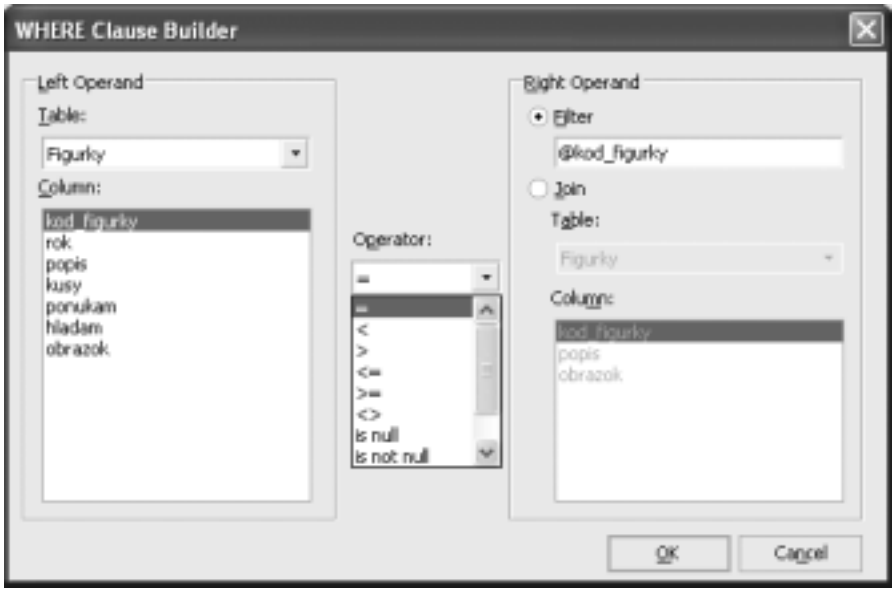
Vytvorením pripojenia sa k správnej databáze generovanie kódu pre výber údajov len začína. Pokračujeme zostavením SQL príkazu SELECT. Aj pri tomto zostavovaní nám pomôže vývojové prostredie návrhovým dialógom. Keďže nás zaujímajú komplexné informácie, zaškrtneme v zozname stĺpcov buď všetky položky alebo zástupný znak hviezdičku.

V spodnej časti návrhového diaógu môžeme sledovať „postup“ návrhu. Po zaškrtnutí poľa kusy bude SQL dotaz v tvare.

```
SELECT [Figurky].* FROM [Figurky]
```

Vytvorenie časti príkazu SELECT - výber stĺpcov

Ešte potrebujem skonštruovať podmienku pre výber záznamu v klauzuli WHERE.

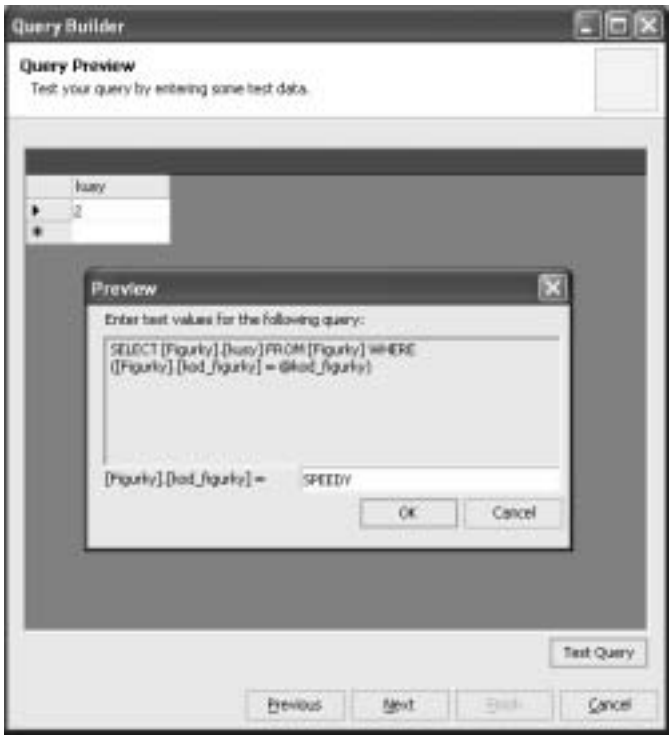


Zostavenie podmienky v klauzule WHERE

Po definovaní podmienky bude SQL dotaz v tvare.

```
SELECT [Figurky].* FROM [Figurky] WHERE ([Figurky].[kod_figurky] = @kod_figurky)
```

V našom prípade budeme testovať, či sa reťazec v poli **`kod_figurky`** rovná nejakej premenne. Zatiaľ môžeme ponechať ponúknutý názov premennej **`@kod_figurky`**, veď ho neskôr môžeme kedykoľvek prepísať. Nakoniec môžeme vytvorený SQL dotaz vrátane podmienky s parametrom otestovať.



Otestovanie zostavenie podmienky

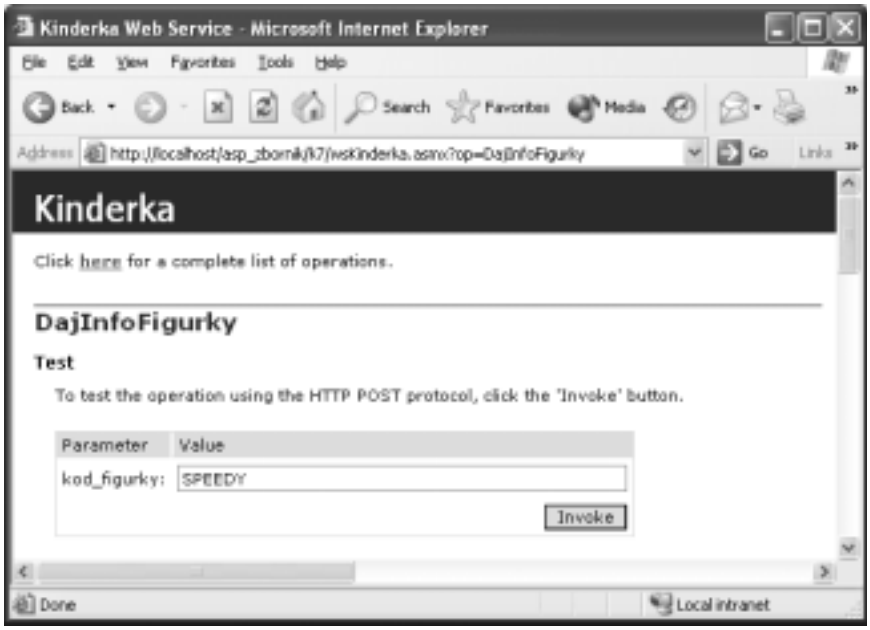
A dostávame sa do finále. Pre výber údajov z databázy môžeme použiť jednak dataset, jednak datareader.

Finálny výber - dataset verzus datareader

Ak sa teraz pozrieme na kód, je to takmer to čo potrebujeme, zhoda je aj v parametroch

```
<%@ WebService language="VB" class="Kinderka" %>

Imports System
Imports System.Web.Services
Imports System.Xml.Serialization

Public Class Kinderka

   <WebMethod>Function DajInfoFigurky(ByVal kod_figurky As String) As System.Data.DataSet
        Dim connectionString As String = "server='localhost'; trusted_connection=true;
Database='DB_kindersurprise'"
        Dim sqlConnection As System.Data.SqlClient.SqlConnection = New
System.Data.SqlClient.SqlConnection(connectionString)

        Dim queryString As String = "SELECT [Figurky].* FROM [Figurky] WHERE
([Figurky].[kod_figurky] = @kod_figurky)"
        Dim sqlCommand As System.Data.SqlClient.SqlCommand = New
System.Data.SqlClient.SqlCommand(queryString, sqlConnection)

        sqlCommand.Parameters.Add("@kod_figurky", System.Data.SqlDbType.VarChar).Value =
kod_figurky

        Dim dataAdapter As System.Data.SqlClient.SqlDataAdapter = New
System.Data.SqlClient.SqlDataAdapter(sqlCommand)
        Dim dataSet As System.Data.DataSet = New System.Data.DataSet
        dataAdapter.Fill(dataSet)

        Return dataSet
    End Function

End Class
```

Webovú službu samozrejme vyskúšame, napríklad pre figúrku SPEEDY.



Vyskúšanie webovej služby

Po zadaní názvu figúrky nám webová služba vráti všetku údaje o nej, samozrejme v XML.

```
<?xml version="1.0" encoding="utf-8" ?>
- <DataSet xmlns="http://tempuri.org/">
- <xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
- <xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:Locale="cs-CZ">
- <xs:complexType>
- <xs:choice maxOccurs="unbounded">
- <xs:element name="Table">
- <xs:complexType>
- <xs:sequence>
  <xs:element name="kod_figurky" type="xs:string" minOccurs="0" />
  <xs:element name="rok" type="xs:short" minOccurs="0" />
  <xs:element name="popis" type="xs:string" minOccurs="0" />
  <xs:element name="kusy" type="xs:short" minOccurs="0" />
  <xs:element name="ponukam" type="xs:boolean" minOccurs="0" />
  <xs:element name="hladam" type="xs:boolean" minOccurs="0" />
  <xs:element name="obrazok" type="xs:string" minOccurs="0" />
  </xs:sequence>
  </xs:complexType>
  </xs:element>
  </xs:choice>
  </xs:complexType>
  </xs:element>
  </xs:schema>
- <diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata" xmlns:diffgr="urn:schemas-microsoft-
com:xml-diffgram-v1">
- <NewDataSet xmlns="">
- <Table diffgr:id="Table1" msdata:rowOrder="0">
  <kod_figurky>SPEEDY</kod_figurky>
  <rok>1999</rok>
  <popis>100% stav</popis>
  <kusy>2</kusy>
```

```
<ponukam>true</ponukam>
<hladam>false</hladam>
<obrazok>zajac1.jpg</obrazok>
</Table>
</NewDataSet>
</diffgr:diffgram>
</DataSet>
```

Metódu webovej služby pre výber údajov z databázy máme otestovanú. S pre človeka nie práve najergonomickejším formátom údajom si nemusíme robiť starosti. Webové služby nebudú priamo používať používatelia, ale budú k údajom tymíto službami poskytovanými pristupovať pomocou rôznych aplikačných rozhraní. A pre ladiace účely sa v XML súbore dokážeme celkom dobre zorientovať.

Informačné zdroje na webe

Informačné zdroje na webe k technológii ASP.NET nájdete na adresách
http://www.aspnetwork.cz/www.asp.net
www.123aspx.com
http://msdn.microsoft.com
http://www.gotdotnet.com
http://www.ibuyspy.com
http://www.asp.net
news://news.microsoft.com
http://www.aspnet.cz

prípadne sa môžete zapojiť do monitorovaných diskusních skupín
http://www.emwac.cz/forums/
news://news.emwac.cz/cz.comp.microsoft.vsnet

Voľné ASP.NET hostingy v Čechách sú:
dotnet.iol.cz
www.aspweb.cz

Zoznam použitej a odporúčanej literatúry

[1] Pope, M. ASP.NET Web Matrix Starter Kit, Microsoft Press 200#
[2] Homer, A. Sussman, D. Inside ASP.NET Web Matrix, Wrox 2002

Vydal: Microsoft, s.r.o., Novodvorská 1010/14B, 140 00 Praha 4, tel.: +420 261 197 111, fax: +420 260 197 100
http://www.microsoft.com/cze, http://www.microsoft.com/slovakia