Vysoká škola báňská – Technická univerzita Ostrava

PROGRAMOVÁNÍ APLIKACÍ PRO INTERNET II

učební text

Marek Babiuch

Ostrava 2007

Recenze: Ing. Pavel Buřil, Prof. RNDr. Alena Lukasová, CSc.

Název: Programování aplikací pro Internet II
Autor: Marek Babiuch
Vydání: první, 2007
Počet stran: 181

Vydavatel a tisk: Ediční středisko VŠB – TUO

Studijní materiály pro studijní obor Automatické řízení a inženýrská informatika fakulty strojní
Jazyková korektura: nebyla provedena.

Určeno pro projekt:
Operační program Rozvoj lidských zdrojů
Název: E-learningové prvky pro podporu výuky odborných a technických předmětů
Číslo: CZ.O4.01.3/3.2.15.2/0326
Realizace: VŠB – Technická univerzita Ostrava
Projekt je spolufinancován z prostředků ESF a státního rozpočtu ČR

ISBN 978-80-248-1504-6

Obsah

POKYNY KE STUDIU

Programování aplikací pro Internet II

Pro předmět Programování aplikací pro Internet II 1.semestru oboru Automatické řízení a inženýrská informatika jste obdrželi studijní balík obsahující:

- integrované skriptum pro distanční studium obsahující i pokyny ke studiu
- CD-ROM s doplňkovými animacemi vybraných částí kapitol
- harmonogram průběhu semestru a rozvrh prezenční části
- rozdělení studentů do skupin k jednotlivým tutorům a kontakty na tutory
- kontakt na studijní oddělení

Prerekvizity

Pro studium tohoto předmětu se předpokládají znalosti z předmětu Programování aplikací pro Internet I. Důležité je, aby student, který chce absolvovat tento předmět měl základy tvorby webových stránek a měl alespoň nějaké zkušenosti se skriptovacími či programovacími jazyky. Znalost HTML, CSS a souvisejících pojmů s tvorbou webových stránek je samozřejmostí.

Cílem předmětu

je seznámení se s technologií ASP.NET a tvorbou webových aplikací. Učební text je rozdělen do dvanácti vybraných kapitol tak, aby na sebe logicky navazovaly a poskytly studentům stavební kámen pro tvorbu pokročilých webových aplikací. Učební text a celý předmět svým rozsahem nemůže konkurovat mnohasetstránkovým knihám, není to ani jeho cílem, naopak má za pomoci vhodně zvolených řešených příkladů podnítit zájem o tuto velmi mladou avšak robustní a perspektivní technologii.

Pro koho je předmět určen

Modul je zařazen do magisterského studia oboru Automatické řízení a inženýrská informatika studijního programu .....

Předmět Programování aplikací pro Internet II je volitelným předmětem a jeho výuka v denní formě probíhá výhradně formou cvičení. Proto je učební text koncipován tak, aby studentům kombinované formy co nejvíce přiblížil reálný semestr studenta v prezenční formě.

Skriptum se dělí na části a kapitoly, které odpovídají logickému dělení studované látky, ale nejsou stejně obsáhlé. Předpokládaná doba ke studiu kapitoly se může lišit u studentů, kteří již mají bohaté zkušenosti s tvorbou webových aplikací a u studentů, kteří jsou pouze mírně pokročilými.

V průběhu semestru bude kladen důraz na osobní přístup pedagoga ke studentům, uvedené kapitoly budou prakticky procvičovány a získané zkušenosti budou využity k tvorbě semestrálního projektu s jehož zadáním budou studenti seznámeni během semestru.

V každé kapitole je procvičována řada ukázkových příkladů, které budou ještě podrobněji vytvářeny v přiložených animacích. Každá kapitola obsahuje důležité otázky, jejichž odpovědi jsou v klíči k řešení a několik samostatných příkladů vycházejících z probíraných témat.

Při studiu každé kapitoly doporučujeme následující postup:

Čas ke studiu: x hodin

Na úvod kapitoly je uveden **čas** potřebný k prostudování látky. Čas je orientační a může vám sloužit jako hrubé vodítko pro rozvržení studia celého předmětu či kapitoly. Někomu se čas může zdát příliš dlouhý, někomu naopak. Jsou studenti, kteří se s touto problematikou ještě nikdy nesetkali a naopak takoví, kteří již v tomto oboru mají bohaté zkušenosti.

Cíl: Po prostudování tohoto odstavce budete umět

- popsat ...
- definovat ...
- vyřešit ...

Ihned potom jsou uvedeny cíle, kterých máte dosáhnout po prostudování této kapitoly – konkrétní dovednosti, znalosti.

Výklad

Následuje vlastní výklad studované látky, zavedení nových pojmů, jejich vysvětlení, vše doprovázeno obrázky, tabulkami, řešenými příklady, odkazy na animace.

Řešený příklad

Zadání a řešení praktického příkladu jako součást výukového textu.

Korespondenční úkol

Zadání domácí úlohy nebo souvisejícího úkolu v rámci výkladu.

Pojmy k zapamatování

Touto ikonkou je upozorněno na pojem, který bychom si měli pamatovat. Neznamená to ale, že ostatní pojmy výkladu si pamatovat nemusíme.

Další zdroje

Seznam další literatury, www odkazů apod. pro zájemce o dobrovolné rozšíření znalostí popisované problematiky. Budou uvedeny na konci učebního textu.

CD-ROM

Informace o doplňujících animacích, které si může student vyvolat z CD-ROMu připojeného k tomuto materiálu.

Shrnutí kapitoly

Na závěr kapitoly jsou zopakovány hlavní pojmy, které si v ní máte osvojit. Pokud některému z nich ještě nerozumíte, vraťte se k nim ještě jednou.

Kontrolní otázka

Na konci každé kapitoly bude zveřejněn seznam otázek, které se mohou v plném nebo modifikovaném znění objevit u zkoušky z předmětu Programování aplikací pro Internet II. Tyto otázky jsou také podkladem ke státnicovým otázkám magisterského oboru Automatické řízení a inženýrská informatika, konkrétně státnicového předmětu Informační technologie.

Úkol k řešení

Na konci kapitol, které kromě teoretických otázek umožňují i praktické procvičení probraného učiva najdete úkol k řešení. Rovněž s tímto úkolem se můžete v obdobném znění setkat u zkoušky či zápočtového testu.

Klíč k řešení

Odpovědi z kontrolních otázek výše jsou uvedeny v závěru učebnice v Klíči k řešení. Používejte je až po vlastním vyřešení úloh, jen tak si samokontrolou ověříte, že jste obsah kapitoly skutečně úplně zvládli.

1. Úvod do ASP.NET

Čas ke studiu: 1,5 hodiny

Cíl Po prostudování tohoto odstavce budete umět

- definovat terminologii v oblasti chodu ASP.NET stránek
- popsat schématicky architekturu .NET Framework
- popsat architekturu ASP.NET
- vysvětlit princip kompilace ASP.NET stránek
- srovnat ASP.NET technologii s PHP

Výklad

V první kapitole se budeme zabývat architekturou .NET Frameworku, základy technologie ASP.NET, popíšeme hlavní rozdíly mezi ASP (Active Server Pages) a ASP.NET. Ukážeme, kdy je užitečné a přínosné nahradit ASP aplikaci technologií ASP.NET a co programovací model .NET přináší vývojářům. Srovnáme také technologii ASP.NET s několika nejvýznamnějšími konkurenčními technologiemi pro vývoj webových aplikací.

1.1 Microsoft .NET Framework

Microsoft .NET Framework je platforma pro vytváření a provozování aplikací zahrnující řadu jazyků (C#, C++, Visual Basic – VB.NET, Pearl a další). Tyto jazyky jsou dále překládány do mezi-jazyka *MSIL* a při spuštění jsou technologií JIT (Just In Time) kompilovány do strojového nativního kódu daného procesoru. Díky tomu mohou různé programovací jazyky sdílet stejné knihovny, a to dokonce na různých platformách.
Microsoft .NET Framework se skládá z několika komponent a mezi ty nejzákladnější patří společný jazykový běhový modul (*Common Language Runtime* – CLR) a knihovna tříd rámce (*Framework Class Library* – FLC). *Microsoft .NET Framework* podporuje mnoho programovacích modelů. Kromě vytváření webových služeb XML můžete psát konzolové aplikace, aplikace s grafickým uživatelským rozhraním GUI (v .NET nazvané „formuláře Windows" – WinForms), webové aplikace („webové formuláře" – WebForms) a nebo dokonce i služby Windows, častěji označované jako služby NT.

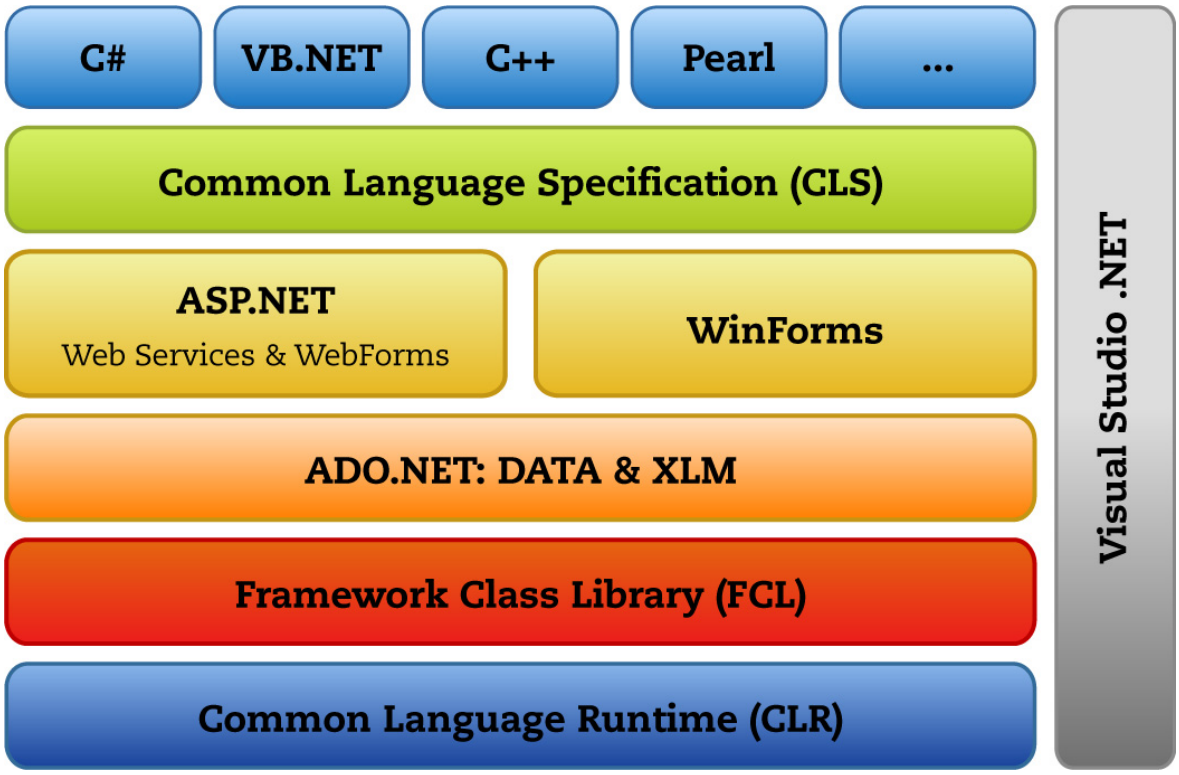
Pojem k zapamatování: .NET Framework

Systém .NET Framework je souhrn objektů a šablon pro vytváření aplikací .NET, jak desktopových, tak webových. Neustále se rozšiřuje a ve svých nových verzích přidává celou řadu nových prvků, podle požadavků současného rozvoje informačních technologií.

- **Common Language Runtime (CLR)**

Společný jazykový běhový modul CLR se nachází nad operačním systémem a poskytuje virtuální prostředí pro řízené aplikace. Běhový modul CLR odvozuje služby operačního systému a slouží jako vykonávací jádro pro aplikace.

CLR poskytuje mnoho služeb, včetně zavedení a spuštění kódu, správu paměti, zpracování výjimek, konverzi mezikódu MSIL (*Microsoft Intermediate Language*) na nativní kód a mnoho dalších funkcí. Kdyby byl .NET Framework živým organismem, CLR bychom si tedy mohli představit jako srdce a duši celého těla.



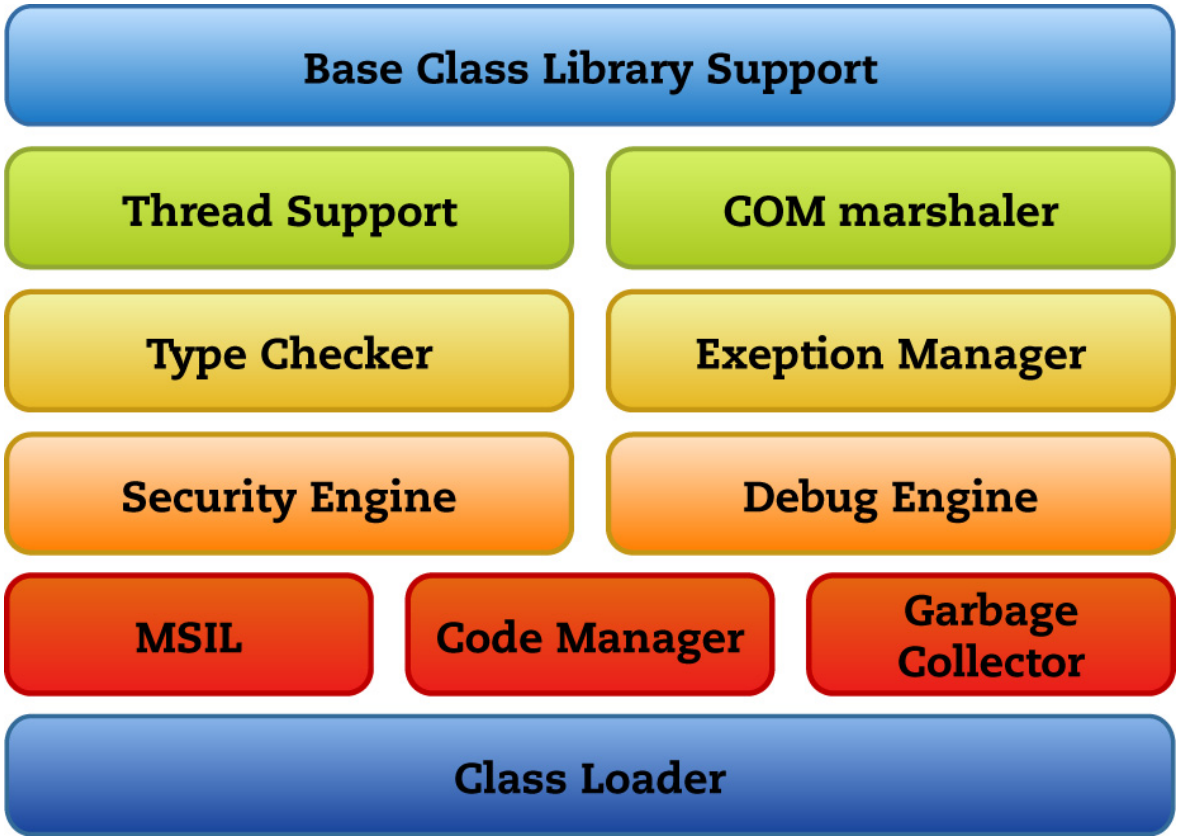
Obr. 1.1 Schéma architektury Microsoft .NET

- **Framework Class Library (FCL)**

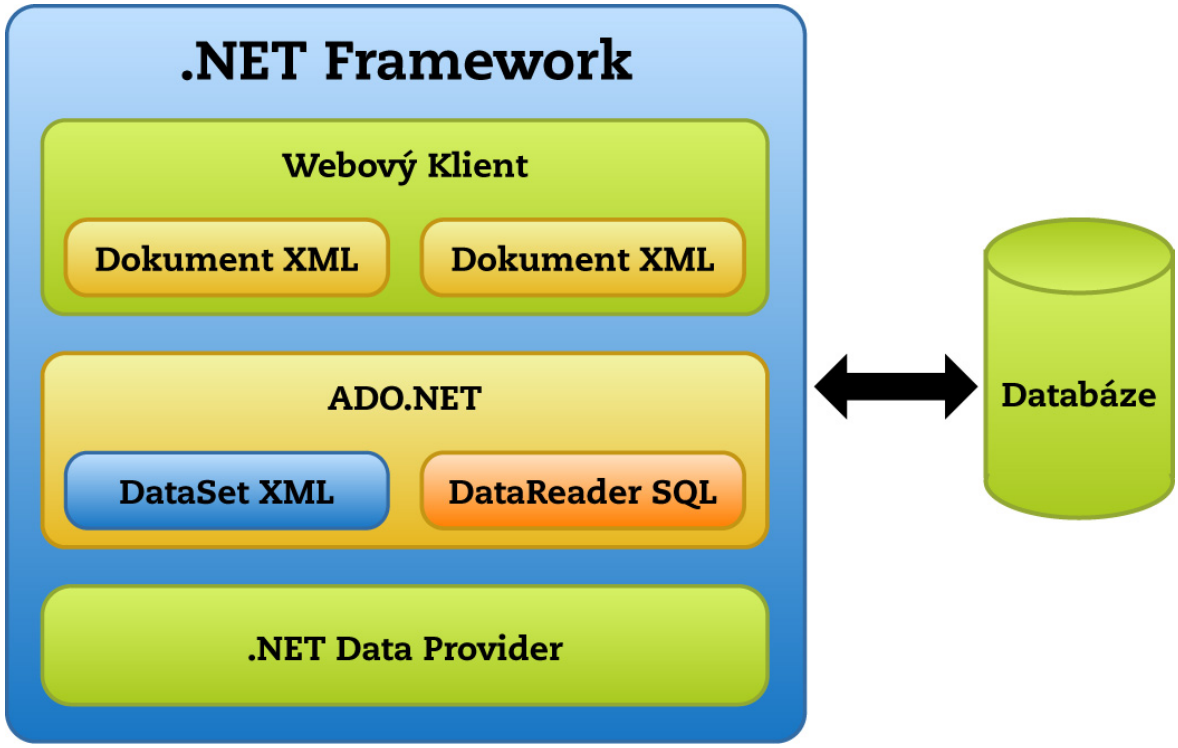
Knihovna tříd FCL (v některých zdrojích nazvaná jako *Base Class Library* – BLC) poskytuje rozsáhlou sbírku tříd, logicky setříděných do hierarchických jmenných prostorů, které umožňují přístup k interním vlastnostem operačního systému.
Při psaní aplikace fungující nad rámcem .NET se nepoužívá rozhraní aplikací API Windows (rozhraní pro programování aplikací), Microsoft COM a další nástroje, ale místo nich se pracuje s knihovnou tříd rámce FCL. V Microsoft .NET Frameworku je možné také zavolat funkci API Windows nebo objekt COM, ale není to žádoucí, protože pak je zapotřebí přechod z řízeného kódu (kódu běžícího pod běhovým modulem CLR) na neřízený kód (nativní strojový kód, který běží bez CLR). Takové přechody omezují výrazně výkonnost.
FCL poskytuje rozsáhlou množinu tříd, logicky setříděných do hierarchických jmenných prostorů a můžeme si jej představit jako velmi rozsáhlý strom funkcí, které umožňují přístup k vlastnostem operačního systému.

- **ADO.NET**

ADO.NET je model Microsoftu pro práci s daty v .NET Frameworku. ADO.NET představuje databázové rozhraní pro aplikace, poskytované jako sada tříd .NET Frameworku. Jedná se o novou generaci technologie přístupu k datům ActiveX Data Object (ADO), obsahující mnoho vylepšení. Na rozdíl od ADO a OLE DB, kteří jsou jejími bezprostředními předchůdci, byla platforma ADO.NET vyvíjena především pro práci na webu. Výborně také podporuje XML a umožňuje tak používat jak relační data, tak i data v podobě XML.



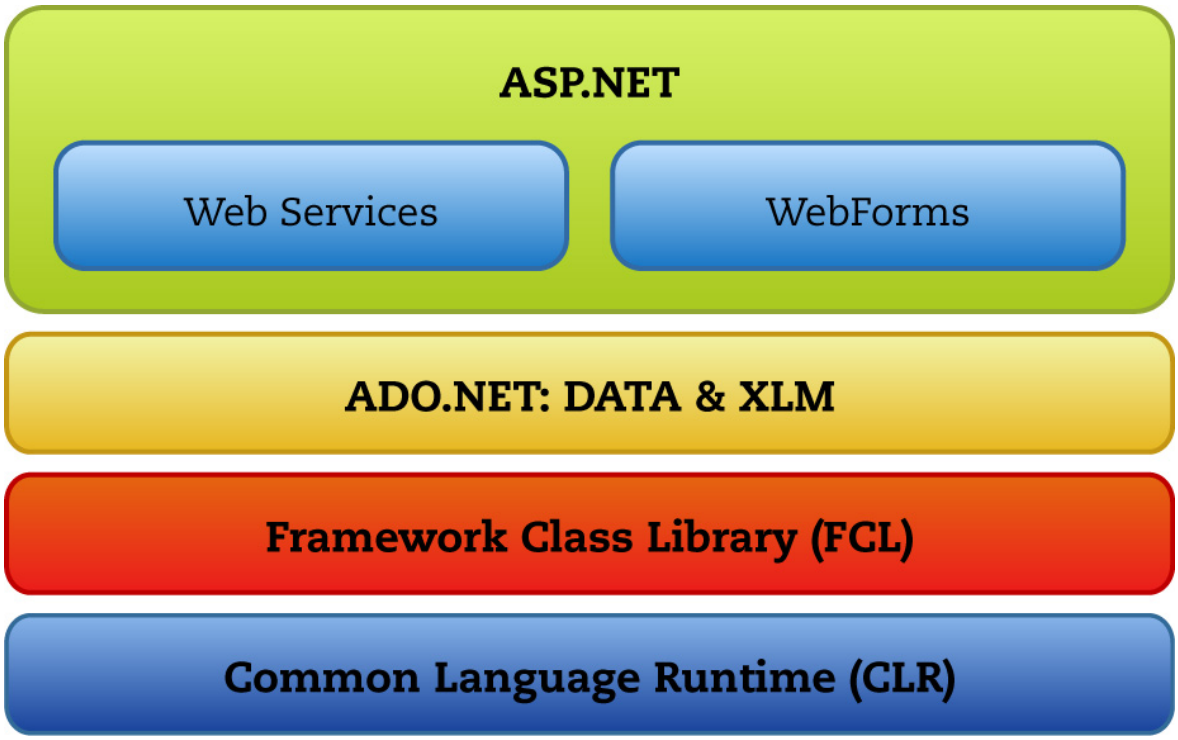
Obr. 1.2 Schéma Common Language Runtime



Obr. 1.3 Schéma hierarchie ADO.NET

1.2 ASP.NET

Název ASP.NET pochází z názvu aktivních serverových stránek ASP (*Active Server Pages*), které vznikly v 90. letech. ASP.NET není jednoduše nová generace ASP, ale kompletně přepsaná a přeorganizovaná technologie pro tvorbu webových aplikací.
Protože je technologie ASP.NET odvozena v přímé odezvě na problémy, které měli vývojáři s klasickými ASP, některé její části vypadají na první pohled velice podobně. Koncepce webových formulářů (WebForms), webových služeb nebo serverových ovládacích prvků (Server Controls) dávají ASP.NET sílu vytvořit reálnou aplikaci, která běží na tenkém klientovi, mnohem výkonnější a v mnohem kratším čase.



Obr. 1.4 Schéma technologie ASP.NET

- **Windows Forms**

Tento programový model a sada ovládacích prvků nabízí robustní architekturu pro vývoj aplikací pro Windows.

- **Common Language Specification (CLS)**

Specifikace, která je zodpovědná za zpřístupnění většiny z výše zmíněných technologií všem jazykům, které platforma .NET podporuje. CLS sama o sobě není technologií a neexistuje pro ni žádný zdrojový kód. Definuje sadu pravidel, která realizují vzájemnou spolupráci mezi kompilátory jazyků a knihovnami.

- **Visual Studio .NET**

Visual Studio .NET je výkonný nástroj, který umožňuje využít vlastnosti .NET Frameworku při vývoji konkrétních aplikací – jak webových, tak i klasických Windows aplikací používající GUI.

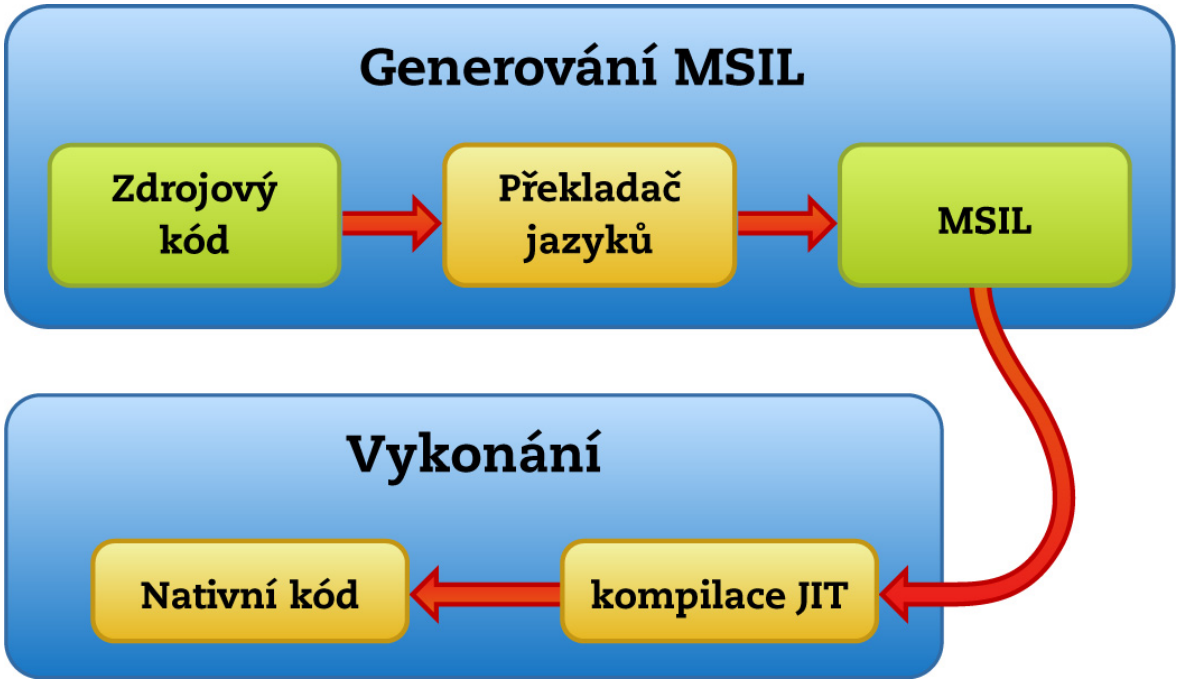
- **Programovací jazyky**

Programovací jazyk v prostředí .NET Frameworku je jen syntaktický prostředek pro vytváření MSIL kódu. S několika málo výjimkami můžete všeho dosažitelného v jednom jazyku docílit i ve všech jiných jazycích. Navíc, bez ohledu na jazyk použitý k jejich vytvoření používají všechny aplikace

stejné API a to z knihovny tříd .NET Frameworku. Mezi podporované jazyky v době psaní této práce patří C#, VB.NET, Managed C++ .NET, J#, Jscript .NET a mnoho dalších. Kompilátory jiných firem rozšiřují možnosti ještě více a přidávají mnoho dalších, někdy i poměrně „exotických“ jazyků. Pro ukázku uvedu např. tyto: Cobol, Pearl, Python a další.

1.3 Průběh kompilace

Jakmile je otevřen vykonatelný soubor např. *.aspx, překladač jazyka .NET vygeneruje řízený (někdy také nazvaný jako spravovaný – managed) kód, který se skládá z instrukcí zapsaných v kódu MSIL. Tento kód je někdy také označován jako společný zprostředkovací jazyk CIL (Common Language Interface). Instrukce MSIL se na požádání zkompilují metodou JIT (Just In Time) do nativního strojového kódu za běhu. Kompilace JIT funguje tak, že kód, který se nikdy nezavolá, se ani nezkompiluje. Ve většině případů se určitá metoda zkompiluje technikou JIT jen jednou — při svém prvním volání — a následně se uloží do paměti, aby ji příště bylo možné vykonat bez zpoždění. Kompilace JIT výrazně snižuje výkonnost při prvním spuštění, ale její negativní efekty jsou minimalizovány tím, že daná metoda se kompiluje jen jednou během celého života aplikace.



Obr. 1.5 Zjednodušený průběh kompilace vykonatelného souboru do nativního kódu

Pojem k zapamatování: Kompilace ASP.NET stránek

ASP.NET stránky se neinterpretují jako skriptovací jazyky, nýbrž se kompilují. Kompilace probíhá ve dvou fázích. Nejprve se kód napsaný v libovolném programovacím jazyce zkompiluje do jazyka MSIL. Druhá fáze kompilace těsně před spuštěním stránky JIT kompiluje kód do nízkoúrovňového strojového jazyka.

1.4 Srovnání s konkurenční technologií

Nejvíce konkurenční technologií je považován interpretovaný jazyk PHP.

PHP

Plusy:

- PHP je vhodné pro rychlé jednodušší projekty.
- Není vázáno na platformu.
- Cena – tato technologie je zcela zdarma.
- Nízké nároky na hardware.

Mínusy:

- Není typově bezpečný.
- Problém s case-sensitiv. V názvech proměnných je, v názvech funkcí ne.
- Objekty pro objektově orientované programování jsou dolepeny ke struktuře jazyka.
- Není třeba definovat proměnné – největší nevýhoda PHP.
- Vývojové prostředí. Neexistuje tak kvalitní vývojový nástroj jako Visual Studio.

ASP.NET

Plusy:

- S objektovým návrhem je počítáno již od počátku vzniku platformy.
- Jazyková nezávislost.
- Vhodné řešení pro robustní projekty.
- Vynikající vývojový nástroj Visual Studio .NET.
- Rychlost. ASP.NET je díky kompilovanému jazyku výrazně rychlejší.

Mínusy:

- Vázáno ostře na Windows/IIS. Existuje projekt mono, který se snaží migrovat .NET Framework na Unixové systémy, ale zatím má mnoho dětských nemocí.
- Cena. Nutný serverový OS od Microsoftu s IIS. Samotný .NET Framework je zdarma a licenční politika MS uvolnila dobře použitelné, ale také omezené, vývojové prostředí Visual Studio 2005 Express a SQL Server 2005 Express zdarma.

Pojem k zapamatování: Jedinečnost ASP.NET stránek

Vývoj ASP.NET stránek se od ostatních technologií odlišuje. V mnoha odlišnostech poskytuje právě výhody tvorby web aplikací pomocí ASP.NET

- ASP:NET je svázáno s .NET frameworkem, stránky se neinterpretují ale kompilují.
- ASP.NET stránky jsou objektově orientované.
- ASP.NET můžeme psát ve svém oblíbeném jazyce.
- ASP.NET poskytuje automatickou správu paměti, typovou bezpečnost, zpracování vyjímek, ladění web aplikací, pokročilé prvky zabezpečení a další.

Shrnutí kapitoly

Microsoft .NET Framework je platforma pro vytváření a provozování aplikací zahrnující řadu jazyků (C#, C++, Visual Basic – VB.NET, Pearl a další). *Microsoft .NET Framework* se skládá z několika komponent a mezi ty nejzákladnější patří společný jazykový běhový modul (*Common Language Runtime – CLR*) a knihovna tříd rámce (*Framework Class Library – FLC*). Společný jazykový běhový modul CLR se nachází nad operačním systémem a poskytuje virtuální prostředí pro řízené aplikace. Běhový modul CLR

odvozuje služby operačního systému a slouží jako vykonávací jádro pro aplikace.
Knihovna tříd FCL (v některých zdrojích nazvaná jako Base Class Library – BLC) poskytuje rozsáhlou sbírku tříd, logicky setříděných do hierarchických jmenných prostorů, které umožňují přístup k interním vlastnostem operačního systému.

ADO.NET je model Microsoftu pro práci s daty v .NET Frameworku. ADO.NET představuje databázové rozhraní pro aplikace, poskytované jako sada tříd .NET Frameworku. Jedná se o novou generaci technologie přístupu k datům ActiveX Data Object (ADO), obsahující mnoho vylepšení.

Název ASP.NET pochází z názvu aktivních serverových stránek ASP (*Active Server Pages*), které vznikly v 90. letech. ASP.NET není jednoduše nová generace ASP, ale kompletně přepsaná a přeorganizovaná technologie pro tvorbu webových aplikací.
Protože je technologie ASP.NET odvozena v přímé odezvě na problémy, které měli vývojáři s klasickými ASP, některé její části vypadají na první pohled velice podobně. Koncepce webových formulářů (WebForms), webových služeb nebo serverových ovládacích prvků (Server Controls) dávají ASP.NET sílu vytvořit reálnou aplikaci, která běží na tenkém klientovi, mnohem výkonnější a v mnohem kratším čase.

Jakmile je otevřen vykonatelný soubor např. *.aspx, překladač jazyka .NET vygeneruje řízený (někdy také nazvaný jako spravovaný – managed) kód, který se skládá z instrukcí zapsaných v kódu MSIL. Tento kód je někdy také označován jako společný zprostředkovací jazyk CIL (Common Language Interface). Instrukce MSIL se na požádání zkompilují metodou JIT (Just In Time) do nativního strojového kódu za běhu. Kompilace JIT funguje tak, že kód, který se nikdy nezavolá, se ani nezkompiluje. Ve většině případů se určitá metoda zkompiluje technikou JIT jen jednou — při svém prvním volání — a následně se uloží do paměti, aby ji příště bylo možné vykonat bez zpoždění. Kompilace JIT výrazně snižuje výkonnost při prvním spuštění, ale její negativní efekty jsou minimalizovány tím, že daná metoda se kompiluje jen jednou během celého života aplikace.

Kontrolní otázka 1.1

Co je to .NET Framework, z čeho se skládá a k čemu slouží?

Kontrolní otázka 1.2

Popište průběh kompilace aspx stránek.

Kontrolní otázka 1.3

Popište základní rysy tvorby stránek v ASP.NET.

Kontrolní otázka 1.4

Co je to ADO.NET?

2. PRÁCE V PROSTŘEDÍ MS VISUAL STUDIO 2005

Čas ke studiu: 3 hodiny

Cíl Po prostudování této kapitoly budete umět

- orientovat se ve vývojovém prostředí MS Visual Studia 2005
- popsat funkčnost jednotlivých součástí prostředí
- definovat výhody, proč k tvorbě ASP.NET stránek využít právě VS 2005
- vytvořit svou první ASP.NET aplikaci
- popsat vlastnosti MS Visual Studia 2005

Výklad

Pro napsání nějaké jednoduché webové stránky nepotřebujeme složité vývojové prostředí. Mnohdy nám stačí obyčejný textový editor, jednotlivá dokonalejší prostředí nám pouze usnadňují práci a poskytují komfort pro psaní kódu. Ano, tak tomu bylo v minulosti, kdy na napsání zdrojového kódu dokázal zkušený tvůrce webu využít třeba jen *notepad*. To samozřejmě za určitých předpokladů lze stále, ale nevyužít pro tvorbu současných web aplikací vývojový nástroj by byla cesta velice obtížná a rozhodně zdlouhavá. Tato kapitola nám poskytne pohled na vývojové prostředí MS Visual Studio 2005 a ukáže právě diskutované výhody komfortního prostředí pro tvorbu webu.

2.1 Základní vlastnosti vývojového prostředí

Tato kapitola by se také mohla nadneseně jmenovat deset důvodů proč používat Visual Studio 2005. Můžete samozřejmě také využít jiné prostředí, Microsoft pro tvorbu web aplikací nabízí také Visual Web Developer 2005, který nabízí obdobný komfort, je ale redukován o některé funkcionality Visual Studia. Visual Studio prošlo také svými vývojovými verzemi. Předchozí verze jsou však omezeny na verze ASP.NET 1.1, takže je nebudeme využívat, i když v případech, kdy nevyužijeme novinek z ASP.NET 2.0, bychom si s nimi vystačili. Nyní popíšeme důvody použití Visual Studia 2005, některé z nich jsou obecné vlastnosti pokročilých nástrojů pro tvorbu webu (např. WYSIWYG model – co vidíte, to dostanete a další), avšak některé jsou charakteristickou vlastností právě pro Visual Studio 2005.

Pojem k zapamatování: Komfortní vlastnosti Visual Studia 2005

Visual studio 2005 je robustní nástroj pro tvorbu konzolových ale také webových aplikací. Umožňuje nám vytvořené stránky navrhovat v design režimu, ladit zdrojový kód, podporují užitečné nástroje jako IntelliSense, vestavěný web server a mnoho dalšího. Proto budeme ASP.NET stránky vytvářet v tomto vývojovém prostředí.

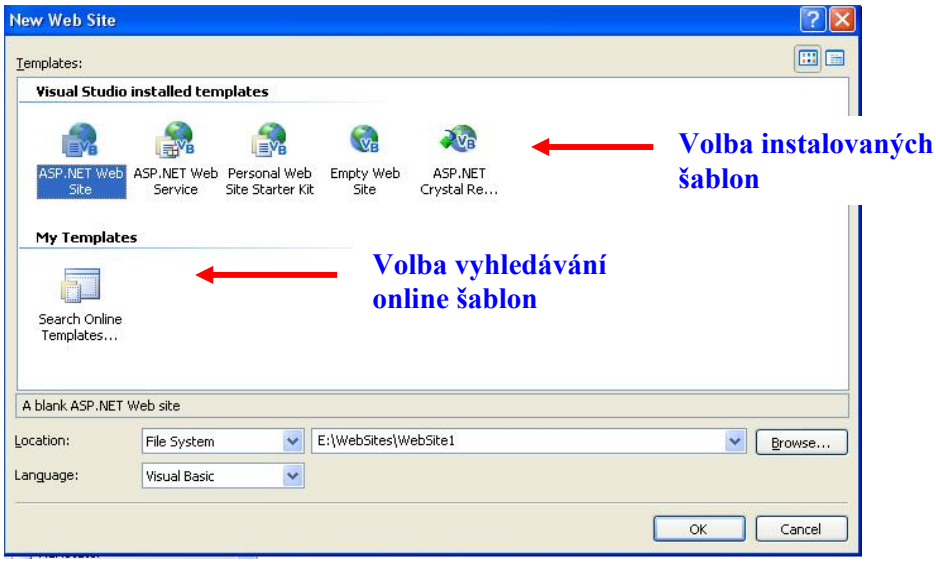
1. **WYSIVYG model**

Součástí studia je tzv. design mód neboli záložka pro tvorbu návrhu. Zde můžete jednotlivé součásti stránky navrhnout bez nutnosti psaní zdrojového kódu, ten se vygeneruje automaticky. Definice

wysiwyg nástrojů je vlastnost, která zaručuje, že výsledek se skutečně zobrazí tak, jak vypadá v době návrhu.

2. Používání šablon

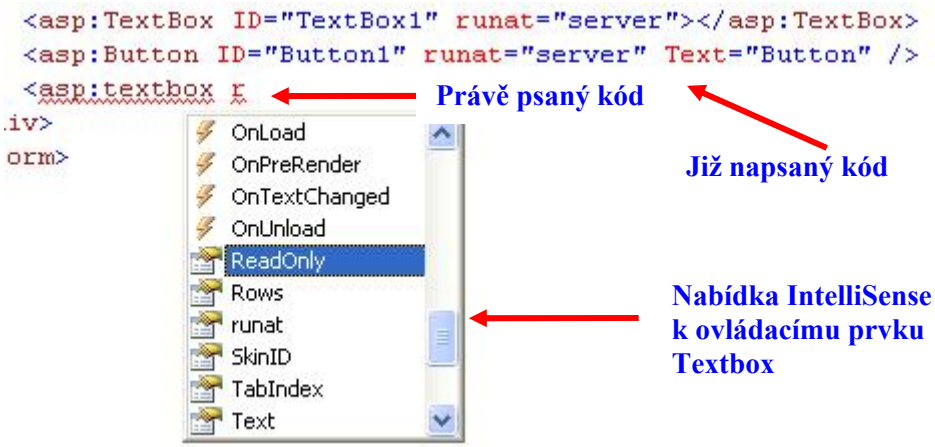
Výběrem šablony zahájíte práci na vašem webovém projektu. V podstatě jde o to, jaký typ aplikace chcete vyvíjet přednostně, ať už ASP stránku či webovou službu. Po zvolení šablony se implicitně vygeneruje příslušná kategorie. Oba typy aplikací se však kompilují a vykonají obdobně a tak můžete během vývoje přidávat do projektu další stránky či služby. Můžete také využít nějakou vyhotovenou šablonu z dřívějška či vyhledat šablonu z online zdrojů. Oblíbenými šablonami jsou také různé s*tarter kity*, ve kterých jsou připraveny šablony šité na míru určitému řešení jako např. webový portál, osobní stránky, diskusní fórum, blog a další.



Obr. 2.1 Volbou šablony založíme nový webový projekt

3. IntelliSense

IntelliSense je výborná pomůcka pro zjednodušení psaní, která mimo jiné také ve svém důsledku minimalizuje počet syntaktických chyb, které uživatel může vytvořit. *IntelliSense* je nástroj, který na základě vámi psaného zdrojového textu nabízí možné použití výběru dalších zdrojových slov. Můžete tedy například vybrat ze seznamu dlouhý název ovládacího prvku, který byste jinak psali ručně. *IntelliSense* také nabízí všechny dostupné metody a vlastnosti při použití konkrétního objektu, třídy či proměnné. Tím je zajištěna zmíněná redukce chyb, neboť při použití *IntelliSense* se vývojáři nemůže stát, že vybere metodu či vlastnost, která pro daný objekt není definována.



Obr. 2.2 Nástroj IntelliSense při použití prvku textbox

4. Méně psaní kódu

Pokud se rozhodneme přidat nějaký prvek do ASP stránky, můžeme ho samozřejmě přidat ručně a nebo použít již zmíněnou záložku *Design*. V tomto případě se po vložení prvků generuje zdrojový kód automaticky. Důležité je, že se generuje včetně ID a svých základních atributů. Ty můžeme pak ručně upravit a nebo využít okno *Properties*, ve kterém příslušné vlastnosti nastavíme a opět Visual Studio kód vygeneruje za nás. Samozřejmostí kvalitního editoru je kontrola párových značek, takže při počáteční značce je ihned generována značka koncová.

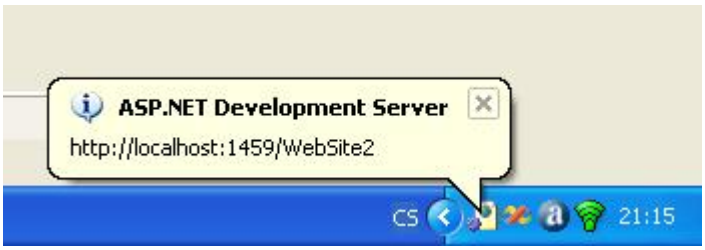
5. Rozbalovací části kódu

Tento nástroj se nazývá *Outlining* a umožňuje větší přehlednost kódu v možnosti takzvaného sbalení a rozbalení textu. Pokud tedy nějakou část kódu nepotřebujeme vidět, můžeme ji jednoduše sbalit kliknutím na znaménko mínus. Skrytá část se opětovně rozbalí po kliknutí na znaménko plus.

Obr. 2.3 Nástroj Outlining pro skrytí a rozbalení zdrojového kódu

6. Vestavěný web server

Předchozí verzi studia jste mohli využívat pro tvorbu ASP.NET stránek ve spolupráci s Internetovou informační službou (IIS). Mnoho administrátorů mi dá za pravdu, že docházelo často k problémům s pořadím instalace jednotlivých produktů, neboť IIS nebyla standardně instalovaná s operačním systémem a když jste ji chtěli doinstalovat až po instalaci Visual Studia, nastávaly problémy. Nyní tyto i další náležitosti jako provoz virtuálního adresáře *Inetpub* a další již odpadají. Visual Studio 2005 obsahuje integrovaný web server a na něm je stránka s vygenerovaným číslem portu spuštěna (viz. obrázek 2.4).

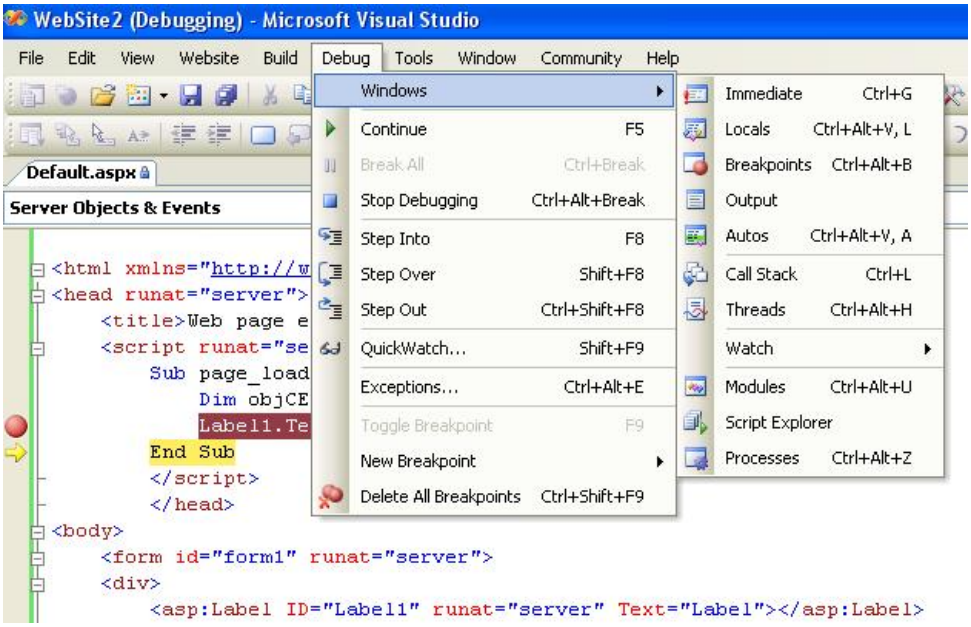


Obr. 2.4 Integrovaný web server umožňuje spouštět stránky bez přítomnosti IIS

7. Možnosti ladění

Visual Studio 2005 je mocný nástroj, v němž se vytvářejí webové ale také desktopové aplikace. K základním vlastnostem vývojového prostředí patří možnosti ladění projektů. To se týká samozřejmě i webových aplikací. Pokud chceme aplikaci ladit, zvolíme z nástrojové lišty tlačítko *Start Debugging*.

Obdobně jako u desktopových aplikací využíváme zarážky – tzv. *breakpointy*. Pokud máme umístěnou zarážku kdekoliv v programu a spustíme ladění, program se zastaví na daném řádku kódu. Na následujícím obrázku 2.5 vidíme menu *Debug* s jeho možnostmi, vidíme rovněž zastavený program na řádku označený červeným puntíkem, jenž symbolizuje vložený *breakpoint*. V menu *Debug* máme mimo jiné nabídku možností *Step Into*, *Step Over*, *Step Out* a *Continue*. Tyto volby mají také přiřazeny horké klávesy a nástrojové ikonky. *Step Into* označuje krokování dovnitř a znamená fakt, že pokud na daném řádku bude volání nějaké funkce či metody, bude krokování pokračovat právě uvnitř této funkce. Pokud zvolíme *Step Over* – krokovat přes, toto volání funkce provedeme celé, neboť je reprezentováno právě jedním řádkem kódu. Odkrokovat ven z volané procedury můžeme příkazem *Step Out*. Pokud nepotřebujeme program krokovat, zvolíme *Continue* – pokračovat a program poběží dále, dokud nenarazí na další *breakpoint*.



Obr. 2.5 Zobrazení menu *Debug*, které umožňuje volby ladění

8. Sledování proměnných

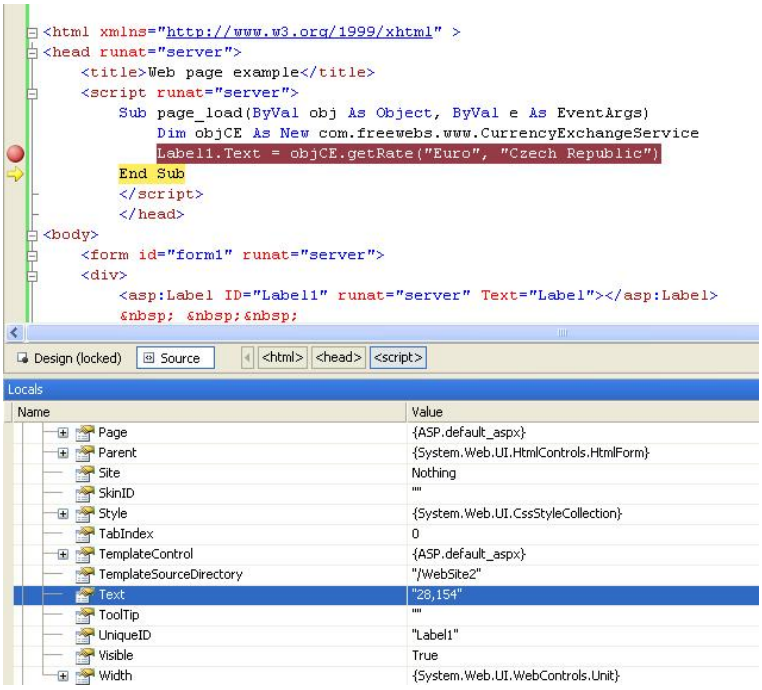
Sledování proměnných úzce souvisí s laděním programu. Na předchozím obrázku vidíme zmíněné okno *Debug* s výběrem *Windows*. Máme zde k dispozici okna *Locals*, *Autos* a *Watch*. Okno Locals zobrazuje proměnné aktuální procedury, okno Autos zobrazuje důležité proměnné programu a okno Watch slouží pro proměnné, které se rozhodneme sledovat a můžeme je tak do tohoto okna přidávat.

Na obrázku 2.6 máme zobrazenou hodnotu uloženou právě v prvku *Label1.Text*, která byla přiřazena voláním metody kurzovního lístku ve vztahu Kč – Euro.

9. Panel nástrojů a drag & drop modul

Panel nástrojů neboli *Toolbox* je nepostradatelným nástrojem pro tvorbu webových formulářů a vkládání ovládacích prvků různého charakteru do ASP stránky. Je rozdělen do několika kategorií podle zařazení k příslušným funkčnostem ovládacích prvků. Na obrázku máme rozbalenu kategorii *Standard*, která obsahuje nejznámější a často používané prvky webových stránek, jako jsou tlačítka, textboxy, různé výběrové menu, zaškrtávací pole a další. Těmto prvkům se budeme věnovat podrobněji v kapitole o webových formulářích. Dalšími kategoriemi jsou ovládací prvky s daty, validační nástroje, navigační a logovací ovládací prvky a další. I těmto prvkům se budeme věnovat v konkrétních kapitolách. Jednotlivé ovládací prvky vkládáme do stránky metodou *drag & drop* – táhni a pusť. Přetažením ovládacího prvku do stránky v návrhovém zobrazení tak umístíme konkrétní prvek a vygeneruje se nám automaticky odpovídající kód. Např. vložením tlačítka získáme tento kód:

```
<asp:Button ID="Button1" runat="server" Text="Button" />
```



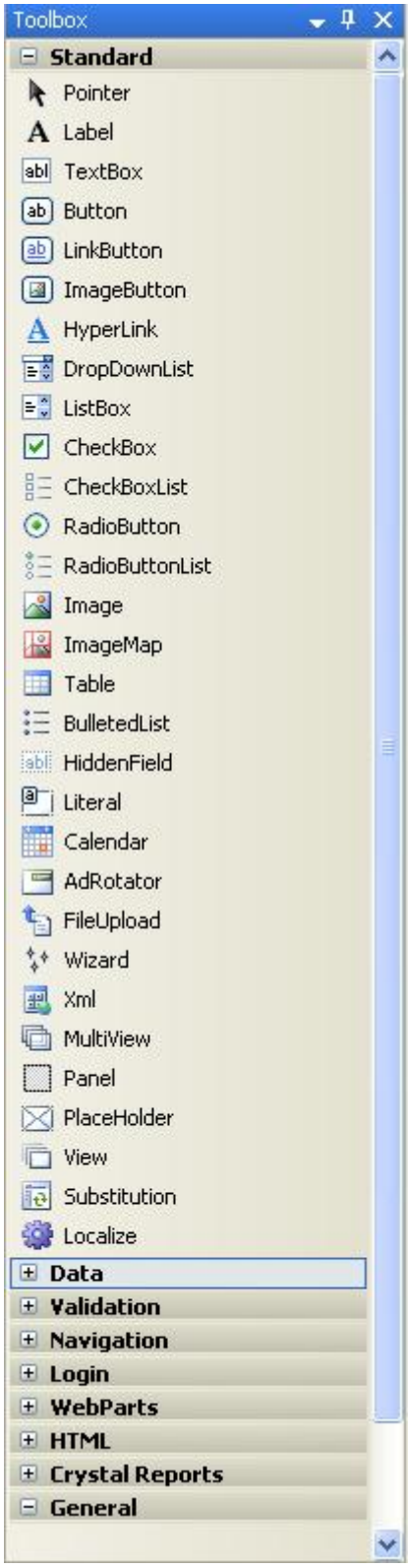
Obr. 2.6 Sledování obsahu proměnných

Jedná se o serverový ovládací prvek, proto atribut runat="server", dále máme vygenerováno jednoznačné ID a implicitní popisek tlačítka. Tyto hodnoty jsou generované Visual Studiem a je často nutné je změnit. To můžeme provést buďto ručně v kódu a nebo v design módu v okně *Properties*. Okno *Properties* je aktivní po kliknutí na konkrétní ovládací prvek a umožňuje nám pohodlně měnit vlastnosti daného ovládacího prvku. Po provedení změn zjistíme, že se v kódu automaticky aktualizovaly námi změněné hodnoty, popřípadě doplnily další.

Pojem k zapamatování:

Pojmenování ovládacích prvků

Je nepsaným pravidlem pojmenovávat prvky složením zkratky názvu ovládacího prvku s naším pojmenováním, např. *btnOdeslat* pro tlačítko, *tbJmeno* pro textbox, *lblNazev* pro Label apod. Je to vhodné zvláště v případech, kdy procházíte zdrojový kód a ihned poznáte o jaký ovládací prvek se jedná. V případě špatného pojmenování tak můžete při ladění kódu narazit např. na *Jmeno.Text = "Pepa"* a v takovém případě opravdu nepoznáte, o jaký ovládací prvek se jedná.

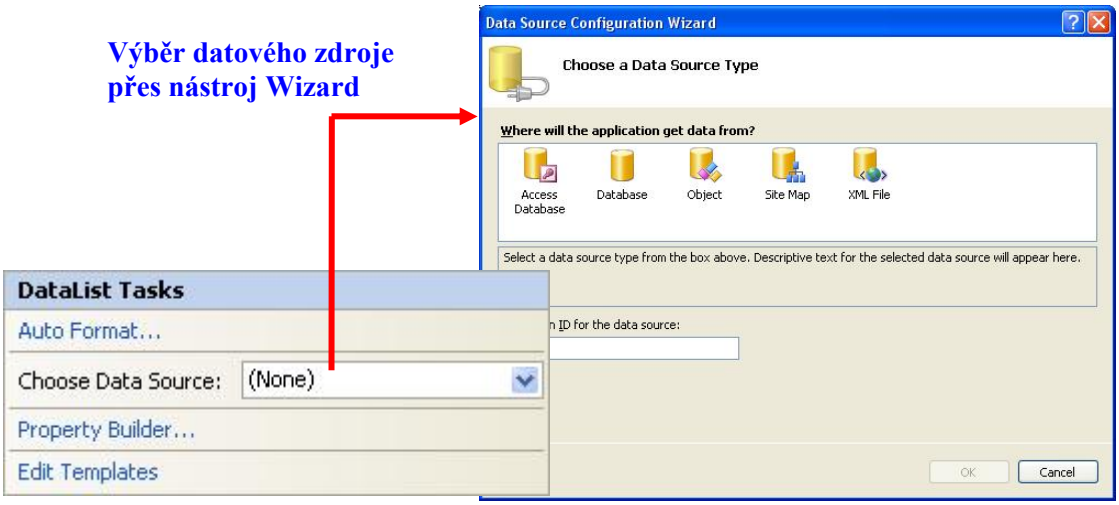


Obr. 2.7 Panel nástrojů

10. Pokročilé průvodce

Visual Studio komfortně usnadňuje práci programátorům prostřednictvím pokročilých průvodců, tzv. *Wizardů*. Práce s nimi je intuitivní i v oblastech, kde si vývojář není stoprocentně jist, jak by postupoval při ručním psaní kódu. V tomto případě mu průvodce pomůže vygenerovat požadovaný kód. Dalším nástrojem k usnadnění práce jsou takzvané *tasky* jednotlivých ovládacích prvků. Jsou

dostupné v pravém horním rohu v příslušném čtverečku, který na kliknutí reaguje vysunutím konkrétních nabídek. Tyto tasky se s oblibou využívají u ovládacích prvků propojených s daty, kde mimo rychlý AutoFormát můžeme prostřednictvím wizardu zajistit rychle propojení s požadovaným datovým zdrojem. Uvedeným vlastnostem se budeme více věnovat v kapitole Vázání dat.



Obr. 2.8 Příklad *Tasku* ovládacího prvku *DataList* a možnosti rychlého výběru datového zdroje

2.2 Popis prostředí

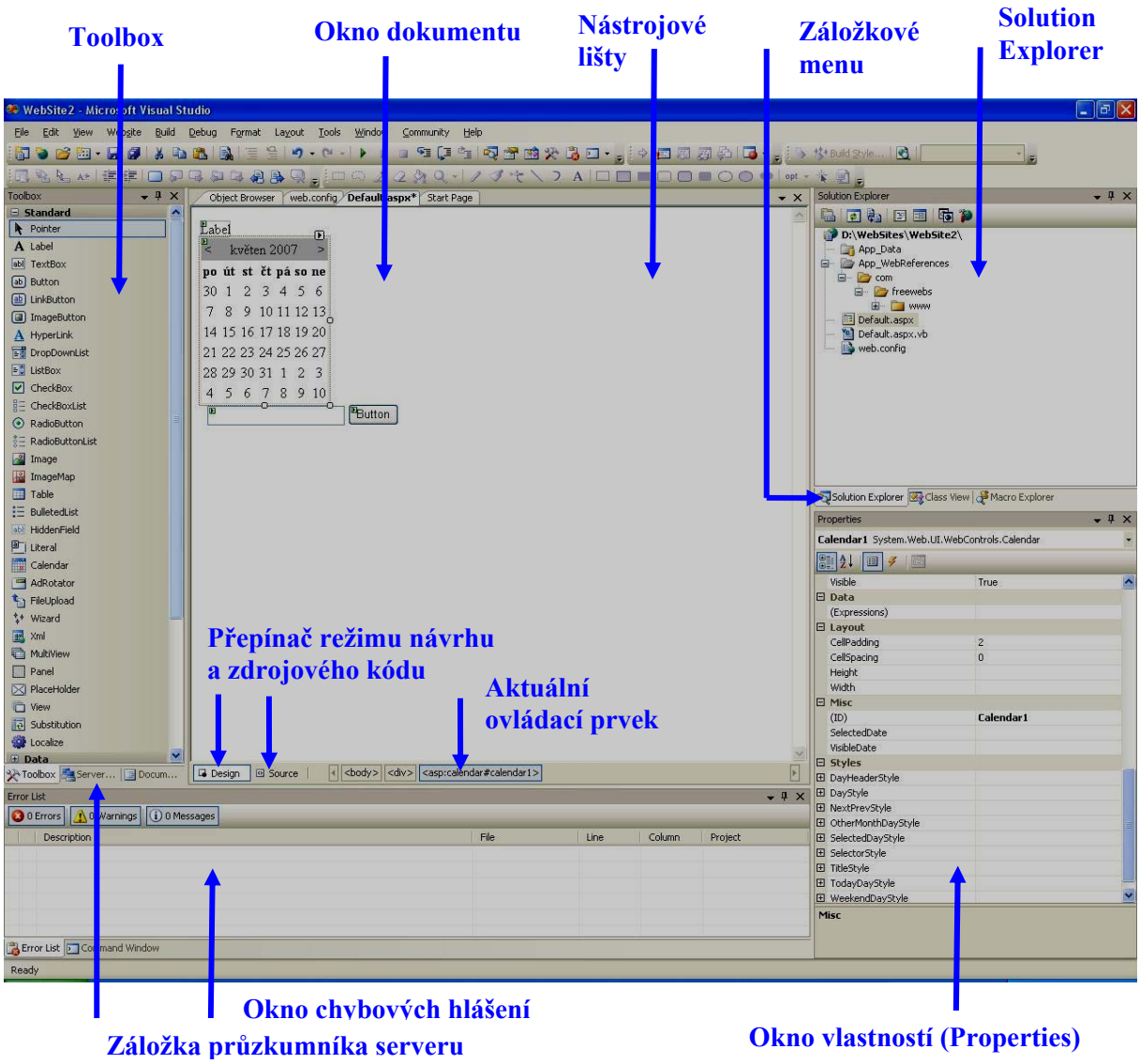
Popsali jsme si vlastnosti a výhody používání Visual Studia. Pojďme se věnovat jeho vzhledu. Ten lze upravit do konkrétní podoby vyhovující uživateli. Jednotlivé nástrojové lišty jdou vybrat a zobrazit v hlavním panelu prostředí.

Obr. 2.9 Rychlé spouštění ladění z nástrojové lišty

Jednotlivá okna se dají přeskupovat, otvírat a zavírat a jsou k dispozici pomocí horkých kláves. Protože vývojové prostředí má mnoho funkcí, jsou jednotlivá okna k dispozici také pomocí záložek. Panely nástrojů a různé typy průzkumníků se dají připnout na své místo pomocí připínáčku v pravém horním rohu.

Okno **Toolboxu** jsme si již popsali v předchozí kapitole, obvykle bývá zobrazeno po levé straně s možností skrytí. Do stejné pozice je často umísťován p**růzkumník serveru** (*Server Explorer*), který slouží pro rychlou administraci databázových spojení. Nejdůležitější částí je samozřejmě **okno dokumentu**, které je pomocí spodních záložek přepínatelné na část návrhu a část zdrojového kódu. Vedle těchto záložek vidíme aktuálně zpracovávaný prvek dokumentu. V horních záložkách máme umístěné již otevřené dokumenty, mezi nimiž se můžeme pohybovat a editovat je. V pravém dolním rohu máme obvykle zobrazeno **okno Properties** – vlastnosti aktuálního ovládacího prvku. V pravém horním rohu můžeme opět pomocí záložek přepínat různé průzkumníky. Nejčastějším průzkumníkem je **Solution Explorer** – průzkumník řešení. V něm jsou zobrazeny všechny adresáře a soubory aktuálně zpracované web aplikace. Solution Explorer je možné také přepínat ze záložkového menu mezi průzkumníkem maker a zobrazením tříd – *Class View*. Zobrazení tříd poskytuje pohled na vytvořenou aplikaci z hlediska obsahu vytvořených tříd a jejich metod a vlastností.

Ve spodní části je zobrazeno okno s chybovými hlášeními. Tyto chyby kódu detekuje Visual Studio a budou zobrazeny, dokud nejsou odstraněny. Okno chybových hlášení obsahuje rovněž záložky. Dá se přepínat mezi seznamem chyb nebo varovných a informačních hlášeních. Chyby a varovná hlášení jsou zobrazována i se svým popisem a číslem řádku, ve kterém se chyba vyskytuje. Mezi varovná hlášení můžeme zařadit detekci zastaralého html kódu, neodpovídající w3c standardu xhtml.



Obr. 2.10 Typické rozvržení vývojového prostředí

Korespondenční úkol – Nápověda k Visual Studiu 2005

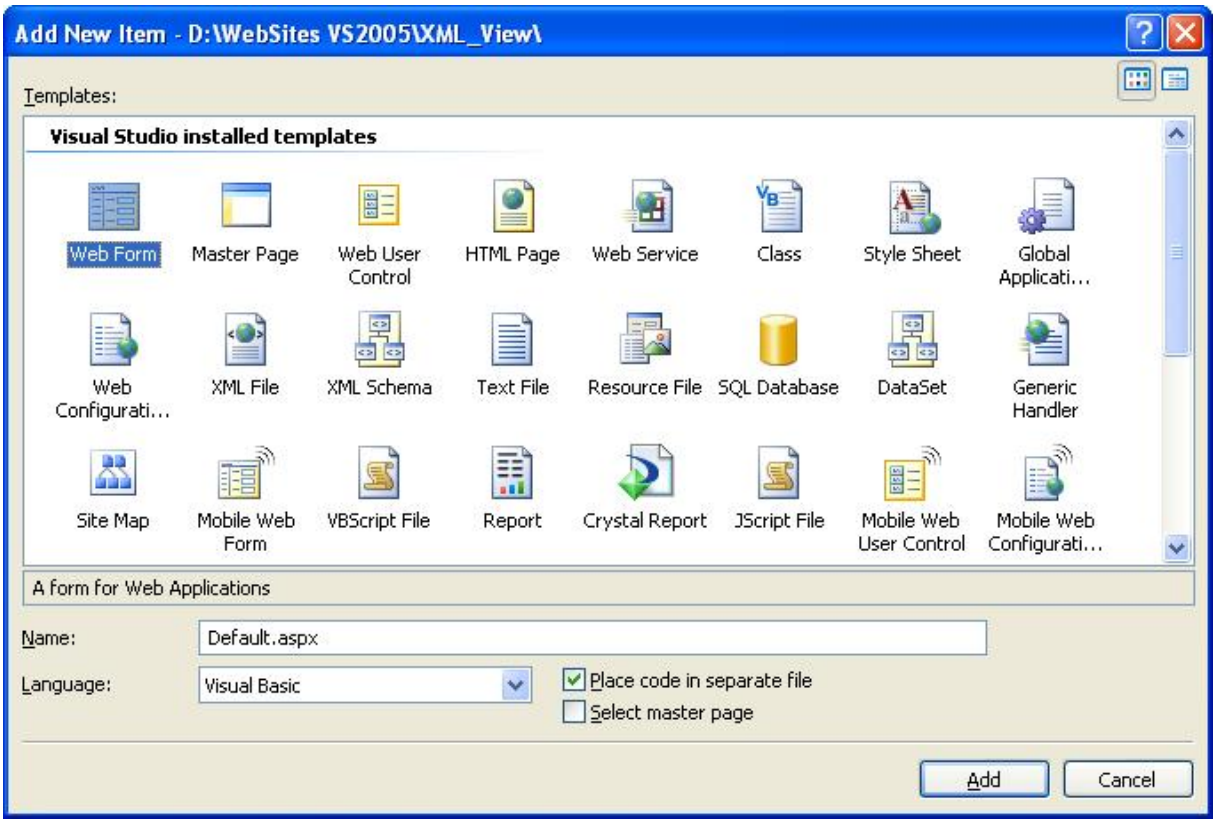
Prostudujte možnosti práce s nápovědou v prostředí Visual Studio 2005. Pokud nejste začátečníkem, což tento kurz nepředpokládá, jistě se dokážete v MSDN nápovědě dobře orientovat.

2.3 Typy souborů web aplikace

V okně průzkumníka řešení máme zobrazen souborový systém otevřené aplikace. Pod názvem aplikace můžeme v rozbalovacím menu vidět soubory a adresáře, které jsou součástí projektu. Mezi adresáře patří *App_Code*, *App_Data*, *App_Themes* s programovým kódem, datovými zdroji jako např. databázové soubory či xml zdroje a aplikační témata vzhledu stránky. Dále pak je možné ukládat obrázky do adresáře *Images*, šablony kaskádových stylů, dokumenty a další.

Mezi typy souborů, se kterými se nejčastěji setkáme, patří: *.aspx*, *.ascx*, *.asmx*, *.vb*, *.cs*, *web.config* a *global.asax*.

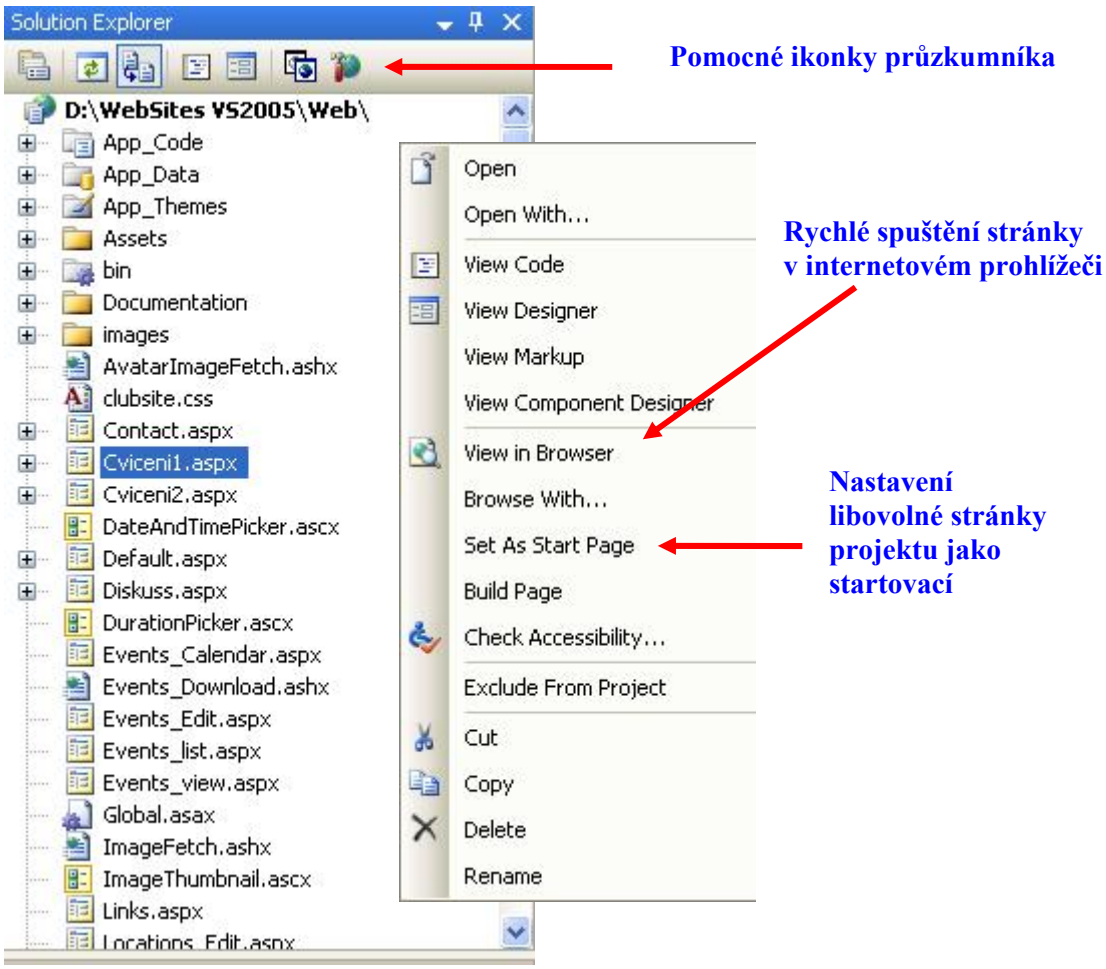
Aspx je přípona ASP web stránek v prostředí .NET. Tyto stránky se po kompilaci zobrazují koncovým uživatelům v internetovém prohlížeči. Příponou *ascx* jsou reprezentovány uživatelské ovládací prvky. Tyto prvky jsou podobné *aspx* stránkám, obsahují však pouze části uživatelského rozhraní, které lze v různých aplikacích opětovně využívat. Tvorbě uživatelských prvků se budeme věnovat v samostatné kapitole. Příponou *asmx* jsou označeny webové služby. Tyto služby se chovají jinak než *aspx* stránky, využívají však stejné zdroje aplikace a různá konfigurační nastavení. Dalším typem dokumentů uložených v projektu aplikace jsou zdrojové kódy oddělené od ASPX stránek. Jsou psány nejčastěji v programovacích jazycích Visual Basic a C# a tomu odpovídají přípony *vb* a *cs*.



Obr. 2.11 Volba vložení nového souboru do aplikace

Projekt web aplikace může obsahovat mnoho typů souborů, neboť v pracovním adresáři máme uloženo prakticky vše, od zdrojových kódů, přes datové zdroje, reporty, dokumentace, šablony a témata, služby, mobilní prvky až po konfigurační nastavení webové aplikace. Vložení nového typu dokumentu provedeme kliknutím pravého tlačítka myši na názvu aplikace v *Solution Exploreru*.
Konfigurační soubor web aplikace se jmenuje *web.config*. Je založen na formátu XML, obsahuje mnoho konfiguračních elementů pro správu stavu aplikace, bezpečnosti, ladění a mnoho dalších nastavení.
Dalším konfiguračním souborem, který však není implicitně vytvořen je soubor *global.asax*. Tento soubor definuje globální proměnné a události a musíme ho v případě potřeby vytvořit sami.
Na obrázku 2.12 vidíme zobrazenou webovou aplikaci v *Solution Exploreru*. Aplikace může obsahovat desítky souborů a proto jsou jednotlivé adresáře sbaleny příslušným tlačítkem. *Aspx* stránky můžeme testovat v prohlížeči kliknutím na příslušnou stránku pravým tlačítkem a volbou *Zobraz v prohlížeči* (*View in Browser*). Tato vlastnost je novinkou ve verzi Visual Studio 2005 a souvisí s vestavěným web serverem. Projekt samozřejmě obsahuje mnoho stránek, a tak při jeho otevření a následném spuštění neví, která stránka je startovací. Tento fakt zajistíme rovněž kliknutím pravého tlačítka na požadované stránce a volbou *Nastav jako startovací stránku* (*Set As Start Page*). Uvedená situace je zobrazena rovněž na obrázku 2.12. *Solution Explorer* obsahuje ikonky pro rychlé přepínaní následujících situací: Zobrazení vlastností dokumentu (*Properties*), Refresh kořenového adresáře

aplikace, zobrazení vztahujících se souborů ke stránce (*Nest related files*), dále pak zobrazení zdrojového kódu stránky nebo vzhledu v design režimu, import aplikace na web server a konfigurační administrační nástroj.



Obr. 2.12 Volba vložení nového souboru do aplikace

2.4 Vytvoření prvního projektu

Po teoretickém úvodu k Visual Studiu 2005 nastal čas na vytvoření naší první ASP.NET stránky. V menu *File* klikneme na položku *New Website* a zobrazí se nám výběr šablon tak jak jsme si ho zobrazili na obrázku 2.1. Vybere volbu *ASP.NET Web Site* a v políčku *location* vybereme cestu pro uložení webové stránky. Je vhodné si vytvořit nějaký souhrnný adresář, do kterého budeme jednotlivé web stránky tohoto učebního textu ukládat. Volbu jazyka ponecháme na hodnotě Visual Basic, neboť v tomto předmětu budeme pracovat právě s tímto jazykem. Potvrdíme tlačítkem OK a otevře se nám připravená plocha s prázdným obsahem v záložce návrhu *Design* a s připravenou základní kostrou stránky v záložce *Source* pro psaní zdrojového kódu. V *Solution Exploreru* vidíme vytvořenou stránku *Default.aspx*, kterou si můžeme přejmenovat podle svého uvážení.

Přepneme-li do režimu psaní zdrojového kódu, uvidíme na prvním řádku tuto konstrukci:

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
Inherits="_Default" %>
```

Uvedené konstrukci se říká stránková direktiva, obsahuje speciální konstrukce pro zpracování stránky. Implicitně vygenerovaná direktiva nám říká, že jazykem bude Visual Basic a stránka je připravena na zpracování zdrojového kódu v souboru *Default.aspx.vb*. To znamená, že zdrojový kód skriptů je oddělen od kódu značkovacího jazyka HTML a ASP ovládacích prvků. Tomuto oddělení se říká „kód

v pozadí“ *(code-behind)* a je výhodný u rozsáhlejších projektů, kdy odpadá nepřehlednost, které se někdy říká „špagety kód“. Tento termín původně označuje míchání kódů skriptu do HTML značek a přes trochu odlišné pojetí ASP.NET stránek zůstal zachován pro označení dlouhého kódu se skripty i značkami v jednom souboru. Přesto však je tento zápis, kterému se říká přímý kód (*inline code*), oblíben a při kratších kódech a méně obsáhlých zdrojových kódech dokonce výhodný. Kód je stejně oddělen v bloku <*script*>, podpora IntelliSense stále funguje a při přepínání mezi design módem a zdrojovým kódem máme veškerý zdrojový kód před sebou.

Budeme tedy vytvářet kratší příklady, proto můžeme zvolit přímý tvar zápisu – Inline kód. Ve stránkové direktivě tedy ponechme pouze atribut použitého jazyka (viz. následující příkad). V prvním příkladu si všimněme atributu `runat="server"` u formuláře. Formulář je tedy serverovým prvkem, nikoli HTML elementem a budeme do něj v další části příkladu vkládat serverové ovládací prvky. Nejprve si ukažme, že můžeme použít v ASP.NET stránkách nějakou běžnou řídicí konstrukci, například cyklus napsaný v programovacím jazyce Basic, samozřejmě ve verzi VB.NET.

Příklad – První stránka v ASP.NET

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Příklad 1</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
    <p>Naše první stránka v ASP.NET</p>
    <%  Dim Krok As Integer
            For Krok = 0 To 5%>
             <font size="<%=Krok%>"> Cyklus For o 6 krocích.
</font>
             Tohle je <%=Krok%>. krok. <br />
          <% Next %>
    </div>
    </form>
</body>
</html>
```

Uvedenou konstrukci cyklu *For* jsme použili přímo v HTML kódu, pomocí tzv. vestavěného bloku kódu. Ten je uvozen podobně jako direktiva znaky `<%` a `%>` a musí být napsán v jazyce, který určuje stránková direktiva. Tento způsob psaní vestavěného kódu je zachován z důvodu zpětné kompatibility s předchozími verzemi ASP a dnes se využívá pouze pro rychlé vypsání nějaké kontrolní hodnoty či výpočtu. A proto i my budeme zdrojový kód raději psát na své patřičné místo. Než budeme pokračovat dále, můžeme si naši první stránku spustit v prohlížeči.

V příkladu budeme pokračovat přidáním serverových ovládacích prvků do formuláře:

Přidání tlačítka, popisku a textboxu do formuláře

```
<asp:Label ID="lblPozdrav" runat="server"></asp:Label>
<asp:Button ID="btnOdeslat" runat="server" Text="OK"
OnClick="btnPozdrav" />
<asp:TextBox ID="tbJmeno" runat="server"></asp:TextBox>
```

Vytvořili jsme tedy tři serverové ovládací prvky popisek, textbox a tlačítko. Serverovým ovládacím prvkům bude věnována samostatná kapitola. V našem příkladě zapíšeme jméno do textboxu a po stisknutí tlačítka se zobrazí do popisku pozdrav. To samozřejmě vyžaduje dopsání procedury, která obslouží událost stisknutí tlačítka definovanou jako `OnClick="btnPozdrav".` Proceduru napíšeme do skriptu, který bude generován na serveru a umístíme ho do elementu hlavičky ASP stránky `<head runat="server">`. Výsledek si můžeme opět vyzkoušet zobrazením stránky v prohlížeči.

Vytvoření skriptu s ovládací procedurou

```
<script runat="server">
        Sub btnPozdrav(ByVal obj As Object, ByVal e As EventArgs)
            lblPozdrav.Text = " Dobrý den pane " + tbJmeno.Text
        End Sub
    </script>
```

Na závěr prvního příkladu si vytvořme ještě několik funkcí a představme funkci Page_Load. Ta provede své tělo při načtení stránky do prohlížeče. Uvedený zdrojový kód vložme opět do stránky uvozené počátkem a koncem elementu skript, přičemž na pořadí zápisu funkcí nezáleží.

Funkce Page_Load a vytvoření dalších funkcí

```
sub Page_Load(obj as object, e as eventargs)
   Nasobeni(8, 9)
   Deleni(4, 2)
   Scitani(4, 5)
   Odcitani(2,4)
end sub

Sub Nasobeni(ByVal intA As Integer, ByVal intB As Integer)
  Response.Write(intA * intB & "<br>")
End Sub

Sub Deleni(ByVal intA As Integer, ByVal intB As Integer)
  Dim Vysledek As Integer
  Vysledek = intA / intB
  Response.Write(Vysledek & "<br>")
End Sub

Sub Scitani(ByVal intA As Integer, ByVal intB As Integer)
  Response.Write(intA & "+" & intB & "=" & intA + intB & "<br>")
End Sub

Sub Odcitani(ByVal intA As Integer, ByVal intB As Integer)
  Response.Write(intA - intB & "<br>")
End Sub
```

Funkce Page_Load obsahuje tři další funkce, které se provedou při načtení stránky do prohlížeče. Uvedené funkce si tedy musíme nejprve vytvořit. Funkce jsou definovány se dvěma parametry typu Integer. Tyto dvě čísla: *intA* a *intB* budeme násobit, dělit, sčítat a odčítat. Všimněme si malých rozdílů v těle funkcí, které byly vytvořeny jen jako ukázka. Funkce dělení obsahuje v těle proměnnou výsledek, která je pak vypsána do prohlížeče. Metoda *Write* objektu *Response* vypisuje obsah v kulatých závorkách do prohlížeče, vidíme že proměnné můžeme kombinovat s HTML kódem, který se má zobrazit. U funkce sčítání je zápis také odlišný a aritmetický výraz je zapsán celý i se svými znaménky. Máme tedy za sebou první příklad tvorby ASP.NET stránek. Pokud nám jsou některé pojmy nejasné, určitě nám je pomohou vysvětlit následující kapitoly.

Korespondenční úkol – Visual Studio 2005

Prostudujte prostředí Visual Studio 2005 podrobněji, vyhledejte možnosti nastavení editoru, prostředí, nastavení cest k projektům, možnosti XHTML validace a další.

Ke kapitole 2 je přiřazena demonstrační animace č. 1

Animace č. 1 je věnována základům práce v ASP.NET, které popisuje tato kapitola.

Shrnutí kapitoly

Součástí studia je tzv. design mód neboli záložka pro tvorbu návrhu. Zde můžete jednotlivé součásti stránky navrhnout bez nutnosti psaní zdrojového kódu, ten se vygeneruje automaticky. Výběrem šablony zahájíte práci na vašem webovém projektu. V podstatě jde o to, jaký typ aplikace chcete vyvíjet přednostně, ať už ASP stránku či webovou službu. Po zvolení šablony se implicitně vygeneruje příslušná kategorie. Oba typy aplikací se však kompilují a vykonají obdobně, a tak můžete během vývoje přidávat do projektu další stránky či služby.

IntelliSense je výborná pomůcka pro zjednodušení psaní, která mimo jiné také ve svém důsledku minimalizuje počet syntaktických chyb, které uživatel může vytvořit. *IntelliSense* je nástroj který na základě vámi psaného zdrojového textu nabízí možné použití výběru dalších zdrojových slov. Můžete tedy například vybrat ze seznamu dlouhý název ovládacího prvku, který byste jinak psali ručně. *IntelliSense* také nabízí všechny dostupné metody a vlastnosti při použití konkrétního objektu, třídy či proměnné.
Pokud se rozhodneme přidat nějaký prvek do ASP stránky, můžeme ho samozřejmě přidat ručně a nebo použít již zmíněnou záložku *Design*. V tomto případě se po vložení prvků generuje zdrojový kód automaticky. Důležité je, že se generuje včetně ID a svých základních atributů. Ty můžeme pak ručně upravit a nebo využít okno *Properties*, ve kterém příslušné vlastnosti nastavíme a opět Visual Studio kód vygeneruje za nás.

Nástroj *Outlining* umožňuje větší přehlednost kódu v možnosti takzvaného sbalení a rozbalení textu. Pokud tedy nějakou část kódu nepotřebujeme vidět, můžeme ji jednoduše sbalit kliknutím na znaménko mínus.

Předchozí verzi studia jste mohli využívat pro tvorbu ASP.NET stránek ve spolupráci s Internetovou informační službou (IIS). Visual Studio 2005 obsahuje integrovaný web server a na něm je stránka s vygenerovaným číslem portu spuštěna. K základním vlastnostem vývojového prostředí patří možnosti ladění projektů. To se týká samozřejmě i webových aplikací. Pokud chceme aplikaci ladit, zvolíme z nástrojové lišty tlačítko *Start Debugging*. Obdobně jako u desktopových aplikací využíváme zarážky – tzv. *breakpointy*. Pokud máme umístěnou zarážku kdekoliv v programu a spustíme ladění, program se zastaví na daném řádku kódu.

Panel nástrojů neboli *Toolbox* je nepostradatelným nástrojem pro tvorbu webových formulářů a vkládání ovládacích prvků různého charakteru do ASP stránky. Je rozdělen do několika kategorií podle zařazení k příslušným funkčnostem ovládacích prvků.
Mezi typy souborů, se kterými se nejčastěji setkáme, patří: *.aspx*, *.ascx*, *.asmx*, *.vb*, *.cs*, *web.config*, a *global.asax*. *Aspx* je přípona ASP web stránek v prostředí .NET. Tyto stránky se po kompilaci zobrazují koncovým uživatelům v internetovém prohlížeči. Příponou *ascx* jsou reprezentovány uživatelské ovládací prvky. Příponou *asmx* jsou

označeny webové služby. Tyto služby se chovají jinak než aspx stránky, využívají však stejné zdroje aplikace a různá konfigurační nastavení. Dalším typem dokumentů uložených v projektu aplikace jsou zdrojové kódy oddělené od ASPX stránek. Jsou psány nejčastěji v programovacích jazycích Visual Basic a C# a tomu odpovídají přípony *vb* a *cs*.

Úkol k řešení 2.1 – Vytvoření elementárního kalkulátoru

Řešený příklad z této kapitoly předělejte tak, aby jednotlivé operace násobení, dělení, sčítání a odečítání bylo možné ovládat tlačítky a obě čísla zadávat z textboxu.

Úkol k řešení 2.2 – Odstranění vestavěného kódu

Předělejte vestavěný kód v řešeném příkladu do skriptu, například do funkce Page_Load.

Kontrolní otázka 2.1

Jak zajistíme spuštění stránky v jiném prohlížeči než v implicitně nastaveném (pravděpodobně IE) ?

Kontrolní otázka 2.2

Vyjmenujte výhody tvorby ASP.NET stánek v prostředí Visual Studia 2005.

Kontrolní otázka 2.3

K čemu slouží nástroj IntelliSense?

Kontrolní otázka 2.4

Jaké typy souborů mohou být součástí web aplikace?

Kontrolní otázka 2.5

Jak se zajišťuje ve Visual Studiu 2005 možnost ladění web aplikace a co jsou to zarážky?

Kontrolní otázka 2.6

K čemu slouží soubor web.config?

3. OBJEKTY V ASP.NET

Čas ke studiu: 2 hodiny

Cíl Po prostudování této kapitoly budete umět

- vysvětlit princip řazení jmenných prostorů .NET Frameworku
- popsat událostní model ASP.NET
- správně programovat obsluhu postBack po načtení stránky do prohlížeče
- pracovat s objekty Cookies, Response a Request
- pracovat s proměnnými relace a aplikace
- nastavit a obsloužit monitorování aplikace při trasování

Výklad

Charakter web aplikace je bezstavový (stateless), to znamená, že celý proces komunikace končí obdržením stránky na klientském počítači. Na serveru jsou uvolněny prostředky, zničeny objekty web stránky a veškerých informací pro klienta. Proto, abychom mohli budovat moderní robustní web aplikace, musela technologie ASP.NET tento problém vyřešit událostmi řízeným modelem, mechanismem stavu zobrazení a technikou postback pro odesílání stránky změněné uživatelem zpět na server.

3.1 ASP.NET a jmenné prostory

Architekturu systému .NET Framework tvoří stovky (dalo by se i říct tisíce) šablon objektů pro tvorbu konzolových aplikací, aplikací Windows Form, ASP.NET aplikací, webových služeb a služeb windows. Tyto objekty jsou organizovány ve jmenných prostorech. Jmenné prostory připomínají adresářovou strukturu a umožňují seskupit spolu související funkce a typy na jednom pojmenovaném logickém místě, čímž je vývojářům usnadněna orientace. Dále řeší konflikty v pojmenování, neboť jméno funkce či typu musí být jedinečné pouze v daném jmenném prostoru. Pro zápis se používá tečková notace, kdy jména jednotlivých do sebe zanořených prostorů a jméno typu umístěného ve jmenném prostoru jsou oddělena tečkami.

Výchozí jmenný prostor, který obsahuje kromě většiny dalších systémových prostorů deklarace všech základních tříd a datových typů, událostí, atributů a výjimek, se jmenuje ***System***. Základní podobu každé stránky ASP.NET aplikace tvoří šablona objektu ***System.Web.UI.Page***. Každá zkompilovaná web stránka bude postavena na uvedené šabloně tohoto objektu a bude mít určen způsob své funkčnosti.

Příklad – Import jmenného prostoru

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

<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.OleDb " %>
```

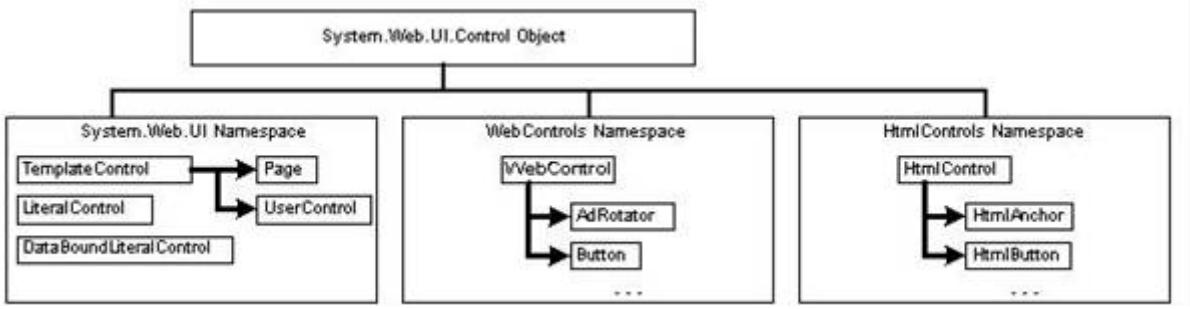
Každý jmenný prostor je kolekcí šablon objektů pro příslušnou funkcionalitu aplikace. Pokud budeme chtít například pracovat s OLE databází a využívat objektů pro vázání dat, budeme muset příslušné jmenné prostory naimportovat.

To provedeme na začátku zdrojové *aspx* stránky ihned za direktivou *Page* podle uvedeného příkladu. Zvídavější čtenáře určitě napadne, že jsme v předchozí kapitole už web stránku vytvářeli i s konkrétními objekty, přesto jsme žádný jmenný prostor neimplementovali. Proč?

Některé jmenné prostory je třeba využívat tak často, že jsou do ASP.NET aplikace importovány automaticky. Nemusíme pro ně využívat direktivu *Import*. Jedná se o tyto jmenné prostory:

- System
- System.Collections
- System.Collections.Specialized
- System.Configuration
- System.IO
- System.Text
- System.Text.RegularExpressions
- System.Web
- System.Web.Caching
- System.Web.Security
- System.Web.SessionState
- System.Web.UI
- System.Web.UI.HtmlControls
- System.Web.UI.WebControls

Důležitým faktem související s hierarchickým řazením jmenných prostorů a jejich tečkovou konvencí je skutečnost, že importem konkrétního jmenného prostoru neimportujete jmenné prostory jež jsou definovány pod ním, tyto jmenné prostory musíme v případě potřeby importovat také (viz. příklad).

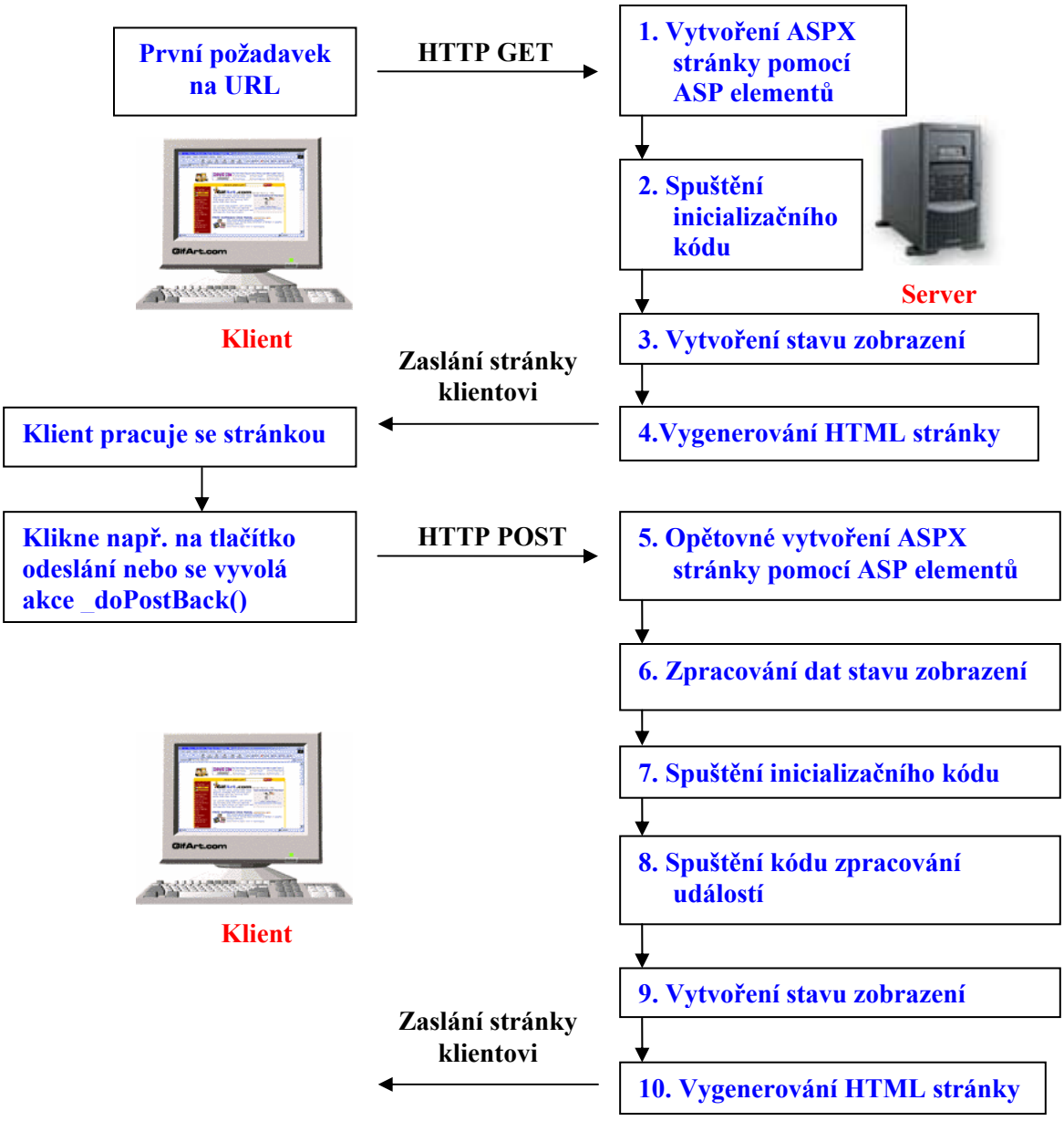


Obr. 3.1 Organizace objektů do jmených prostorů

3.2 Princip zpracování událostí ASP.NET

V úvodu kapitoly jsme si popsali web jako bezstavové médium a sdělili možnosti technologie ASP.NET. Jak tedy funguje událostní model? Po zaslání prvního požadavku klienta, jenž je pouhé zapsání URL adresy do webového prohlížeče, musí server vytvořit objekty stránky, vykonat inicializační kód a přetvořit webové prvky do HTML podoby, které poskytne klientovi. Poté, co se na straně klienta spustí událost, která zajistí odeslání stránky zpět na server (postback), server znovu vytváří objekty stránky a kontroluje tzv. stav zobrazení stránky. V další fázi musí zjistit, která operace vyvolala *postback* a tuto událost obslouží. V této chvíli se obvykle zpracovává na serveru nějaká zásadní akce, pro kterou byla stránka poslána zpět serveru. Následuje aktualizace objektů stránky a stavu zobrazení. Poté se opět generuje HTML kód z příslušných ASP objektů, stránka je poskytnuta klientovi a objekty stránky jsou uvolněny z paměti. Pokud dojde k další události *postback*, celý proces se znovu zopakuje. Uvedený model zpracování ilustruje obrázek 3.2.
Vzpomeňme na příklad z předcházející kapitoly, kde jsme do textboxu psali jméno a poté odeslali formulář stiskem tlačítka. Stránka po postbacku vyvolá následující události:

1. Page.Init -> 2. Page.Load -> 3. TextBox.TextChanged
4. Buton.Click -> 5. Page.PreRender -> 6. Page.Unload



Obr. 3.2 Model událostí ASP.NET stránky

3.3 Objekty se kterými budeme pracovat

Page

Pohledem do struktury jmenných prostorů zjistíme, že každý webový formulář je instancí třídy Page jmenného prostoru *System.Web.UI*. To znamená, že máme ihned k dispozici příslušné vestavěné funkce a vlastnosti jako například IsPostBack. Tato vlastnost nám tedy určuje, zdali došlo k události postBack na základě vyvolání nějaké události od uživatele, a nebo je stránka načtena do prohlížeče úplně poprvé. V demonstračním příkladu si uvedenou vlastnost můžete ověřit.

Jednou z nejdůležitějších událostí aplikace je samotné načtení stránky do prohlížeče. Tento proces je reprezentován procedurou `sub Page_Load(obj as object, e as eventargs).`

Pojem k zapamatování: Provedení Page_Load

Důležitou skutečností je fakt, že stránka je načítána po každém postBacku, tak jak jsme si to demonstrovali na obrázku modelu událostí. Stránka je načtena do prohlížeče, a tudíž je vyvolána procedura Page_Load i v situaci, kdy pouze klikneme na tlačítko refresh prohlížeče či klávesu F5. To znamená, že se pokaždé provede tělo procedury Page_Load.

To samozřejmě mnohdy nechceme a proto je tato událost úzce svázána právě s výše popsanou vlastností IsPostBack. V demonstračním příkladu tedy vidíme, které události jsou provedeny po načtení stránky do prohlížeče a které po zpracování postBacku od uživatele. V přiložené animaci vidíme i krokování této stránky.

Příklad – Vlastnost IsPostBack objektu Page

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
<title>Příklad PostBack </title>

<script runat="server">
sub Page_Load(obj as object, e as eventargs)
  If not Page.IsPostBack then
      lblMessage.Text = "Ahoj, právě jsi navštívil tuto stránku."
  else
      lblMessage.Text = "A není to poprvé, co se načetla."
 end if
end sub

sub Submit(obj as object, e as eventargs)
    lblMessage2.Text = " A tohle je text po stisknutí tlačítka"
end sub
</script>
</head>
<body>
    <form id="Form1" runat="server">
         <asp:button id="btSubmit" onclick="Submit" Text="odeslat"
         <asp:label ID="lblMessage2" Runat="server" /><br />
         <asp:label id="lblMessage" Runat="server" />
</form>
</body>
</html>
```

Response a Request

Objekt Response umožňuje komunikaci serveru s klientem a jeho protipólem je objekt Request, který naopak od klienta odesílá požadavky na server.
Tyto objekty využijeme při práci s Cookies jejich nastavením a opětovným získáním. Již v minulé lekci jsme využili metodu *Write* objektu Response, když jsme chtěli něco zapsat na výstup. Tuto metodu často používají programátoři i při výpisu proměnných místo samotného ladění. Objekt Response můžeme také využít při přesměrování metodou *Redirect,* viz. následující příklad.

Příklad – přesměrování pomocí metody Redirect objektu Response

```
Sub BtnPresmeruj(ByVal sender As Object, ByVal e As EventArgs)
  Response.Redirect("http://www.google.com")
End Sub
```

Cookies

Cookies zajisté všichni známe. Jsou to malé textové soubory uložené na pevném disku klienta, které jsou k dispozici pro opětovné použití, když uživatel po nějaké době opět navštíví příslušnou stránku. Aby nemusel některé hodnoty, popřípadě nastavení znovu zadávat do prohlížeče, aplikace si je načte z Cookies. Obvykle se jedná o oslovení, datum poslední návštěvy webu a podobně, tedy informace, se kterou se nemusí zatěžovat server ukládáním a načítáním z nějaké databáze. Tyto informace také nejsou klíčové, protože mohou expirovat, popřípadě je uživatel může smazat nebo nemusí být vůbec povoleny.

Pojem k zapamatování: Práce s Cookies

Práce s Cookies není složitá. Buďto využíváme Cookies pro jednu hodnotu, nebo jedno Cookies udržuje informaci o více položkách. V prvním případě definujeme Cookies tímto způsobem: Response.Cookies("userName").Value = "Pepa".
Ve druhém případě je Cookies definováno jako několik dvojic klíč/hodnota:

```
Response.Cookies("MojeCookie")("Jmeno") = "Pepa"
Response.Cookies("MojeCookie")("velikostFotky") = "50"
```

Jako demonstrační příklad uveďme načtení dvojic klíč/hodnota Cookies pro jméno uživatele, velikost ovládacího tlačítka, datum návštěvy stránky a informaci o používaném prohlížeči. Nejprve vytvořme HTML formulář se dvěma tlačítky a labely.

Příklad – Formulář pro příklad s Cookies

```
<body>
    <form id="form1" runat="server">
    <div>
    <asp:Label ID="Label1" Runat="server" ForeColor="Red" /><br />
    <asp:Button ID="Button1" runat="server" Style="z-index: 100;
    left: 296px; position: absolute;
    top: 16px" Text="ok" OnClick="btnClick" Height="20px" />
    <asp:Label ID="Label2" Runat="server" ForeColor="Blue" style="z-
    index: 101; left: 280px; position: absolute; top: 72px" />
    <asp:Button ID="Button2" runat="server" Style="z-index: 103;
    left: 352px; position: absolute;top: 16px" Text="Zpět velikost
    tlačítka 1" OnClick="btnZpet" />
    </div>36,
    </form>
</body>
</html>
```

Při prvním načtení stránky vytvořme uvedené Cookies pomocí objektu Response. Při další události *_doPostBack*, například stisknutí tlačítka, získejme hodnotu jména uživatele z Cookies pomocí objektu Request, vypišme jej do prohlížeče a nastavme velikost ovládacího tlačítka, kterou

získáme rovněž z Cookies. Pro kontrolu vytvořme druhé tlačítko, které hodnotu tlačítka vrátí zpět na původní hodnotu. Pro připravený formulář dopišme následující skript a ověřme funkčnost programu.

Příklad – Cookies

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Příklad Cookies</title>
    <script runat="server">
sub Page_Load(obj as object, e as eventargs)
dim strVariable as string
  If Not (Page.IsPostBack) Then
   Response.Cookies("MojeCookie")("Jmeno") = "Marek"
   Response.Cookies("MojeCookie")("velikost") = "50"
   Response.Cookies("MojeCookie")("LastVisit") =
   DateTime.Now.ToString
   Response.Cookies("MojeCookie")("UserAgent") =
   Request.ServerVariables("HTTP_USER_AGENT")
   Response.Cookies("MojeCookie").Expires = DateTime.Now.AddYears(1)
     For Each strVariable In Response.Cookies("MojeCookie").Values
       Label1.Text += "<b>" & strVariable & "</b>:" &
       Request.Cookies("MojeCookie")(strVariable) & "<br>"
     Next
  Else
    Page.Title = Request.Cookies("MojeCookie")("Jmeno")
    Button1.Height = Request.Cookies("MojeCookie")("Velikost")
    Label1.Text = "Vaše jméno je:" &
    (Request.Cookies("MojeCookie")("Jmeno"))
    Label2.Text += "Tlačítko má:" &
    (Request.Cookies("MojeCookie")("Velikost") & "pixelů protože je
    velikost načtena z Cookies<br>")
  End If
End Sub

Sub btnClick(ByVal obj As Object, ByVal e As EventArgs)
   'pouze se provede postBack na server
End Sub

Sub btnZpet(ByVal obj As Object, ByVal e As EventArgs)
  Button1.Height = "20"
  Page.Title = "Stránka nemá jméno"
   Label2.Text = "Tlačítko má původní velikost:" &
   Button1.Height.ToString & "<br>"
End Sub
</script>
</head>
```

Korespondenční úkol – Expirace Cookies

Ve zdrojovém kódu můžeme vidět i nastavení exspirace hodnot Cookies. Vyhledejte v nápovědě podrobnější informace o možnostech nastavení vypršení Cookies.

Session

Session, neboli relace je důležitým pojmem v oblasti správy stavu webových aplikací a často využívanou skutečností. Každý klient, který má v prohlížeči spuštěnou web aplikaci, udržuje se serverem tzv. relaci neboli Session. V této relaci udržuje svou kolekci informací, která je jedinečná právě pro tohoto klienta.

Pojem k zapamatování: Použití Session

V relaci se často v praxi udržují proměnné nákupního košíku internetových obchodů. Je to typické použití, neboť aplikace je složena z více stránek navazujících za sebou a je nutné tyto informace mezi nimi neustále udržovat.

Jakmile je relace přerušena, dojde ke ztrátě těchto informací. To je možné několika způsoby. Zavřením prohlížeče, přistoupením aplikace z jiného okna prohlížeče, automatickým vypršením z důvodu nečinnosti a nebo programovým kódem na ukončení relace. Vytvořme ukázkový příklad relace složený ze dvou stránek.

Příklad – proměnné Session – Page1.aspx

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
<title>Příklad Session</title>
<script runat="server">
Sub Submit_click(ByVal obj As Object, ByVal e As EventArgs)
  If tbName.Value <> "" Then
      Session("Name") = tbName.Value
      Response.write("Dobrý den pane " & Session("Name") & " !")
  Else
      Response.write("Zapoměl jste napsat své jméno")
  End If
  End Sub
</script>
</head>
 <body>
   <form id="form1" runat="server">
       Zadejte prosím své jméno: <input type="text" id="tbName"
       runat="server"/>
       <asp:button ID="btSubmit" Text="odeslat" Runat="server"
       OnClick="submit_click" />
    </form>
 </body>
</html>
```

Ve zdrojovém kódu první stránky jsme vytvořili proměnnou relace *Name*, do které uložíme uživatelovo jméno. Do procedury Page_Load druhé stránky umístíme kód, který vytvoří uvítací zprávu ziskem jména z této proměnné relace. Obdobným způsobem bychom mohli sdílet proměnné relace mezi více stránkami.

Příklad – proměnné Session – Page2.aspx

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">
    Sub Page_Load(ByVal obj As Object, ByVal e As EventArgs)
        lblOzn.Text = "vítejte  na další stránce pane " &
Session("Name") & " !"
    End Sub
</script>

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Stránka 2</title>
</head>
<body>
    <form id="form1" runat="server">
    <asp:label ID="lblOzn" Runat="server" />
     </form>
</body>
</html>
```

Korespondenční úkol – Ukončení Session

Prostudujte možnosti vypršení relace prostřednictvím *TimeOut.*

Vyhledejte informace o okamžitém zrušení relace.

HttpApplication

Objekt httpApplication umožňuje obdobně jako relace udržet proměnnou typu název/hodnota. Tento objekt však pracuje s aplikací globálně, to znamená že pro proměnnou budou mít všichni klienti přiřazenou stejnou hodnotu. V určitých případech totiž není nutné pro každou relaci vytvářet nové proměnné, pokud od nich očekáváme stejné hodnoty. Takto například můžeme umístit poznámku do zápatí, copyright stránky a podobně. Následující příklad ukazuje použití proměnné Application.

Příklad –Globální HttpApplication

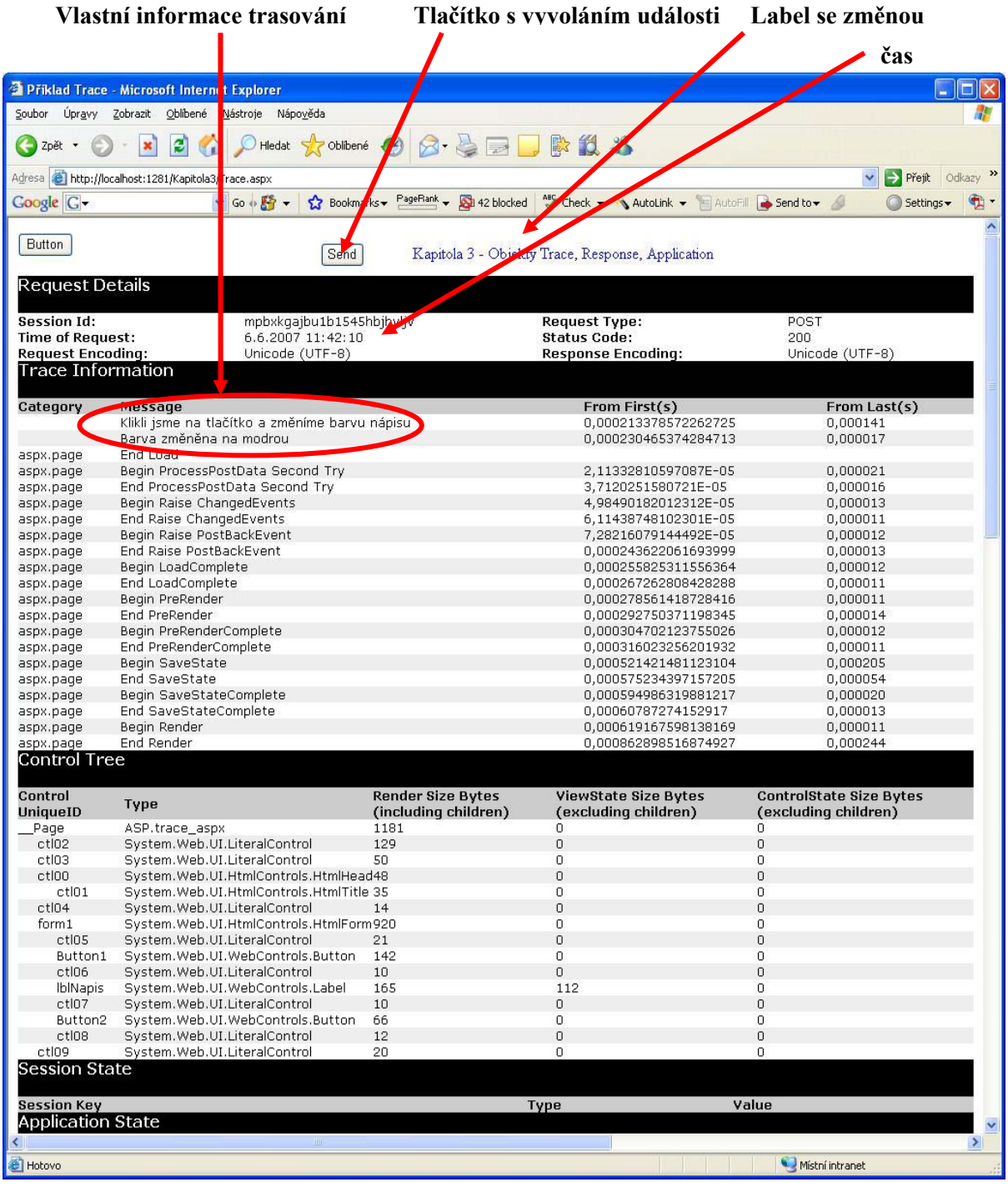
```
Sub page_load(ByVal obj As Object, ByVal e As EventArgs)
       Application("znacka") = "VSB TUO"
End Sub

Sub btn_Click(ByVal sender As Object, ByVal e As EventArgs)
        Page.Title = Application("znacka")
End Sub
```

Trace

Objekt Trace je mocný monitorovací nástroj, využívající nejen při ladění web aplikace. Umožňuje zapisovat události do konkrétního log souboru, ve kterém můžeme spatřit i časové údaje o proběhnutých událostech. Do protokolu o sledování můžeme zapisovat i vlastní zprávy pomocí metod *Trace.Write()* nebo *Trace.Warn()*. Následující příklad demonstruje trasování aplikace se změnou barvy nápisu. Informace o trasování se zobrazují do speciálně vytvořené tabulky, která je zobrazena na obrázku 3.3.

Příklad – Trasování

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">

   Sub btnSend(ByVal sender As Object, ByVal e As System.EventArgs)
        Trace.Write("Klikli jsme na tlačítko a změníme barvu
nápisu")
        If lblNapis.ForeColor = Drawing.Color.Red Then
            lblNapis.ForeColor = Drawing.Color.Blue
            Trace.Write("Barva změněna na modrou")
        Else
            lblNapis.ForeColor = Drawing.Color.Red
            Trace.Write("Barva změněna na červenou")
        End If
    End Sub
</script>

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Trace příklad</title>
    <script runat="server">
        Sub Page_load(ByVal obj As Object, ByVal e As EventArgs)
            Trace.IsEnabled = True
            Trace.TraceMode = TraceMode.SortByCategory
        End Sub
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:Button ID="Button1" runat="server" OnClick="btnSend" />
         <asp:Label ID="lblNapis" runat="server" ForeColor="Red"
         Text="Kapitola 3 - Objekt Trace"></asp:Label>
         <asp:Button ID="Button2" runat="server"
         OnClick="btnPresmeruj" Text="Button" /></div>
    </form>
</body>
</html>
```



Obr. 3.3 Okno s trasovanou aplikací

Ke kapitole 3 je přiřazena demonstrační animace č. 2

Animace č. 2 obsahuje krokování všech příkladů této kapitoly zahrnující práci s objekty Page, Response, Request, Cookie, Session, Application a Trace.

Shrnutí kapitoly

Architekturu systému .NET Framework tvoří stovky (dalo by se i říct tisíce) šablon objektů pro tvorbu konzolových aplikací, aplikací Windows Form, ASP.NET aplikací, webových služeb a služeb windows. Tyto objekty jsou organizovány ve jmenných prostorech.

Výchozí jmenný prostor, který obsahuje kromě většiny dalších systémových prostorů deklarace všech základních tříd a datových typů, událostí, atributů a výjimek, se jmenuje *System*. Základní podobu každé stránky ASP.NET aplikace tvoří šablona objektu *System.Web.UI.Page*.

Po zaslání prvního požadavku klienta, jež je pouhé zapsání URL adresy do webového prohlížeče, musí server vytvořit objekty stránky, vykonat inicializační kód a přetvořit webové prvky do HTML podoby, které poskytne klientovi. Poté co se na straně klienta spustí událost, která zajistí odeslání stránky zpět na server (postback), server znovu vytváří objekty stránky a kontroluje tzv. stav zobrazení stránky. V další fázi musí zjistit, která operace vyvolala *postback* a tuto událost obslouží. V této chvíli se obvykle zpracovává na serveru nějaká zásadní akce, pro kterou byla stránka poslána zpět serveru. Následuje aktualizace objektů stránky a stavu zobrazení. Poté se opět generuje HTML kód z příslušných ASP objektů, stránka je poskytnuta klientovi a objekty stránky jsou uvolněny z paměti. Pokud dojde k další události *postback*, celý proces se znovu zopakuje.
Pohledem do struktury jmenných prostorů zjistíme, že každý webový formulář je instancí třídy Page jmenného prostoru Systém.Web.UI. To znamená, že máme ihned k dispozici příslušné vestavěné funkce a vlastnosti jako například IsPostBack. Tato vlastnost nám tedy určuje, zdali došlo k události postBack na základě vyvolání nějaké události od uživatele a nebo je stránka načtena do prohlížeče úplně poprvé.

Objekt Response umožňuje komunikaci serveru s klientem a jeho protipólem je objekt Request, který naopak od klienta odesílá požadavky na server. Tyto objekty využijeme při práci s Cookies jejich nastavením a opětovným získáním.

Session neboli relace je důležitým pojmem v oblasti správy stavu webových aplikací a často využívanou skutečností. Každý klient, který má v prohlížeči spuštěnou web aplikaci, udržuje se serverem tzv. relaci neboli Session. V této relaci udržuje svou kolekci informací, která je jedinečná právě pro tohoto klienta.

Objekt httpApplication umožňuje obdobně jako relace udržet proměnnou typu název/hodnota. Tento objekt však pracuje s aplikací globálně, to znamená, že pro proměnnou budou mít všichni klienti přiřazenou stejnou hodnotu.

Objekt Trace je mocný monitorovací nástroj, využívající nejen při ladění web aplikace. Umožňuje zapisovat události do konkrétního log souboru, ve kterém můžeme spatřit i časové údaje o proběhnutých událostech. Do protokolu o sledování můžeme zapisovat i vlastní zprávy pomocí metod *Trace.Write()* nebo *Trace.Warn()*.

Úkol k řešení 3.1 – Vytvoření stránky s Cookies a Session

Vytvořte dvě jednoduché stránky, kde budete zobrazovat hodnotu Cookies a Session.

Úkol k řešení 3.2 – Trasování aplikace

Trasujte aplikaci elementárního kalkulátoru z minulé kapitoly.

Kontrolní otázka 3.1

Co je to jmenný prostor?

Kontrolní otázka 3.2

Jak zajistíme aby se po každém obnovení stránky neprováděl kód procedury Page_Load?

Kontrolní otázka 3.3

Jakým způsobem může být ukončena Session?

Kontrolní otázka 3.4

Jaký je rozdíl mezi objekty Session a HttpApplication?

Kontrolní otázka 3.5

Popište princip zpracování událostí v ASP.NET.

Kontrolní otázka 3.6

K čemu slouží objekt Trace?

4. WEBOVÉ FORMULÁŘE

Čas ke studiu: 3 hodiny

Cíl Po prostudování této kapitoly budete umět

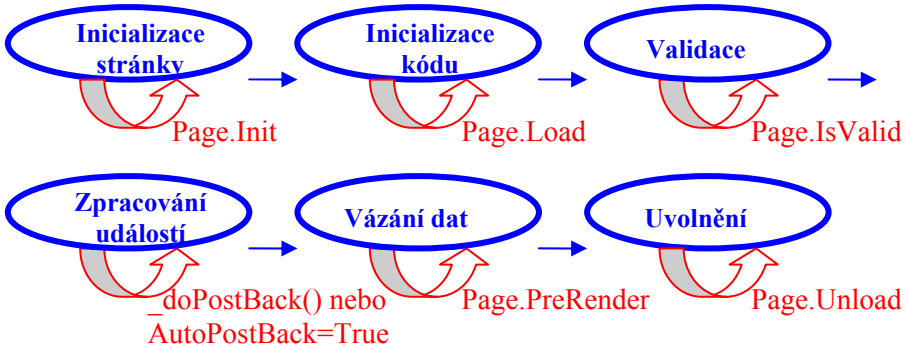
- popsat fáze zpracování webového formuláře
- demonstrovat zpracování událostí na příkladu
- rozdělit hierarchii HTML ovládacích prvků
- používat nástroj Toolbox pro design web formulářů
- popsat serverové webové ovládací prvky
- zkompletovat do skupin dostupné ovládací prvky a popsat jejich použití

Výklad

Visual studio 2005 nám nabízí bohatou škálu ovládacích prvků, pomocí nichž můžeme webové aplikace komfortně vytvořit a poskytnout tak uživateli opravdu příjemný vzhled aplikace. Většina ASP.NET stránek tyto ovládací prvky využívá v hojném počtu, a tak je důležité tyto prvky znát. Možnosti, které ovládací prvky nabízejí, by mohly zabrat mnohasetstránkovou publikaci, a tak si je představíme formou, která nás naučí uvedené prvky používat v konkrétních situacích. Tato kapitola nám ovládací prvky představí a v několika následujících kapitolách si ukážeme použití na častých a vhodných příkladech.

4.1 Zpracování formuláře

Ještě jednou se vraťme k příkladu z kapitoly 2, kdy jsme zapsali do textboxu jméno a odeslali jej na server stisknutím tlačítka. V kapitole 3.2 jsme si popsali princip zpracování události v ASP.NET. Protože toto téma je velice důležité a opakování je matka moudrosti, vytvořme web formulář, kde si uvedené skutečnosti procvičíme. Na základě životního cyklu stránky si vytvoříme příklad, který nám pomůže zobrazit pořadí jednotlivých událostí.



Obr. 4.1 Zpracování web formuláře

Vytvoříme si tedy jednoduchý příklad s textovým polem, tlačítkem a popiskem, do kterého budeme jednotlivé události stránky vepisovat. Cílem ukázkového příkladu je zobrazit sled událostí zpracování formuláře, včetně uživatelských, tak jak je nám zobrazuje obrázek 4.1.

Příklad – Zpracování stránky – příprava jednoduchého formuláře

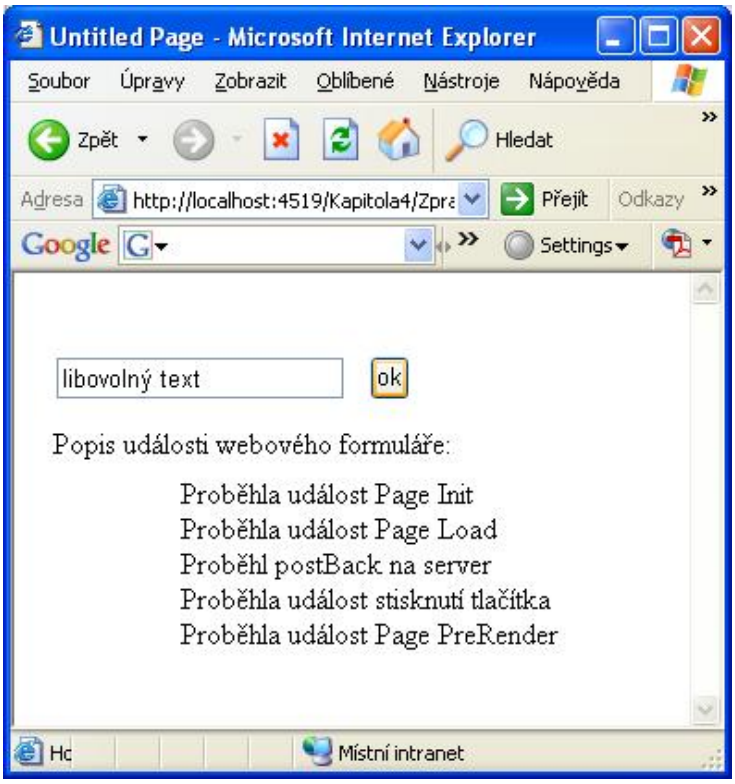
```
<form id="form1" runat="server">
  <asp:Label ID="Label1" runat="server" Text="Popis události
   webového formuláře:"></asp:Label>
  <asp:Label ID="lblUdalost" runat="server"
   EnableViewState="False"></asp:Label>
  <asp:TextBox ID="tbText" runat="server" OnTextChanged=
   "tbText_TextChanged" ></asp:TextBox>
  <asp:Button ID="btnTlacitko" runat="server" Text="ok"
   OnClick="btnClick" />
 </form>
```

Formulář tedy obsahuje dva popisky, textbox a tlačítko. Tlačítko obsahuje proceduru *btnClick*, kterou vyvolá stisk tlačítka. Za zmínku stojí ještě atribut `EnableViewState="False"` popisku, který zajistí zobrazení vždy posledního stavu popisku. Kdyby byl atribut nastaven na True, což je implicitní hodnota, popisek by díky obslužnému skriptu načítal další a další text při každém následujícím stavu, takto uvidíme pouze stav aktuální.
Zbývá dopsat skript, který se skládá z uživatelské procedury stisknutí tlačítka *btnClick*, změny obsahu textboxu *tbText_Changed* a událostí stránky *Init*, *Load* a *Prerender*. Validaci dat zatím do příkladu nezařadíme, neboť ji probereme až v následující kapitole a událost Page_Unload() uvažovat také nebudeme, protože by stejně data nezobrazila, neboť je provedena v okamžiku, když už je stránka vystavena v prohlížeči.

Příklad – Zpracování stránky – obslužný skript

```
<script runat="server">
Sub page_Load(ByVal obj As Object, ByVal e As EventArgs)
  lblUdalost.Text += "Proběhla událost Page Load <br/>"
     If Page.IsPostBack Then
        lblUdalost.Text += " Proběhl postBack na server <br/>"
     End If
End Sub

Sub page_Init(ByVal obj As Object, ByVal e As EventArgs)
        lblUdalost.Text += " Proběhla událost Page Init <br/>"
End Sub

Sub page_preRender(ByVal obj As Object, ByVal e As EventArgs)
        lblUdalost.Text += " Proběhla událost Page PreRender <br/>"
End Sub

Sub btnClick(ByVal obj As Object, ByVal e As EventArgs)
    lblUdalost.Text += " Proběhla událost stisknutí tlačítka <br/>"
End Sub

Sub tbText_TextChanged(ByVal sender As Object, ByVal e As EventArgs)
     lblUdalost.Text += " Proběhla událost TextChanged, neboť je
      změněn obsah textboxu <br/>"
End Sub
</script>
```

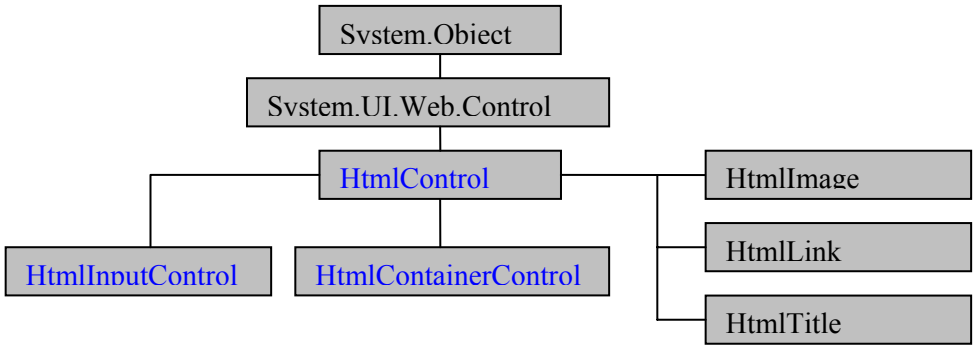
Zajímavou situaci dostaneme v případě, že dopíšeme atribut textboxu `AutoPostBack="true"`. V tomto případě dojde k postbacku na server i bez kliknutí na tlačítko. Doporučuji vyzkoušet příklad i s touto variantou a také se znovunastavením atributu `EnableViewState="True"` popisku.



Obr. 4.2 Průběh zpracování událostí ukázkového příkladu

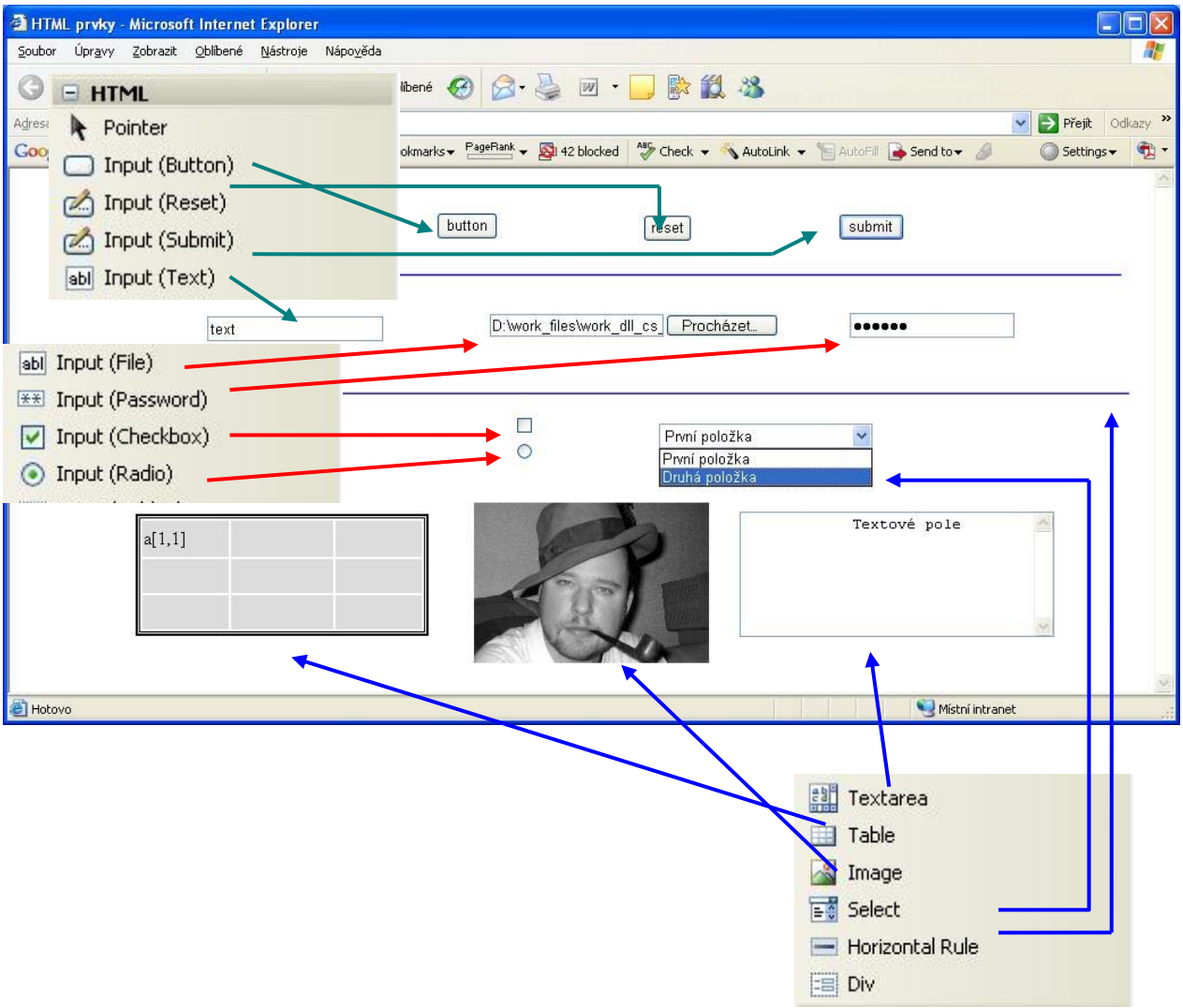
4.2 Ovládací prvky HTML

Tyto prvky dobře zná každý tvůrce HTML stránek, i těch základních. Stránky tvořené v ASP.NET s nimi samozřejmě musí počítat. Jedná se o klasické INPUT (vstupní) prvky typu textbox, tlačítko, zaškrtávací pole a další prvky jako textarea, obrázky a tabulky. Všechny ovládací prvky HTML jsou odvozeny ze základní třídy *HtmlControl*. Tato třída obsahuje vlastnosti *Attributes, Disabled, Style a TagName*. Vlastnost *Attributes* umožňuje přístup k atributům, *Disabled* umožňuje znepřístupnění prvku, které se projevuje zašednutím, *Style* umožňuje nastavit CSS styly na html prvek a *TagName* vrací název značky ovládacího prvku.
V hierarchii níže vidíme na obrázku třídu *HtmlInputControl*, která odvozuje všechny třídy již zmíněných prvků INPUT, např.`<input id="Chckbox1" type="checkbox" />`. Třída *HtmlInputControl* nabízí známé vlastnosti *Name*, *Type* a *Value*. Další HTML prvky jako select, textarea, table, anchor jsou odvozeny od třídy *HtmlContainerControl*, která nabízí dvě vlastnosti: *innerHtml* a *innerText* pro vrácení nebo nastavení textu. Další třídy jsou přímo odvozeny od *HtmlControl*, jedná se o Image, Link, Title (viz obr. 4.3).



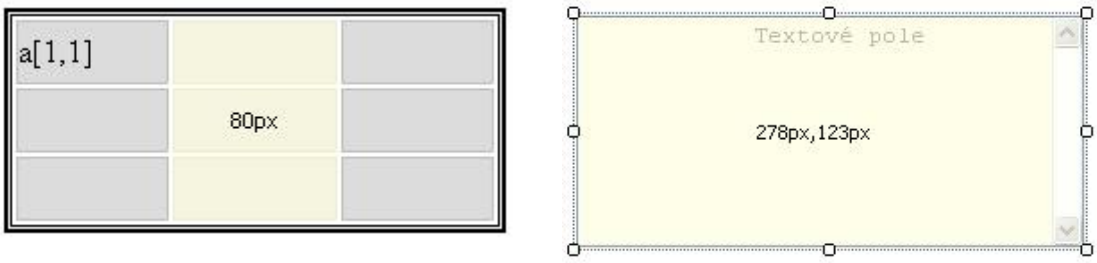
Obr. 4.3 Hierarchie tříd HTML ovládacích prvků

Na obrázku 4.4 vidíme ukázku aplikace se všemi dostupnými HTML prvky z Toolboxu design módu. Nelze opomenout, že serverovým ovládacím HTML prvkem odvozeným ze třídy *HtmlGenericControl* je jakákoliv značka HTML s atributem *runat="server"*.



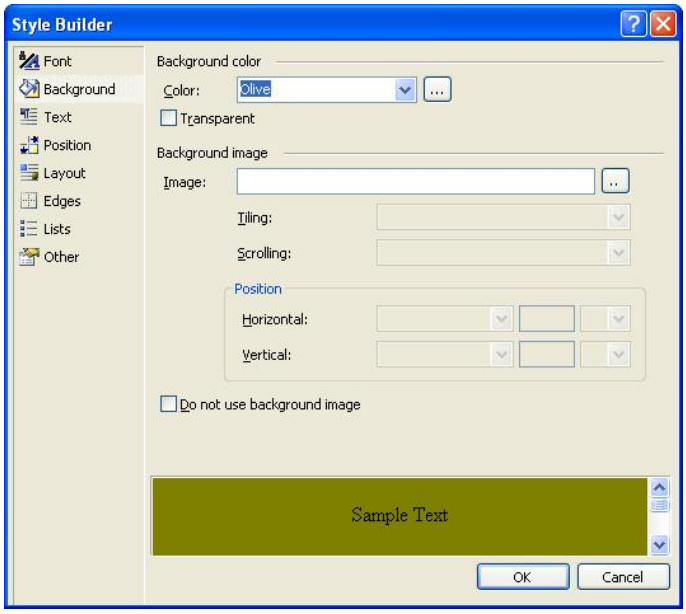
Obr. 4.4 ASP stránka s příkladem nabízených HTML ovládacích prvků

Visual Studio nabízí v design módu komfortní práci s ovládacími prvky. Při práci s myší můžeme uvedené prvky na jeho okrajích uchytit a velice rychle pak měnit jeho rozměry či polohu. Při práci s tabulkou vidíme na následujícím obrázku nastavení velikosti prostředního sloupce v pixelech pomocí myši.



Obr. 4.5 Práce s HTML prvky v design módu

Jednotlivé ovládací prvky také můžeme upravovat pomocí kaskádových stylů, při kliknutí na prvek pravým tlačítkem a volbě *style*. Zobrazí se *Style Buider* viz. následují obrázek, který umožňuje pohodlně nastavit vlastnosti stylu prvku.



Obr. 4.6 Práce s HTML prvky v design módu

4.3 Webové ovládací prvky

Ovládací prvky HTML jsou jistou alternativou, se kterou ovšem mnohdy nevystačíme. Proto je k dispozici vybudována celá základna webových ovládacích prvků, které nabízejí oproti HTML prvkům bohatší výbavu a funkcionalitu.
Všechny webové ovládací prvky jsou odvozeny od třídy *WebControl*. Ta je odvozena obdobně jako *HtmlControl* ze třídy *System.UI.Web.Control*, proto některé vlastnosti budou obdobné HTML prvkům. Třída *WebControl* však přidává celou řadu vlastností, které umožňují snadnou konfiguraci webového ovládacího prvku. Jedná se přímo o atributy barev (*BackColor*, *BorderColor*, *ForeColor*), parametry rozměrů (*BorderWidth*, *Height*, *Width*), fonty, styly a další.

Webové ovládací prvky si rozdělme do třech kategorií:

1. Standardní webové ovládací prvky – serverové ASP prvky obdobné k HTML prvkům (např. tlačítko, popisek, textbox, atd.).
2. Webové ovládací prvky pro seznamy – ASP prvky pro různé druhy seznamů, tzv. listů.
3. Speciální webové ovládací prvky – speciální ASP prvky jako např. kalendář.

Pojem k zapamatování: Webové ovládací prvky

Všechny webové ovládací prvky se deklarují značkou `<asp:prvek>`

Takže webový ovládací prvek tabulka nadefinujeme `<asp:table>` na rozdíl od HTML tabulky s elementem `<table>`

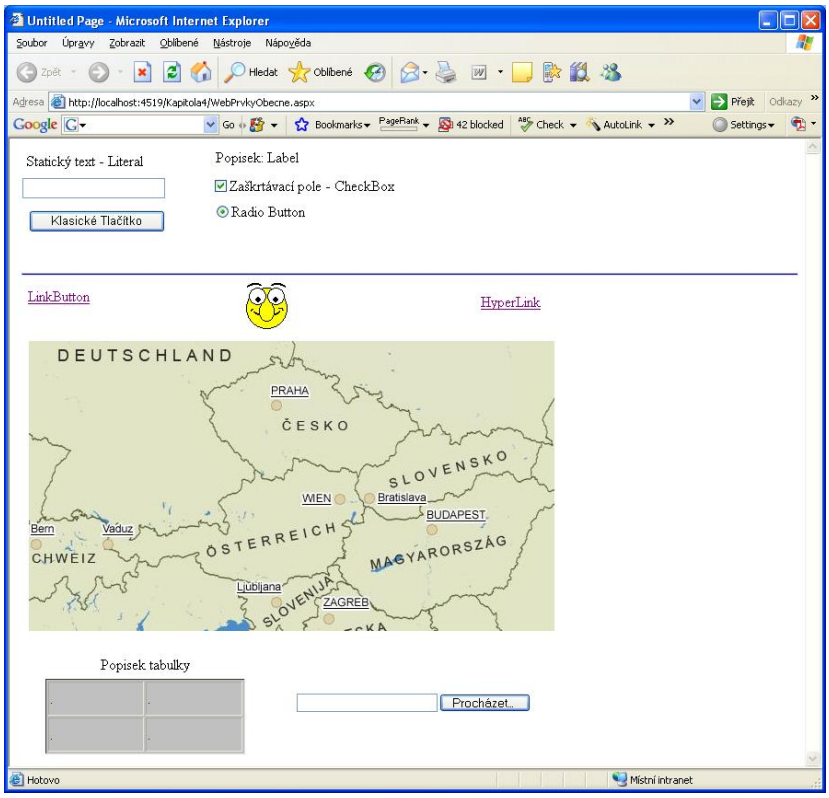
Všechny webové ovládací prvky jsou serverové, takže musí obsahovat povinný atribut `runat="server"`

Nyní si webové ovládací prvky představme. Nejprve uvedeme tabulku klasických webových prvků

s jejich HTML ekvivalentem a klíčovými členy:

ASP.NET značka	HTML ekvivalent	Klíčové členy
<asp:Buton>	<input type="buton">	Text, CausesValidation, PostBackUrl, ValidationGroup, Click
<asp:CheckBox>	<input type="checkbox">	AutoPostBack, Checked, Text, TextAlign, CheckedChanged
<asp:FileUpload>	<input type="file">	FileBytes, FileContent, FileName, HasFile, PostedFile, SaveAs()
<asp:HiddenField>	<input type="hidden">	Value
<asp:HyperLink>	<a>	ImageUrl, NavigateUrl, Target, Text
<asp:Image>	<img>	AlternateText, ImageAlign, ImageUrl
<asp:ImageButton>	<input type="image">	CausesValidation, ValidationGroup, Click
<asp:ImageMap>	<map>	HotSpotMode, HotSpots, AlternateText, ImageAlign, ImageUrl
<asp:Label>		Text, AssociatedControlID
<asp:LinkButton>	<a><img>	Text, CausesValidation, ValidationGroup, Click
<asp:Panel>		BackImageUrl, DefaultButton, GroupingText, HorizontalAlign, Scrollbars, Wrap
<asp:RadioButton>	<input type="radio">	AutoPostBack, Checked, GroupName, Text, Textalign, CheckedChanged
<asp:Table>	<table>	BackImageURL, CellPadding, CellSpacing, GridLines, HorizontalAlign, Cells
<asp:TableCell> <asp:TableRow>	<td> <tr>	ColumnSpan, HorizontalAlign, RowSpan, Text, VerticalAlign, Wrap, Cells
<asp:TextBox>	<input type="text"> <textarea>	AutoPostBack, Columns, MaxLength, ReadOnly, Rows, Text, TextMode, Wrap, TextChanged

Tab. 4.1 Standardní webové ovládací prvky



Obr. 4.7 Standardní ASP webové ovládací prvky

Pojem k zapamatování: Události webových ovládacích prvků

Webové ovládací prvky narozdíl od prvků HTML mají schopnost automatického odeslání zpět na server (AutoPostBack).

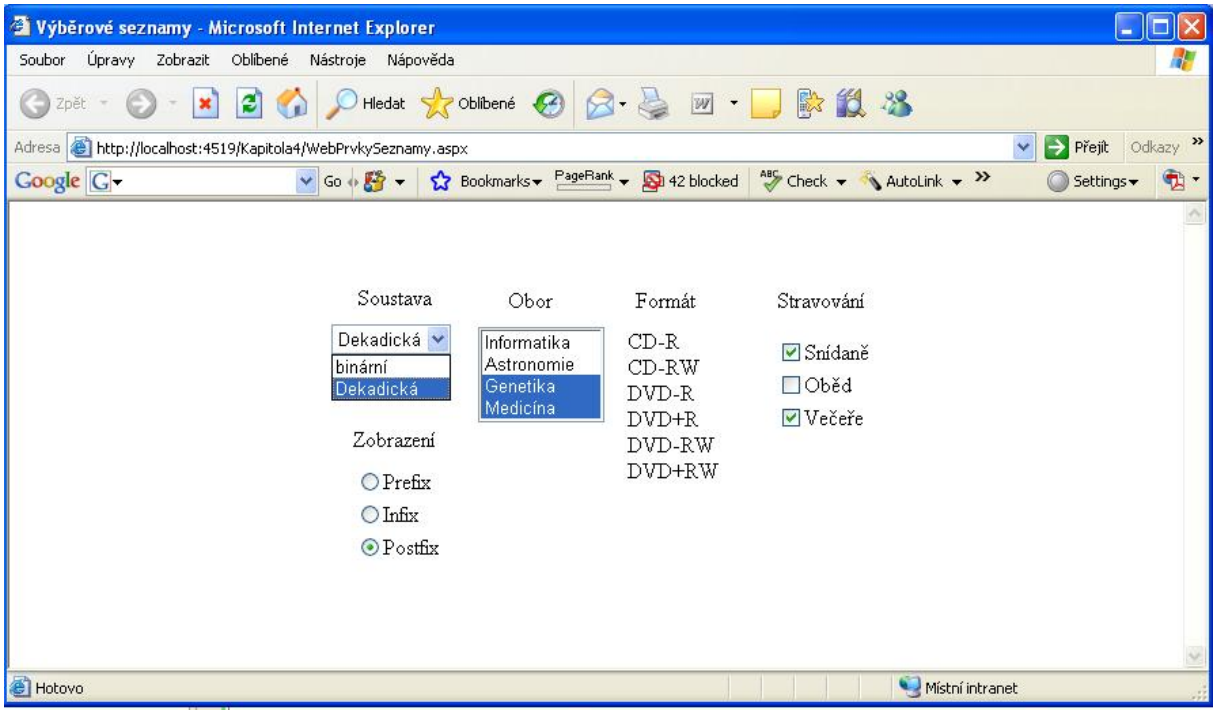
Druhou skupinou webových ovládacích prvků jsou prvky pro seznamy. Podporují stejnou kolekci vlastností jako ostatní prvky, navíc však obsahují vlastnosti a metody zděděné ze třídy *ListControl*.

Ovládací prvek	Charakteristika
<asp:ListBox>	Otevřený seznam s kolekcí prvků <asp:listItem>
<asp:DropDownList>	Rozbalovací seznam s kolekcí prvků <asp:listItem>
<asp:RadioButtonList>	Seznam s přepínači
<asp:CheckBoxList>	Seznam se zaškrtávacími poli
<asp:BulletedList>	Odrážkový či číselný seznam s HTML ekvivalenty

Tab. 4.2 Webové ovládací prvky pro seznamy

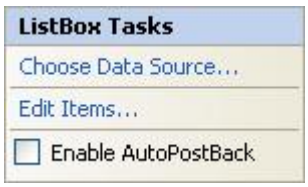
Vlastnost	Charakteristika
AutoPostBack	Formulář je automaticky odeslán při změně prvku na server, pokud je atribut nastaven na hodnotu True
Items	Vrací kolekci položek ListItem
SelectedIndex	Vrací nebo nastavuje index zvolené položky
SelectedItem	Vrací první vybranou položku
DataSource	Nastavení datového zdroje pro zobrazení
DataMember	Datový člen při více tabulek datového zdroje
DataTextField	Zdroj dat pro položky seznamu
DataValueField	Zdroj dat pro atribut položek seznamu
DataTextFormatString	Nastavení formátovacího řetězce

Tab. 4.3 Vlastnosti třídy ListControl



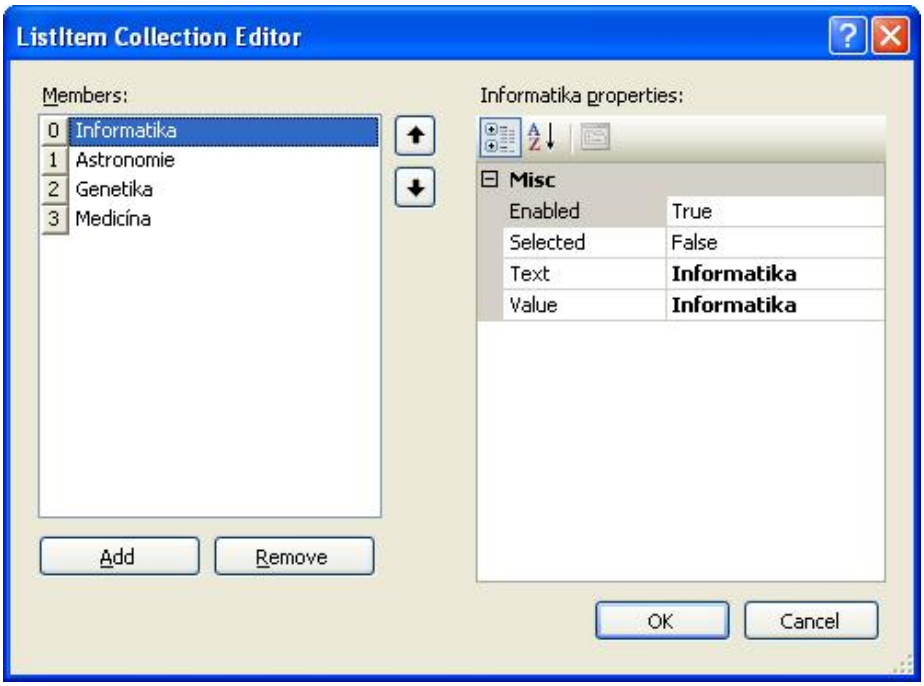
Obr. 4.8 Ukázka webových ovládacích prvků pro seznamy

Na obrázku 4.8 je ukázka všech pěti typů ovládacích prvků pro seznamy. Při prostudování příkladu na CD je vhodné si všimnout, že některé situace vyžadují možnost pouze jedné volby, zatímco některé volbu vícenásobnou. Pro tento případ zvolíme nastavení atributu příslušného prvku, např. pro ListBox `SelectionMode="Multiple"` nebo `"Single"`.
Visual Studio nabízí v design módu rychlé nastavení funkčnosti ovládacích prvků, tzv. T*asks*. Tato možnost je přístupná v pravém horním rohu při editaci prvku. Na následujícím obrázku vidíme *Tasks* ovládacího prvku ListBox.



Obr. 4.9 Rychlé nastavení funkčnosti Listboxu

Tímto způsobem se můžeme ihned pustit do uživatelsky přívětivého nastavení. Můžeme zvolit přímo datový zdroj pro ovládací prvek (Datové zdroje využijeme v dalších kapitolách) nebo můžeme přímo vytvářet položky v nabízeném editoru (viz. obr. 4.10). Pokud dáváme přednost psaní zdrojového kódu, můžeme vše vyřešit i tímto způsobem za použití všudypřítomného nástroje *IntelliSense*.



Obr. 4.10 Editor položek listboxu

```
<asp:ListBox ID="ListBox1" runat="server"
    Style="z-index: 101; left: 125px; position: absolute; top: 41px" s >
    <asp:ListItem>Informatika</asp:ListItem>
    <asp:ListItem>Astronomie</asp:ListItem>
    <asp:ListItem>Genetika</asp:ListItem>
    <asp:ListItem>Medicína</asp:ListItem>

</asp:ListBox>
    <asp:Label ID="Label1" runat="server" Style="z-index: 102; lef
        top: 114px" Text="Zobrazení"></asp:Label>
    <asp:Label ID="Label2" runat="server" Style="z-index: 103; lef
        top: 12px" Text="Obor"></asp:Label>
```

OnTextChanged
OnUnload
Rows
runat
SelectionMode
SkinID
TabIndex
ToolTip
ValidationGroup
Visible

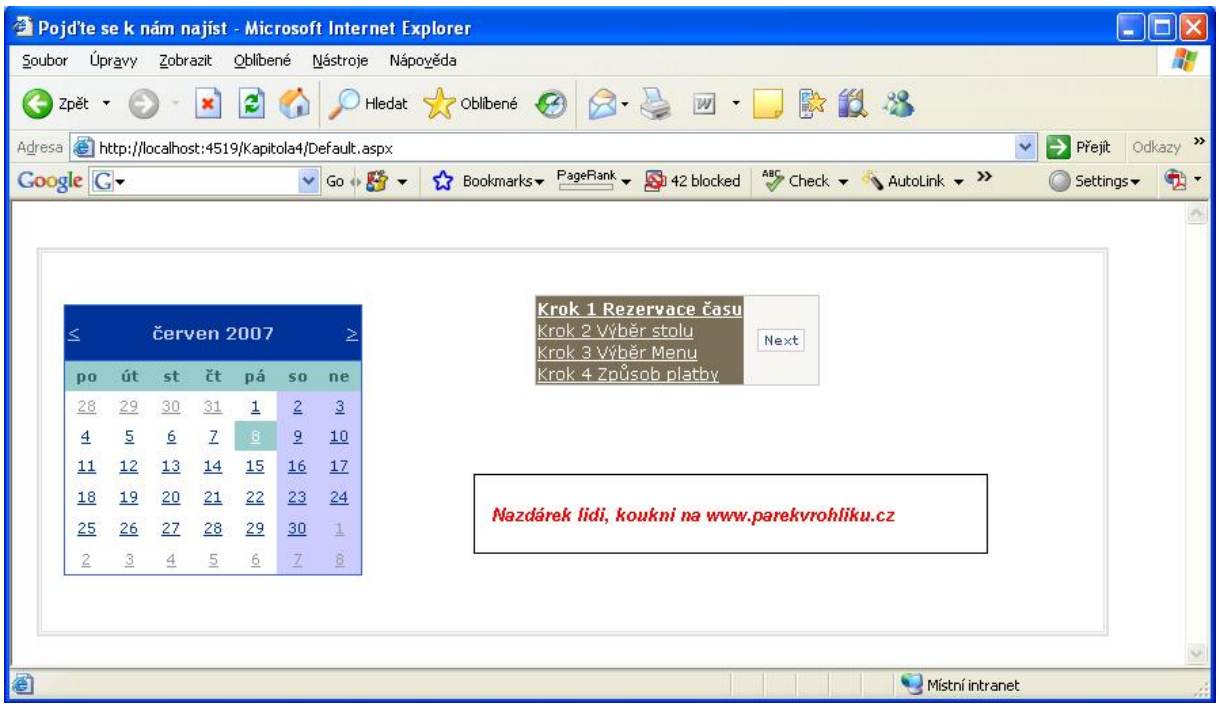
Obr 4.11 Nastavení atributů listboxu přímo ve zdrojovém kódu pomocí IntelliSense

Třetí skupinou webových ovládacích prvků jsou speciální prvky. Tyto prvky poskytují složitější uživatelská rozhraní a vývojáři webu dají jistě za pravdu, že dříve tyto funkcionality bylo nutné složitě programovat. Mezi takové prvky patří jistě kalendář, který umožňuje libovolný pohyb po dnech či měsících, AdRotator, který reprezentuje reklamní banner z několika připravených obrázků, jež jsou definovány v XML souboru, různé vyspělé pohledy a panely, které umožňují přepínání celých skupin ovládacích prvků a další.

Korespondenční úkol – XML soubor

Nejen adRotator jako zdroj dat využívá XML soubor, tento formát reprezentace dat je pro ASP.NET velice důležitý. Zopakujte si základní pravidla pro tvorbu XML souboru.

V demonstračním příkladu vidíme také ovládací prvek Wizard, který již v sobě obsahuje navigační logiku, která uživatele vede po jednotlivých krocích. Tyto prvky jsou ve většině případů novinkami v ASP.NET verze 2.0 a bude jim věnována samostatná kapitola.



Obr. 4.12 Příklad speciálních webových ovládacích prvků Calendar, adRotator a Wizard

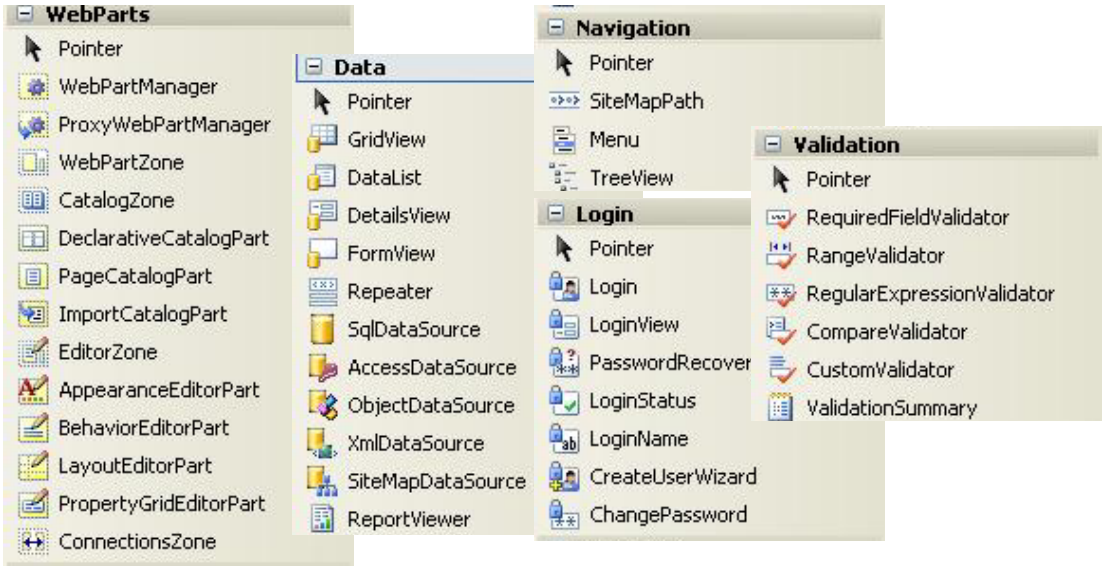
Ke kapitole 4 je přiřazena demonstrační animace č. 3

Animace č. 3 obsahuje příklad zpracování událostí stránky a také práci s webovými ovládacími prvky na příkladech, které budou demonstrovány v následující kapitole. Proto je vhodné animaci číslo 3 spustit až po přečtení následující kapitoly.

4.4 Další skupiny ovládacích prvků

V této kapitole jsme si představili postupně HTML prvky a standardní prvky. Těch je samozřejmě největší množství a jsou používány nejčastěji. Web aplikace v ASP.NET však obsahují ještě daleko bohatější paletu nástrojů, některé z nich byly přidány až pro verzi 2.0. Jedná se především o skupiny WebParts, Login a Navigation, jež svou funkčnost charakterizuje již samotný název. Ihned po kapitole věnované standardním ovládacím prvkům se budeme věnovat validaci formulářů, která

s těmito prvky úzce souvisí. V dalších kapitolách pro změnu využijeme bohatou paletu ovládacích prvků pro práci s daty. Webové ovládací prvky jsou tedy denní náplní vývojáře stránek a proto je vhodné tyto prvky dobře ovládat. Zmíněné skupiny prvků vidíme na obr. 4.13, tímto obrázkem jsme završili výčet všech prvků toolboxu.



Obr. 4.13 Další skupiny ovládacích prvků pro ASP.NET aplikace

Shrnutí kapitoly

Zpracování formuláře probíhá v tomto sledu: 1. Inicializace stránky - 2. Inicializace kódu - 3. Validace dat - 4. Zpracování událostí - 5. Vázaní dat - 6. Uvolnění zdrojů.

Ovládací prvky HTML dobře zná každý tvůrce HTML stránek, i těch základních. Jedná se o klasické INPUT (vstupní) prvky typu textbox, tlačítko, zaškrtávací pole a další prvky jako textarea, obrázky a tabulky. Všechny ovládací prvky HTML jsou odvozeny ze základní třídy *HtmlControl*. Tato třída obsahuje vlastnosti *Attributes, Disabled, Style* a *TagName*. Vlastnost *Attributes* umožňuje přístup k atributům, *Disabled* umožňuje znepřístupnění prvku, které se projevuje zašednutím, *Style* umožňuje nastavit CSS styly na html prvek a *TagName* vrací název značky ovládacího prvku.

V hierarchii níže figuruje třída *HtmlInputControl*, která odvozuje všechny třídy již zmíněných prvků INPUT, např:<input id="Chckbox1" type="checkbox" />. Třída *HtmlInputControl* nabízí známé vlastnosti *Name*, *Type* a *Value*. Další HTML prvky jako select, textarea, table, anchor jsou odvozeny od třídy *HtmlContainerControl*, která nabízí dvě vlastnosti: *innerHtml* a *innerText* pro vrácení nebo nastavení textu. Další třídy jsou přímo odvozeny od *HtmlControl*, jedná se o Image, Link a Title.
Visual Studio nabízí v design módu komfortní práci s ovládacími prvky. Při práci s myší můžeme uvedené prvky na jeho okrajích uchytit a velice rychle pak měnit jeho rozměry či polohu.

Jednotlivé ovládací prvky také můžeme upravovat pomocí kaskádových stylů, při kliknutí na prvek pravým tlačítkem a volbě *Style*. Zobrazí se *Style Builder*, který umožňuje pohodlně nastavit vlastnosti stylu prvku. Ovládací prvky HTML jsou jistou alternativou, se kterou ovšem mnohdy nevystačíme. Proto je k dispozici vybudována celá základna webových ovládacích prvků, které nabízejí oproti HTML prvkům bohatší výbavu a funkcionalitu.

Všechny webové ovládací prvky jsou odvozeny od třídy *WebControl*. Ta je odvozena

obdobně jako *HtmlControl* ze třídy *System.UI.Web.Control*, proto některé vlastnosti budou obdobné HTML prvkům. Třída *WebControl* však přidává celou řadu vlastností, které umožňují snadnou konfiguraci webového ovládacího prvku. Jedná se přímo o atributy barev (*BackColor, BorderColor, ForeColor*), parametry rozměrů (*BorderWidth, Height, Width*), fonty, styly a další. Webové ovládací prvky si rozdělme do třech kategorií:
1. Standardní webové ovládací prvky – serverové ASP prvky obdobné k HTML prvkům (např. tlačítko, popisek, textbox, atd.).
2. Webové ovládací prvky pro seznamy – ASP prvky pro různé druhy seznamů, tzv. listů.
3. Speciální webové ovládací prvky – speciální ASP prvky jako např. kalendář.
Všechny webové ovládací prvky se deklarují značkou <asp:prvek>. Webové ovládací prvky, narozdíl od prvků HTML mají schopnost automatického odeslání zpět na server (AutoPostBack).

Úkol k řešení 4.1 – Návrh formuláře

Navrhněte vlastní formulář, ve kterém využijete různé druhy ovládacích prvků. Obsluhou událostí se zatím nezabývejte. Důležité je procvičit tvorbu formuláře s množstvím ovládacích prvků. Jednotlivé prvky upravte do vhodné grafické podoby, využijte vlastnosti ovládacích prvků, *StyleBuilder* a skutečnosti popsané v této kapitole.

Kontrolní otázka 4.1

Popište životní cyklus stránky.

Kontrolní otázka 4.2

Co znamená vlastnost AutoPostBack?

Kontrolní otázka 4.3

Jak byste rozdělili ovládací prvky v ASP.NET?

Kontrolní otázka 4.4

Jaký je rozdíl mezi webovými ovládacími prvky a HTML prvky?

Kontrolní otázka 4.5

Jakým způsobem vložíme položky do ovládacího prvku ListBox?

5. SERVEROVÉ OVLÁDACÍ PRVKY

Čas ke studiu: 3 hodiny

Cíl Po prostudování této kapitoly budete umět

- nastavit vlastnosti serverových ovládacích prvků
- vysvětlit princip metody focus()
- orientovat se ve vývoji serverových ovládacích prvků
- využít rozmanitou paletu ovládacích prvků
- vytvořit rozsáhlý formulář obsahující serverové ovládací prvky
- obsloužit události serverových ovládacích prvků pomocí skriptu

Výklad

V minulé kapitole jsme si rozdělili do skupin bohatou sadu ovládacích prvků. Tato kapitola má za úkol serverové ovládací prvky ještě více přiblížit a to na konkrétních situacích. Pomocí řešených příkladů si představíme serverové ovládací prvky a jejich vlastnosti. Uvedené příklady procvičíme také v přiložené animaci.

5.1 Vlastnosti serverových ovládacích prvků

Serverové ovládací prvky mají tvar ***<asp:NázevPrvku runat="server" seznamAtributů/>***. Typy a hodnoty atributů jsou závislé na funkčnosti ovládacího prvku. Ke každému prvku je přiřazena množina vlastností, kterou zjistíme podle *Intellisense*. Všechny vlastnosti ovládacích prvků, které definujeme podle nějakých hodnot, máme možnost zadávat relativně nebo absolutně. Relativní definice se vztahuje k velikosti aktuálního okna prohlížeče a zadává se v procentech. Absolutní velikosti zadáváme v pixelech a zapisujeme jej zkratkou px:

<asp:prvek runat="server"Height="250px" Width="10%" />

Známou vlastností nejen ovládacích prvků je barva. Barvy zajišťuje objekt *Color* ze jmenného prostoru System.Drawing. Barvy definujeme známým způsobem ve formátu *#<Red><Green><Blue>*. Takže například zelenou barvu nastavíme hodnotou *#00FF00*. Druhou možností je použití předdefinovaného názvu barev .NET ze třídy *Color*. Obdobným způsobem můžeme zadat název HTML barvy, tu specifikujeme jako řetězec pomocí třídy *ColorTranslator*. Posledním způsobem je možnost vytvořit barvu pomocí RGB ještě s hodnotou alfa kanálu, tzv. ARGB.

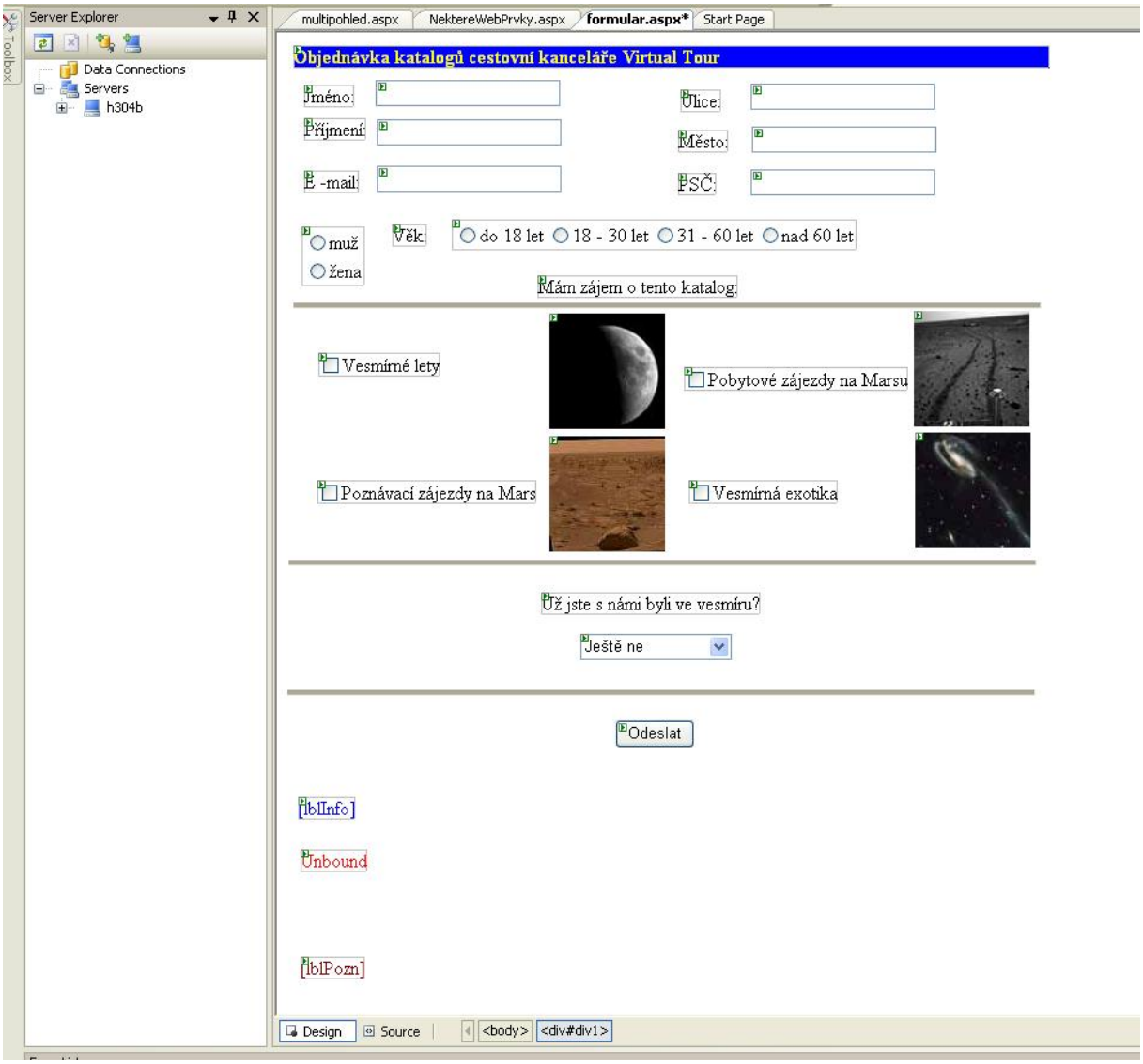
Pojem k zapamatování: Metoda Focus()

Novinkou v ASP.NET 2.0 je metoda Focus(). U ovládacích prvků, kde je vyžadován vstup z klávesnice, můžeme nastavit *focus*, což je vlastnost, která nám umožňuje fakt, že po načtení formuláře do prohlížeče se začne pracovat jako první s tímto ovládacím prvkem.

Metodu focus nelze uplatnit u HTML ovládacích prvků a z dřívějších dob si pamatujeme, že jsme tuto vymoženost museli programovat javascriptem.

5.2 Příklad tvorby formuláře se serverovými ovládacími prvky

Serverové ovládací prvky se naučíme nejlépe na příkladech, které si procvičíme. Bez dlouhého úvodu pojďme na první příklad. Zadáním bude vytvořit objednávkový formulář cestovní kanceláře, který slouží k volbě katalogů pro zaslání poštou. Budeme potřebovat zjistit několik základních údajů o klientovi jako jméno a příjmení, e-mail a adresu, kam mu má být zaslán katalog. Tyto údaje budou získány z textboxů. Samozřejmě budeme na stránku umísťovat popisky (*Label*) a obrázky (*Image*). Poté budeme potřebovat získat ještě několik údajů z více možností, proto použijeme *RadioButtonList* a *DropDownList*. Rovněž pro výběr katalogů potřebujeme ovládací prvky, zvolíme zaškrtávací tlačítka a na závěr pro potvrzení nezbytné tlačítko. Pro některá informativní hlášení budeme definovat ještě několik popisků. Výsledný formulář, který si vytvoříme, je zobrazen na obrázku 5.1.



Obr. 5.1 Příklad formuláře pro objednávání katalogů

Začneme tedy vytvářet navržený formulář. Můžeme psát velice rychle za pomocí IntelliSense zdrojový kód, nebo můžeme přetahovat ovládací prvky z *Toolboxu* a definovat jejich vlastnosti v okně *Properties*. Nejprve tedy vytvoříme úvodní popisek cestovní kanceláře a popisky s textboxy pro získání údajů od klienta:

Příklad – Příprava formuláře pro stránku objednávky

```
<form id="form1" runat="server">

<asp:Label ID="lblObj" runat="server" Text="Objednávka katalogů
cestovní kanceláře Virtual Tour" Width="635px"></asp:Label>

<asp:Label ID="lblJmeno" runat="server" Text="Jméno:"></asp:Label>
<asp:Label ID="lblPrijmeni" runat="server" Text="Příjmení:">
</asp:Label>
<asp:Label ID="lblEmail" runat="server" Text="E -mail:"></asp:Label>
<asp:Label ID="lblUlice" runat="server" Text="Ulice:"></asp:Label>
<asp:Label ID="lblMesto" runat="server" Text="Město:"></asp:Label>
<asp:Label ID="lblPsc" runat="server" Text="PSČ:"></asp:Label>

<asp:TextBox ID="tbJmeno" runat="server" ></asp:TextBox>
<asp:TextBox ID="tbPrijmeni" runat="server" </asp:TextBox>
<asp:TextBox ID="tbEmail" runat="server" ></asp:TextBox>
<asp:TextBox ID="tbUlice" runat="server" </asp:TextBox>
<asp:TextBox ID="tbMesto" runat="server" ></asp:TextBox>
<asp:TextBox ID="tbPSC" runat="server" </asp:TextBox>
```

Při tvorbě většího formuláře jako je tento musíme zavést nějakou štábní kulturu pro názvy jednotlivých prvků, abychom nevytvořili nejasnosti a vyhnuli se pozdějším problémům. Budeme se držet našeho zavedeného pravidla z úvodu učebního textu, kde jsme si dali za úkol pojmenovávat ID ovládacích prvků prefixem jejich typu a názvem jejich významu (např. *tbMesto* pro ovládací prvek určující město).

Úvodní popisky a textboxy máme za sebou, pojďme vytvořit seznamy, které slouží pro výběr jedné hodnoty. K tomuto výběru se nejlépe hodí *RadioButtonList*. Použijeme jej dvakrát, pro výběr pohlaví a pro určení věku klienta. Zdrojový kód ještě obsahuje několik jednoduchých popisků.

Příklad – Příprava formuláře pro stránku objednávky – pokračování 1

```
<asp:RadioButtonList ID="rblPohlavi" runat="server">
      <asp:ListItem>muž</asp:ListItem>
      <asp:ListItem>žena</asp:ListItem>
</asp:RadioButtonList>

<asp:Label ID="lblVek" runat="server" Text="Věk:"></asp:Label>

<asp:RadioButtonList ID="rblVek" runat="server" RepeatDirection=
"Horizontal">
    <asp:ListItem>do 18 let</asp:ListItem>
    <asp:ListItem>18 - 30 let</asp:ListItem>
    <asp:ListItem>31 - 60 let</asp:ListItem>
    <asp:ListItem>nad 60 let</asp:ListItem>
</asp:RadioButtonList>

<asp:Label ID="lblZajem" runat="server"Text="Mám zájem o tento
katalog:"></asp:Label>

<hr id="hr1" />
```

Nyní definujme zaškrtávací pole pro výběr katalogů a jejich obrázkové zobrazení. Zaškrtávací pole volíme proto, že uživatel může zvolit více variant, tyto prvky na sobě nejsou závislé tak jako v listu,

kde volíme pouze jednu z několika variant. Obrázky jsou uloženy v aktuálním projektu ve složce images.

Příklad – Příprava formuláře pro stránku objednávky – pokračování 2

```
<asp:CheckBox ID="chbExot" runat="server" Text="Vesmírná exotika" />
<asp:CheckBox ID="chbPobyt" runat="server" Text="Pobytové zájezdy na
Marsu" />
<asp:CheckBox ID="chbPozn" runat="server" Text="Poznávací zájezdy na
Mars" />
<asp:CheckBox ID="chbVesmir" runat="server" Text="Vesmírné lety" />

<asp:Image ID="Img1" runat="server" ImageUrl="~/images/01.jpg" />
<asp:Image ID="Img2" runat="server" ImageUrl="~/images/02.jpg" />
<asp:Image ID="Img3" runat="server" ImageUrl="~/images/03.jpg" />
<asp:Image ID="Img4" runat="server" ImageUrl="~/images/04.jpg" />

<hr id="Hr2" />
```

Dále musíme definovat *DropDownList* a odesílací tlačítko. Tím celý formulář odešleme na server, takže na událost *OnClick* definujme proceduru *btnOdeslano*:

Příklad – Příprava formuláře pro stránku objednávky – pokračování 3

```
<asp:Label ID="lblOtazka" runat="server" Text="Už jste s námi byli
   ve vesmíru?"></asp:Label>

<asp:DropDownList ID="ddlPocet" runat="server">
     <asp:ListItem>Ještě ne</asp:ListItem>
     <asp:ListItem>Jednou či dvakrát</asp:ListItem>
     <asp:ListItem>Vícekrát</asp:ListItem>
</asp:DropDownList>

<hr id="Hr3"/>

<asp:Button ID="btOdeslat" runat="server" Text="Odeslat"
OnClick="btnOdeslano" />
```

Poslední ovládací prvky, které zbývají, jsou jednoduché. Jedná se o popisky, do kterých umístíme hlášení uživateli na základě vyhodnocení formuláře a *BulletList*, do kterého zobrazíme seznam katalogů, které si objednal.

Příklad – Příprava formuláře pro stránku objednávky – pokračování 4

```
<asp:Label ID="lblInfo" runat="server" ForeColor="Blue"></asp:Label>
<asp:BulletedList ID="blSeznam" runat="server"
EnableViewState="False">
</asp:BulletedList>
<asp:Label ID="lblPozn" runat="server" EnableViewState="False"
</asp:Label>

</form>
```

Všimněte si u ovládacích prvků atributu *EnableViewState*, který je nastaven na hodnotu *false*. Ten je zde nastaven pro náš případ, kdy budeme při testování příkladu formulář odesílat zřejmě několikrát, a nechceme tudíž při novém vyplnění formuláře zachovat předchozí hodnotu ovládacího prvku.

Formulář máme nyní, co se týká návrhové části hotov. Přejdeme tedy k obsluze jednotlivých ovládacích prvků podle potřeby naší aplikace. Nutno zdůraznit, že náš příklad je pouze demonstrační a proto reakce na vyplněný formulář po odeslání tlačítkem budeme vyplňovat do připravených popisků stejné stránky. V praxi bychom pravděpodobně otevřeli stránku jinou, kde bychom pokračovali v aplikaci.

Příklad – Zpracování stránky – obsluha ovládacích prvků 1

```
<script runat="server" >
    Sub btnOdeslano(ByVal obj As Object, ByVal e As EventArgs)
        Dim Osloveni As String
        If rblPohlavi.SelectedIndex = 0 Then
            Osloveni = " pane "
        Else
            Osloveni = " paní "
        End If
      blInfo.Text = "Dobrý den " & Osloveni & tbJmeno.Text & " " &
      tbPrijmeni.Text & ". Děkujeme Vám za Vaši objednávku. <br/>
      Následující katalogy Vám budou zaslány poštou:"
     End Sub
```

Nejprve zařídíme oslovení. Podle hodnoty *RadioButtonListu* zjistíme pohlaví uživatele, abychom jej správně pozdravili. Jméno a příjmení získáme z vyplněných textboxů.

Příklad – Zpracování stránky – obsluha ovládacích prvků 2

```
If chbVesmir.Checked Then blSeznam.Items.Add("Vesmírné lety")
 If chbPozn.Checked Then blSeznam.Items.Add("Poznávací zájezdy na
Mars")
If chbPobyt.Checked Then blSeznam.Items.Add("Pobytové zájezdy na
Marsu")
If chbExot.Checked Then blSeznam.Items.Add("Vesmírná exotika")
```

Hlavní částí je získat informace od uživatele, které katalogy si objednal. Opět připomínáme, že se jedná o ilustrativní příklad. V praxi bychom tyto hodnoty museli uložit do nějaké proměnné či databáze, v našem příkladě je pouze vypisujeme. Takže tato část skriptu plní vypisovaný seznam jako položky ovládacího prvku *BulletList*. Položky jsou naplněny v případě zaškrtnutí příslušného zaškrtávacího pole.

Příklad – Zpracování stránky – obsluha ovládacích prvků 3

```
If (rblVek.SelectedIndex = 0 Or rblVek.SelectedIndex = 3) Then
    lblPozn.Text = "Upozorňujeme na nutnost zdravotní prohlídky. "
End If
```

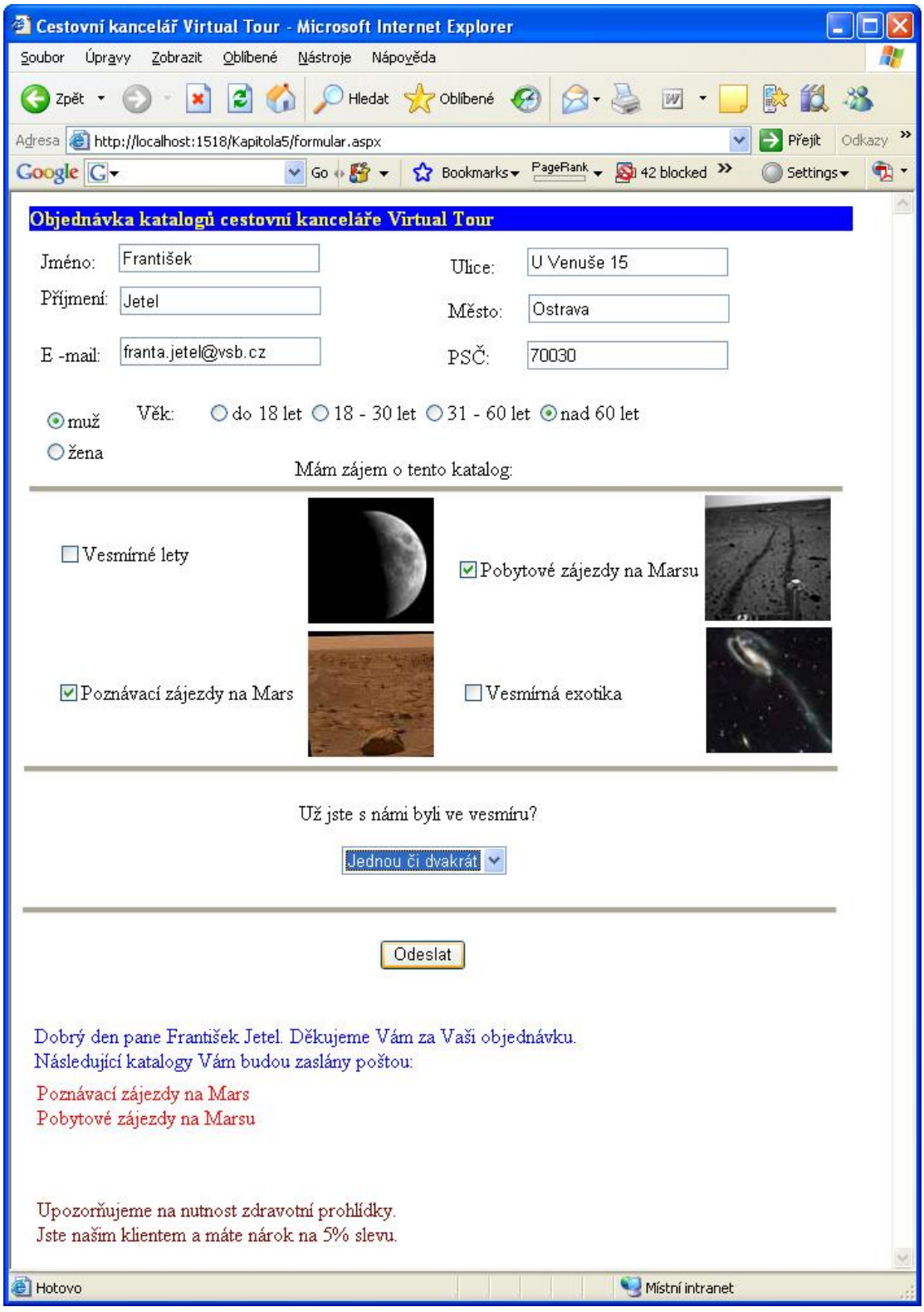
Požadovaný údaj o věku klienta není žádným diskriminačním prvkem, ale pouze informativním hlášením pro určité věkové kategorie o nutnosti zdravotní prohlídky. Tato nutnost platí pro věkové kategorie, které jsou v RadioButtonListu na prvním a posledním místě, tedy pro indexy 0 a 3.

Příklad – Zpracování stránky – obsluha ovládacích prvků 4

```
If ddlPocet.SelectedIndex = 1 Then
   lblPozn.Text += " <br/> Jste našim klientem a máte nárok na 5%
    slevu."
ElseIf ddlPocet.SelectedIndex = 2 Then
```

```
        lblPozn.Text += " <br/> Jste našim váženým klientem a máte nárok
        na 15% slevu."
    End If
```

Poslední část obsluhy skriptu se týká *DropDownListu* informujícího o slevě pro zákazníky. Ta je různá a je závislá na tom, kolikrát již klient využil služby cestovní kanceláře. Tomu odpovídá opět hodnota indexu, tentokrát v *DropDownListu*. Formulář i skript je hotov. Nastal čas otestovat aplikaci:



Obr. 5.2 Výsledná aplikace s formulářem pro objednávání katalogů

5.3 Některé další serverové ovládací prvky

V předchozím příkladu jsme si ukázali práci s ovládacími prvky, se kterými se pravděpodobně budete setkávat nejčastěji. Na obrázku 5.3 vidíme ukázku malé aplikace s ovládacími prvky `<asp:LinkButton>`, `<asp:HyperLink>` a `<asp:ImageButton>`.

Obr. 5.3 Aplikace s dalšími serverovými ovládacími prvky

Aplikace je jednoduchá a demonstruje různé druhy linků, kliknutím na *HyperLink* aplikaci přesměrujeme na předchozí bohatě vybavený formulář, kliknutím na *LinkButton* vypíšeme nějakou informaci do textu a kliknutím na *ImageButton*y budeme zobrazovat text do připraveného popisku katalogů pod obrázky. Nejprve si prohlídneme zdrojový kód formuláře:

Příklad – Příprava jednoduchého formuláře pro další ovládací prvky

```
<form id="form1" runat="server">

<asp:Label ID="Label1" runat="server" Text="Některé další webové
prvky"></asp:Label>
```

```
<asp:Panel ID="Panel1" runat="server" >
<asp:LinkButton ID="LinkButton1" runat="server"
OnClick="LinkButton1_Click">Tohle je LinkButton</asp:LinkButton>

<asp:HyperLink ID="HyperLink1" runat="server"
NavigateUrl="~/formular.aspx">Objednejte si zdarma katalog
!!!</asp:HyperLink>

<asp:ImageButton ID="imgbtMars" runat="server"
AlternateText="Zajistíme Vám misi na Mars"
ImageUrl="~/images/mars2.jpg" OnClick="ImageButton1_Click" />

<asp:ImageButton ID="ImageButton1" runat="server"
ImageUrl="~/images/04.jpg" OnClick="ImageButton1_Click1"  />

<asp:Label ID="lblInfo" runat="server"
EnableViewState="False"></asp:Label>

<asp:Label ID="lblTour" runat="server" </asp:Label>

</asp:Panel>
</form>
```

Všimněme si, že máme definované procedury na událost *OnClick* u *ImageButtonu* a *LinkButtonu*. Přesto, že jsou to linky, jedná se v pravém smyslu o tlačítka. Na rozdíl od nich má ovládací prvek *HyperLink* atribut *NavigateUrl*, kde zapíšeme URL stránky, na kterou chceme přejít.

Podíváme-li se do skriptu, máme nadefinovány tři procedury pro stisk jednotlivých *ImageButtonů* a *LinkButtonu*. Ve všech případech měníme textový obsah popisku.

Příklad – Zpracování stránky – příprava jednoduchého formuláře

```
<script runat="server">

Sub LinkButton1_Click(obj As Object, e As EventArgs)
    lblInfo.Text = " Klikli jste na LinkButton"
End Sub

Sub ImageButton1_Click(obj As Object, e As ImageClickEventArgs)
    lblTour.Text = " Zajistíme pro Vás dovolenou na Marsu dle Vašich
    představ"
End Sub

Sub ImageButton1_Click1(obj As Object, e As ImageClickEventArgs)
    lblTour.Text = " Můžete se podívat kamkoliv v galaxii !"
End Sub
</script>
```

5.4 Ovládací prvek MultiView

Ovládací prvky rostou jako houby po dešti a dá se předpokládat, že v dalších verzích ASP.NET se setkáme s novými prvky. Proto si v posledním příkladu této kapitoly ukážeme nový prvek v ASP.NET 2.0 *Multiview*. *Multiview* je pouze zlomkem toho, co si z nových ovládacích prvků ukážeme. Kurz i učební text má stanoveny své proporce, a tak další nové prvky a jejich použití jsou ponechány zvídavějším studentům jako samostudium. Mezi ovládací prvky s více zobrazeními patří kromě *Multiview* i nový prvek *Wizard*.

Pojem k zapamatování: Ovládací prvek Multiview

MultiView je ovládací prvek umožňující deklarovat více zobrazení a přitom v dané situaci zobrazit pouze jedno. Nemá ale žádné výchozí rozhraní, vše musíme obstarat sami pomocí HTML prvků a dalších serverových ovládacích prvků.

Obr. 5.4 Práce s ovládacím prvkem MultiView v návrhovém režimu

Více pohledů umožníme pomocí prvku *MultiView* následujícím způsobem: nejprve si nadefinujme počet pohledů a nějaký výběrový prvek pro volbu jednoho z nich. K tomu se nejlépe hodí *DropDownList*. Poté nadefinujeme prvek *MultiView* a pro každé samostatné zobrazení jeden prvek `<asp:View>`. První krok máme tedy hotov a zdrojový kód vypadá takto:

Příklad Multiview – příprava formuláře

```
<form id="form1" runat="server">

<asp:DropDownList ID="ddl1" runat="server"
OnSelectedIndexChanged="DropDownList1_SelectedIndexChanged"
AutoPostBack="true">
     <asp:ListItem>Zvolený pohled</asp:ListItem>
     <asp:ListItem Value="0">Pohled 1</asp:ListItem>
     <asp:ListItem Value="1">Pohled 2</asp:ListItem>
     <asp:ListItem Value="2">Pohled 3</asp:ListItem>
     </asp:DropDownList>

<asp:Label ID="Label1" runat="server" Text="Zvol pohled menu">
</asp:Label>

<asp:MultiView ID="MultiView1" runat="server">
<asp:View ID="pohled1" runat="server">
</asp:View>
 <asp:View ID="pohled2" runat="server">
 </asp:View>
 <asp:View ID="pohled3" runat="server">
 </asp:View>
 </asp:MultiView>
</form>
```

Poté již definujeme jednotlivé pohledy dopsáním zdrojového kódu do každého samostatného pohledu, či vložením příslušných prvků v návrhovém režimu. Naši aplikaci tak založíme na volbě ze tří pohledů, přičemž první a druhý pohled bude zvolen zaškrtávacím polem *CheckBoxList*, jeden ve směru vodorovném a druhý ve směru svislém:

Příklad – Multiview – definice pohledu 1

```
<asp:View ID="pohled1" runat="server">
<asp:CheckBoxList ID="CheckBoxList1" runat="server"
RepeatDirection="Horizontal">
      <asp:ListItem>Let do vesmíru</asp:ListItem>
      <asp:ListItem>Parabolický let</asp:ListItem>
      <asp:ListItem>Přípravný kurz</asp:ListItem>
 </asp:CheckBoxList>
 </asp:View>
```

První a druhý pohled se tedy liší pouze atributem `RepeatDirection="Vertical|Horizontal"`

Příklad – Multiview – definice pohledu 2

```
<asp:View ID="pohled2" runat="server">
 <asp:CheckBoxList ID="CheckBoxList2" runat="server"
RepeatDirection="Vertical">
      <asp:ListItem>Let do vesmíru</asp:ListItem>
      <asp:ListItem>Parabolický let</asp:ListItem>
      <asp:ListItem>Přípravný kurz</asp:ListItem>
      </asp:CheckBoxList>
 </asp:View>
```

Třetí pohled trochu vylepšíme tím, že dodáme k jednotlivým volbám ještě ilustrativní obrázky:

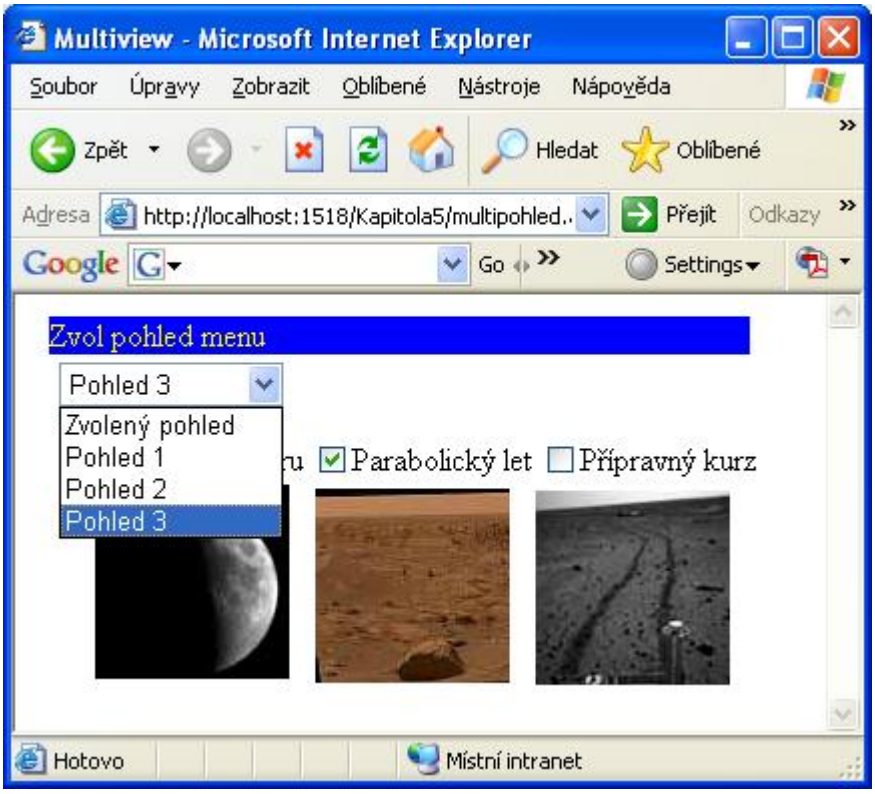
Příklad – Multiview – definice pohledu 3

```
<asp:View ID="pohled3" runat="server">
<asp:CheckBox ID="CheckBox1" runat="server" Text="Let do vesmíru" />
<asp:Image ID="Image1" runat="server" ImageUrl="~/images/01.jpg" />
<asp:CheckBox ID="CheckBox2" runat="server" Text="Parabolický let"/>
<asp:Image ID="Image2" runat="server" ImageUrl="~/images/02.jpg" />
<asp:Image ID="Image3" runat="server" ImageUrl="~/images/03.jpg" />
<asp:CheckBox ID="CheckBox3" runat="server" Text="Přípravný kurz" />
 </asp:View>
```

Pohledy máme vytvořeny, zbývá už jen obsloužit výběr zvoleného pohledu z *DropDownListu*. To provedeme tak, že ovládacímu prvku *MultiView* nastavíme aktivní pohled z vybrané hodnoty *DropDownListu*.

Příklad – Multiview – jednoduchý skript

```
<script runat="server">

    Sub DropDownList1_SelectedIndexChanged(ByVal sender As Object,
   ByVal e As System.EventArgs)
        MultiView1.ActiveViewIndex = ddl1.SelectedValue
    End Sub
</script>
```



Obr. 5.5 Výsledná aplikace s ovládacím prvkem MultiView

Zdrojový kód pro přepínání pohledů v některých případech ani nemusíme psát. *MultiView* totiž disponuje příkazy, které rozpozná v ovládacích prvcích tlačítek. Můžeme tedy použít libovolné tlačítko (*Button*, *LinkButton*, *ImageButton*) a přiřadit tlačítku pomocí příkazu *MultiView* jeho funkcionalitu. Příkazy jsou celkem 4 a vidíme je v následující tabulce.

Příkaz	Popis
PrevView	Přejde na předchozí pohled
NextView	Přejde na následující pohled
SwitchViewByID	Přejde na pohled s konkrétním ID získaným s ovládacího prvku tlačítka
SwitchViewByIndex	Přejde na pohled s konkrétním indexem získaným s ovládacího prvku tlačítka

Tab. 5.1 Názvy příkazu MultiView

Pokud tedy definujeme tlačítko *Předchozí* a *Další* s uvedeným *CommandName* z tabulky, získáme automaticky tlačítka přepínající jednotlivé pohledy *Multiview*. Definujeme tedy do pohledu č. 1 a 2 tlačítko *Další*:

```
<asp:Button ID="btnDalsi" runat="server" Text="Další"
CommandName="NextView"/>
```

Do pohledu číslo 2 a 3 definujeme tlačítko *Předchozí*:

```
<asp:Button ID="btnPredchozi" runat="server" Text="Předchozí"
CommandName="PrevView"/>
```

Zbývá ještě dopsat ovladač události *MultiView1_ActiveViewChanged* pro zpětnou vazbu na *DropDownList*. Pohybujeme-li se po pohledech tlačítky, *DropDownList* by se měl o tom dovědět:

```
Protected Sub MultiView1_ActiveViewChanged(obj As Object, e As EventArgs)
        ddl1.SelectedIndex = MultiView1.ActiveViewIndex
End Sub
```

Ke kapitole 5 je přiřazena demonstrační animace č. 4

Animace č. 4 obsahuje všechny příklady v této kapitole, tzn. obsáhlý formulář se serverovými ovládacími prvky, některé další ovládací prvky MultiView.

Shrnutí kapitoly

Serverové ovládací prvky mají tvar ***<asp:NázevPrvku runat="server" seznamAtributů/>***. Typy a hodnoty atributů jsou závislé na funkčnosti ovládacího prvku. Ke každému prvku je přiřazena množina vlastností, kterou zjistíme podle *IntelliSense*.

Novinkou v ASP.NET 2.0 je metoda Focus(). U ovládacích prvků, kde je vyžadován vstup z klávesnice můžeme nastavit *focus*, což je možnost zajišťující skutečnost, že po načtení formuláře do prohlížeče se začne pracovat jako první s tímto ovládacím prvkem.

Serverové ovládací prvky se neustále rozvíjejí. Kapitola detailně předvedla nový ovládací prvek v ASP.NET 2.0 MultiView.

MultiView je ovládací prvek umožňující deklarovat více zobrazení a přitom v dané situaci zobrazit pouze jedno. Nemá ale žádné výchozí rozhraní, vše musíme obstarat sami pomocí HTML prvků a dalších serverových ovládacích prvků.

Úkol k řešení 5.1 – Vytvoření formuláře s ovládacími prvky

Navrhněte formulář, ve kterém použijete širokou škálu serverových ovládacích prvků. Zajistěte vhodnou obsluhu výsledků z tohoto formuláře.

Úkol k řešení 5.2 – Metoda Focus()

Zajistěte ve vašem formuláři, aby měl uživatel nastaven kurzor v textboxu, který je vhodné vyplnit nejdříve.

Úkol k řešení 5.3 – Tlačítka v MultiView

Řešený příklad demonstrovaný v animaci doplňte o tlačítka, které zajišťují přepínání mezi pohledy vpřed a vzad.

Kontrolní otázka 5.1

Vyjmenujte standardní serverové ovládací prvky a některé jejich klíčové členy.

Kontrolní otázka 5.2

Jaké serverové ovládací prvky máme k dispozici pro seznamy?

Kontrolní otázka 5.3

K čemu slouží metoda Focus()?

Kontrolní otázka 5.4

Charakterizujte ovládací prvek MultiView.

Kontrolní otázka 5.5

Jakým způsobem definujeme barvy u serverových ovládacích prvků?

6. VALIDAČNÍ OVLÁDACÍ PRVKY

Čas ke studiu: 3 hodiny

Cíl Po prostudování této kapitoly budete umět

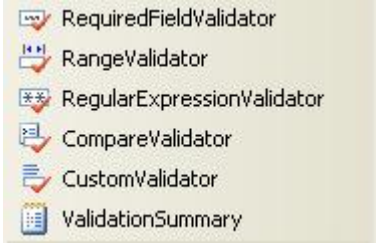
- popsat funkcionalitu validačních ovládacích prvků
- zajistit kontrolu vstupních údajů od uživatele
- validovat formuláře všemi nabízenými ovládacími prvky pro validaci
- vytvářet regulární výrazy pro kontrolu složitějších vstupních údajů
- naimplementovat vlastní validační logiku
- vhodnou formou informovat uživatele o chybách vstupních údajů

Výklad

Tato kapitola je věnována ověřovacím ovládacím prvkům. Web aplikace jsou ve velké míře navrženy tak, aby spolupracovaly s uživatelem. Minulé kapitoly nám představily mnoho ovládacích prvků, pomocí nichž uživatel odesílá své údaje na server. Bez těchto mechanismů by nemohly naplnit svou podstatu internetové obchody, diskusní fóra a mnoho dalších typů aplikací. Jak ale donutit uživatele, aby zadal správné údaje? Samozřejmě je v jeho vlastním zájmu, aby údaje ve své objednávce byly pravdivé a správci těchto aplikací mají i jiné možnosti pro ověření identity (např. telefon). Kromě úmyslných chyb a nepravdivých údajů však musíme vzít v potaz i fakt, že člověk není neomylný a některé údaje může opomenout nebo udělat tu a tam překlep. Tyto chyby by mohly mít za následek spousty problémů jako nesprávně vyplněné tabulky uživatelů, časová náročnost při dohledávání chyb apod. Proto musí existovat mechanismus, který tyto chyby odhalí již na samotném počátku celé akce.

6.1 Ovládací prvky pro ověřování obecně

Ten, kdo již dříve vyvíjel aplikace, kde musel implementovat ověření, zda-li uživatel zadal správná data potvrdí, že se nejednalo o jednoduchou práci. Především zdlouhavá časová náročnost a odlišný přístup ke kontrole různých typů vstupních informací způsoboval časté problémy s aplikacemi. Druhým problémem je odlišný přístup na straně klienta a na straně serveru. Na straně klienta se typicky pro kontrolu vstupních údajů používal *javascript*. Dobře napsaný skript tak odhalí spoustu chyb a překlepů ještě před odesláním na server a ušetří se tak serverové zdroje. Jenže platnost dat je nutné ověřit také na serveru. Proto byly v ASP.NET vyvinuty validační ovládací prvky, které je možné svázat s ovládacími prvky, které vyžadují nějaký vstup od uživatele a tyto validátory automaticky ověřují platnost na obou stranách.



Obr. 6.1 Validační ovládací prvky v ASP.NET

Validačních ovládacích prvků je šest, jejich ikonky z toolboxu vidíme na předchozím obrázku. Tyto validátory pamatují na většinu obvyklých situací, jak kontrolovat data, a jsou ve své podstatě předepsanou sadou kontrol, které bychom psali zdlouhavým kódem. Pokud tato předepsaná sada validátorů nevyhovuje naší situaci pro kontrolu dat, máme možnost definovat vlastní konstrukci validátoru. Nejprve ale popišme princip jednotlivých validačních prvků:

Validační prvek	Charakteristika
<asp:RequiredFieldValidator>	Zkontroluje, zda-li ovládací prvek, do kterého se vyplňují data, není prázdný.
<asp:CompareValidator>	Zkontroluje porovnáním, zda-li požadovaná hodnota vyhovuje předepsané hodnotě.
<asp:RangeValidator>	Zkontroluje, zda-li vstupní hodnota vyhovuje určitému povolenému rozsahu hodnot.
<asp:RegularExpressionValidator>	Zkontroluje, zda-li vstupní hodnota vyhovuje předepsanému regulárnímu výrazu.
<asp:CustomValidator>	Zkontroluje, zda-li vstupní data odpovídají předepsané vlastní logice, kterou implementoval vývojář aplikace.
<asp:ValidationSummary>	Zobrazí souhrné chybové hlášení ze všech přítomných validátorů.

Tab. 6.1 Validační ovládací prvky

Pojem k zapamatování: Upozornění na práci s validátory

Ověřujeme-li data od uživatele, mějme na paměti, že pokud kontrolujeme data jiným validátorem než je *RequiredFieldValidator* a uživatel ponechá pole prázdné, validace skončí úspěšně.

Z výše uvedeného faktu plyne skutečnost, že budeme muset většinou kombinovat pro kontrolu vstupních dat *RequiredFieldValidátor* s dalším validátorem.

Používáme-li pro odesílání formuláře na server tlačítko, měli bychom znát jeho vlastnost *CausesValidation*, která je typu *Boolean*. Pokud je její hodnota nastavena na *false,* ASP.NET validační prvky ignoruje a stránka je odeslána zpět na server, kde pokračuje dále obsluha událostí. Proto je výchozí hodnotou *true* a ASP.NET automaticky ověřuje stránku ihned po odeslání na server.

Pojem k zapamatování: Ne všechny ovládací prvky lze ověřovat

Nejčastěji se ověřují vstupní data z textboxů, ale i další prvky můžeme validovat. Patří mezi ně ListBox, DropDownList a RadioButtonList, u kterých ověřujeme platnost vybrané položky. V seznamu možných prvků pro validaci jsou i HTML ovládací prvky InputText, TextArea a Select.

Nemáme však žádné mechanismy pro ověřování zaškrtnutí položky v RadioButtonu či CheckBoxu.

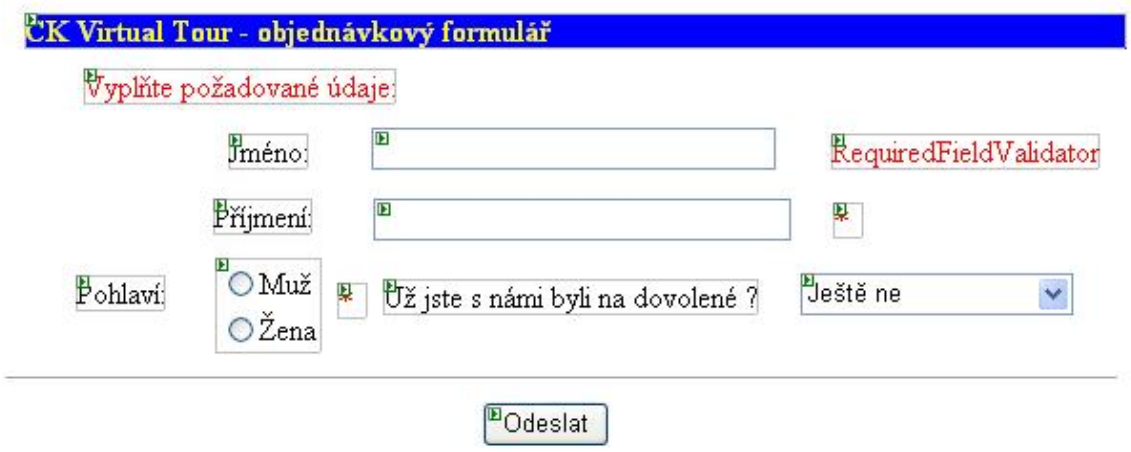
Vlastnosti konkrétních validátorů jsou dané jejich funkcionalitou. Následující vlastnosti mají ale společné, neboť jsou zděděny z třídy *BaseValidator*. Třída *BaseValidator* je rovněž součástí jmenného prostoru *System.Web.UI.WebControls*.

Vlastnost	Charakteristika
ControlToValidate	Definuje ovládací prvek, se kterým je validátor svázán.
Display	Určuje, zdali se chybová zpráva zobrazuje staticky či dynamicky.
EnableClientScript	Určuje vlastnost typu Boolean, zda-li se bude ověřovat platnost u klienta.
Enabled	Zapnutí nebo vypnutí validátoru. Nepřístupný prvek dočasně nic neověřuje.
ErrorMessage	Chybová zpráva validátoru.
Text	Chybový text ve validačním ovládacím prvku.
IsValid	Určuje, zda-li je ovládací prvek svázaný s validátorem platný.
SetFocusOnError	Pokud prvek neprojde validací, je na něm nastaven focus.
ValidationGroup	Seskupuje více validátorů do skupiny.
Validate()	Metoda provedení validace a nastavení vlastnosti IsValid.

Tab. 6.2 Vlastnosti validačních ovládacích prvků

6.2 Příklady validačních ovládacích prvků

Všechny výše uvedené validační prvky si procvičíme na konkrétních příkladech. Prvním a nejčastěji používaným validátorem je *RequiredFieldValidator*. Vraťme se k naší virtuální cestovní kanceláři. Vytvořme zkrácenou verzi formuláře, kde budeme vyžadovat po uživateli, aby zadal některé údaje. Formulář máme zobrazen na následujícím obrázku.



Obr. 6.2 Vstupní formulář s *ReguiredFieldValidátor*y

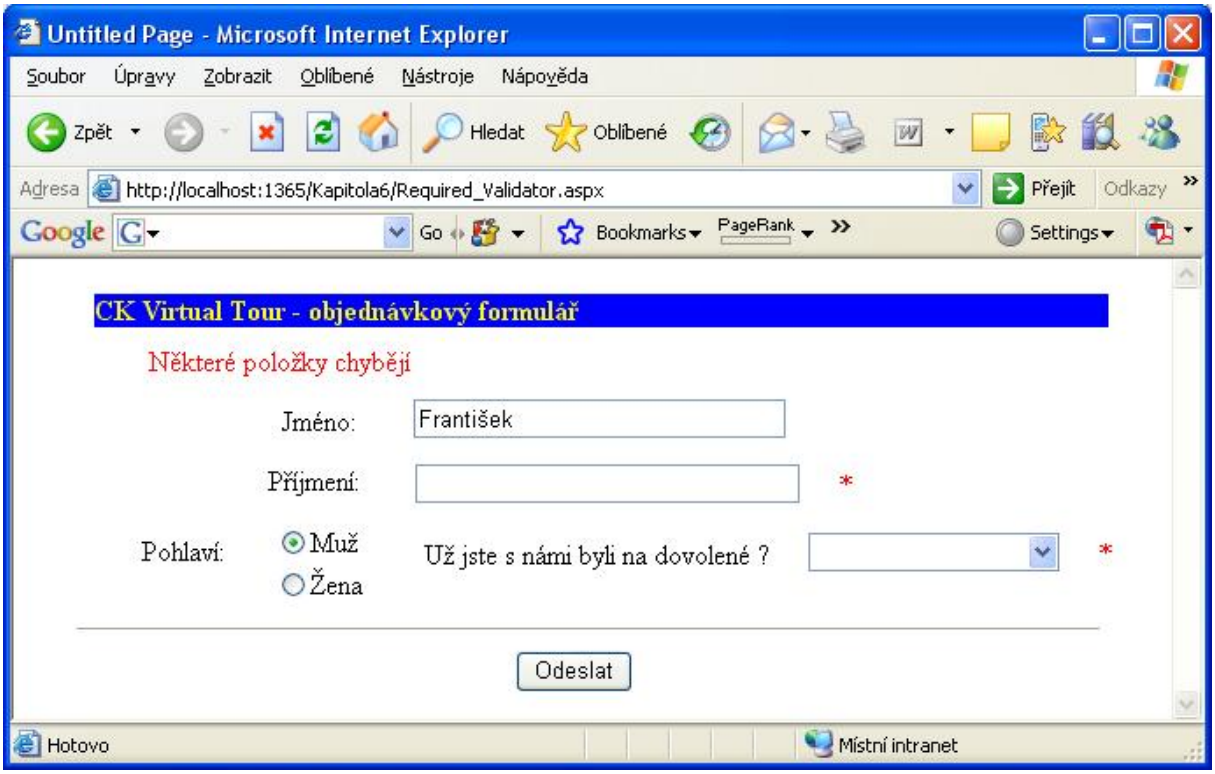
Navržený formulář bude nutný vyplnit celý, tzn. kromě jména a příjmení požadujeme vyplnit i položku pohlaví a výběr z možností odpovědi na otázku. Na obrázku vidíme formulář v návrhovém zobrazení. Validátor je zobrazen červeně a pokud vyplníme v jeho vlastnostech atribut chybového hlášení (*Error Message*), zobrazí se tímto hlášením.

(Expressions)	
(ID)	**RequiredFieldValidator1**
AccessKey	
BackColor	
BorderColor	
BorderStyle	NotSet
BorderWidth	
ControlToValidate	
CssClass	
Display	Static
EnableClientScript	True
Enabled	True
EnableTheming	True
EnableViewState	True
ErrorMessage	**RequiredFieldValidator**
Font	
ForeColor	Red
Height	
InitialValue	
SetFocusOnError	False
SkinID	
TabIndex	0
Text	

Obr. 6.3 Okno *properties RequiredFieldValidátoru* umožňuje pohodlné nastavení vlastností

Pojem k zapamatování: ErrorMessage RequiredFieldValidátoru

ErrorMessage je vlastnost validátoru, která je zobrazena v případě chyby. Pokud tedy vstupní hodnota vázaného ovládacího prvku je prázdná, zobrazí se chybové hlášení. U *RequiredFieldValidátoru* se nejčastěji v praxi vypisovaná chyba oznamuje hvězdičkou.



Obr. 6.4 Nevyplníme-li povinné položky, objeví se chybové hlášení, v tomto případě známá hvězdička

Formulář tedy obsahuje dva *Textboxy*, *DropDownList* a *RadioButtonList*. Všechny tyto ovládací prvky jsou svázány se svými *RequiredFieldValidátory* atributem *ControlToValidate*. Formulář je odeslán tlačítkem, které obsluhuje výslednou validaci na serveru.

Příklad – Formulář s RequiredFieldValidátorem

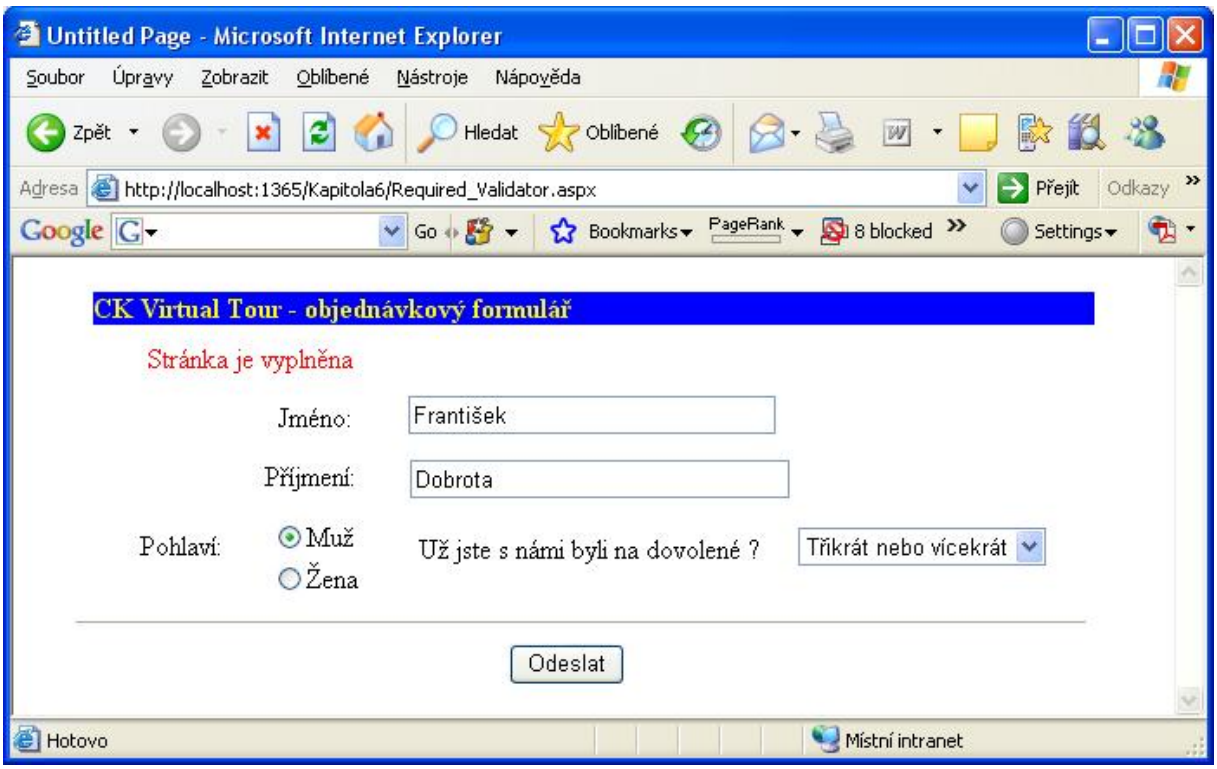
```
<form id="form1" runat="server">
    <asp:TextBox ID="tbJmeno" runat="server" </asp:TextBox>
    <asp:TextBox ID="tbPrijmeni" runat="server" </asp:TextBox>
    <asp:RadioButtonList ID="rblVek" runat="server">
            <asp:ListItem>Muž</asp:ListItem>
            <asp:ListItem>Žena</asp:ListItem>
    </asp:RadioButtonList>
   <asp:DropDownList ID="ddlOtazka" runat="server">
            <asp:ListItem></asp:ListItem>
            <asp:ListItem>Ještě ne</asp:ListItem>
            <asp:ListItem>jednou nebo dvakrát</asp:ListItem>
            <asp:ListItem>Třikrát nebo vícekrát</asp:ListItem>
   </asp:DropDownList>
   <asp:Label ID="Label1" runat="server" Text="Už jste s námi byli
      na dovolené ?"></asp:Label>
  <asp:Label ID="lblPrijm" runat="server" Text="Příjmení:">
    </asp:Label>
   <asp:Label ID="Label3" runat="server" Text="Pohlaví:">
      </asp:Label>
   <asp:Label ID="lbJmeno" runat="server" Text="Jméno:">
      </asp:Label>
   <asp:Label ID="Label2" runat="server" Text="CK Virtual Tour -
      objednávkový formulář"></asp:Label>
   <asp:Button ID="btnOdeslat" runat="server" Text="Odeslat"
      OnClick="validuj"/>
   <hr/>
   <asp:Label ID="lblInfo" runat="server">Vyplňte požadované
      údaje:</asp:Label>
   <asp:RequiredFieldValidator ID="RFV1" runat="server"
      ErrorMessage="*" ControlToValidate ="tbJmeno">
      </asp:RequiredFieldValidator>
   <asp:RequiredFieldValidator ID="RFV2" runat="server"
      ErrorMessage="*" ControlToValidate="tbPrijmeni">
      </asp:RequiredFieldValidator>
   <asp:RequiredFieldValidator ID="RFV3" runat="server"
      ErrorMessage="*" ControlToValidate="rblVek">
   </asp:RequiredFieldValidator>
  <asp:RequiredFieldValidator ID="RFV4" runat="server"
      ErrorMessage="*" ControlToValidate="ddlOtazka">
      </asp:RequiredFieldValidator>
   </form>
```

Pokud jsou všechny ovládací prvky vyplněny nějakou hodnotou (byť nemusí dávat vůbec žádný smysl – od toho máme jiné validátory), je formulář platným.

Pojem k zapamatování: Vlastnost IsValid objektu Page

Pokud jsou všechny ovládací prvky platné, je hodnota vlastnosti *Page.isValid* rovna *True*. V jediném okamžiku se tak dovíme, zda-li jsou všechny ovládací prvky v pořádku.

Vraťme se k naší aplikaci. Na předcházejícím obrázku jsme viděli chybové hlášení ve tvaru červené hvězdičky u nevyplněných ovládacích prvků. Nyní jsou ovládací prvky vyplněny všechny, stránka je tedy validní. To zajišťuje obsluha následujícího skriptu.



Obr. 6.5 Správně vyplněný formulář

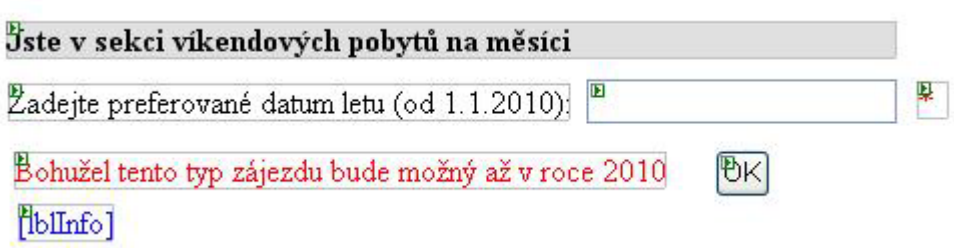
Příklad – Skript stránky s validátorem

```
<script runat="server">
    Sub Validuj(ByVal sender As Object, ByVal e As EventArgs)
        If (Page.IsValid) Then
            lblInfo.Text = "Stránka je vyplněna"
        Else
            lblinfo.Text = "Některé položky chybějí"
        End If
    End Sub

</script>
```

Pokud máme všechny ovládací prvky vyplněny, neznamená to, že máme vyhráno. Ovládací prvek *ReguiredFieldValidátor* pouze zkontroloval, zda-li jsou všechny povinné prvky vyplněny. O vyplněnou hodnotu se musíme postarat jinými validátory, respektive musíme definovat podmínky pro vstupní hodnoty ovládacích prvků, které jsme schopni akceptovat. Nemůžeme uživatele donutit říci pravdu, můžeme však obdržet hodnoty, které jsou pro nás přípustné (těžko bychom akceptovali rok narození, který ještě nenastal, popřípadě 150 let starý).

Postupně se tedy dostáváme k dalším validačním prvkům. Druhým validátorem, který si názorně předvedeme, bude *CompareValidátor*. Tento validátor využijeme, požadujeme-li od uživatele nějakou hodnotu, kterou můžeme porovnat s konstantní hodnotou nebo hodnotou jiného ovládacího prvku.

V našem příkladu budeme porovnávat hodnotu ze vstupu, která je typu datum. Na obrázku 6.6 vidíme připravený formulář s *CompareValidátorem*. Samozřejmě prázdná hodnota by byla vyhodnocena jako správná, proto použijeme na kontrolu textboxu také *RequiredFieldValidátor*.



Obr. 6.6 Formulář s CompareValidátorem pro porovnání zadaného data z textboxu

Pojem k zapamatování: Vlastnosti CompareValidátoru

Vlastnost *Type* určuje datový typ, který budeme porovnávat. Dalšími vlastnostmi jsou *ValueToCompare* pro porovnání s konstantní hodnotou nebo *ControlToCompare* pro porovnání s hodnotou jiného ovládacího prvku. Máme možnost použít pouze jednu z těchto možností.

Příklad – Formulář s CompareValidátorem

```
<form id="form1" runat="server">
 <asp:TextBox ID="tbDatum" runat="server" ></asp:TextBox>
 <asp:Label ID="Label1" runat="server" Text="Zadejte preferované
     datum letu (od 1.1.2010):"></asp:Label>
 <asp:Label ID="Label2" runat="server" Text="Jste v sekci
     víkendových pobytů na měsíci"></asp:Label>
 <asp:Button ID="Bt" runat="server" Text="OK" OnClick= "Validuj" />

 <asp:CompareValidator ID="CmpV" runat="server"
   ControlToValidate="tbDatum" Type="Date" ValueToCompare="1/1/2010"
   Operator="GreaterThan" ErrorMessage="Bohužel tento typ zájezdu
   bude možný až v roce 2010"></asp:CompareValidator>
 <asp:Label ID="lblInfo" runat="server"></asp:Label>
 <asp:RequiredFieldValidator ID="RFV1" runat= "server"
   ControlToValidate="tbDatum" ErrorMessage="*">
   </asp:RequiredFieldValidator>
</form>
```

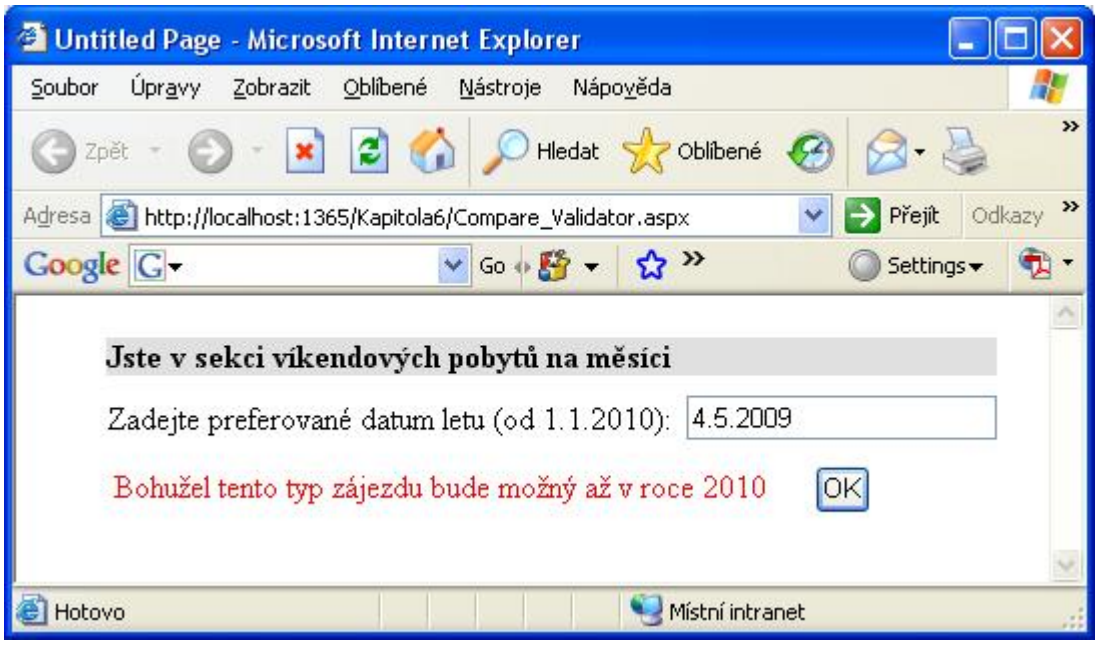
Pojem k zapamatování: Nastavení datového typu u validátoru

S vlastností *Type* se setkáme také u RangeValidátoru. Datový typ, který budeme porovnávat, může nabývat hodnot Integer, Double, String, Currency a Date.

Příklad – Skript příkladu s CompareValidátorem

```
<script runat="server">
    Sub Validuj(ByVal sender As Object, ByVal e As EventArgs)
        If (Page.IsValid) Then
            lblInfo.Text = "Váš požadavek byl přijat"
        End If
    End Sub
</script>
```

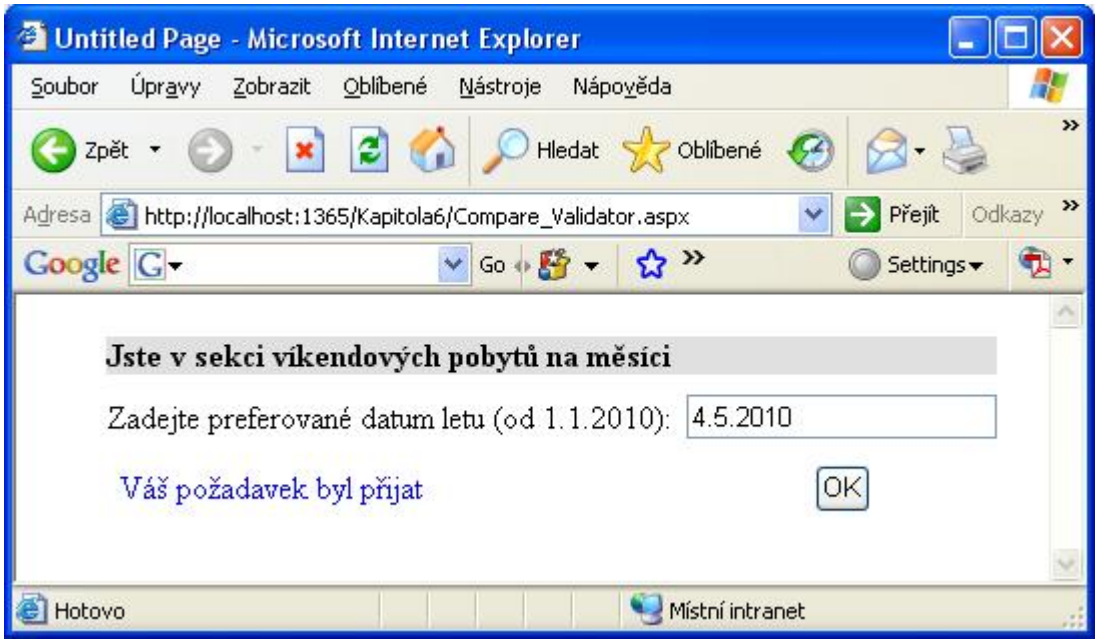
Pokud tedy zadáme vstupní hodnotu, která s porovnanou hodnotou vyhodnotí výraz záporně, validátor ohlásí chybu:



Obr. 6.7 Výsledkem porovnání hodnot je *False*

Pojem k zapamatování: Operátor porovnání

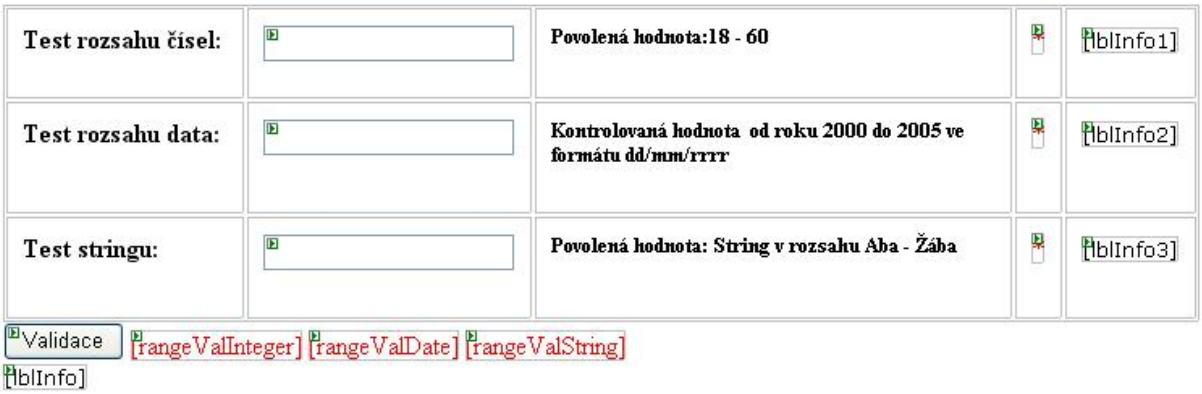
Vlastnost *Operator* určuje, jakým způsobem se hodnoty porovnají. Máme na výběr tyto možnosti: *Equal*, *NotEqual*, *GreaterThan*, *GreaterThanEqual*, *LessThan* a *LessThanEqual*. Kromě těchto hodnot známých relačních operátorů máme ještě možnost zvolit hodnotu *DataTypeCheck*, která zkontroluje, zda-li hodnota odpovídá datovému typu ve vlastnosti *Type*.



Obr. 6.8 Správně vyplněný formulář po kontrole CompareValidátoru

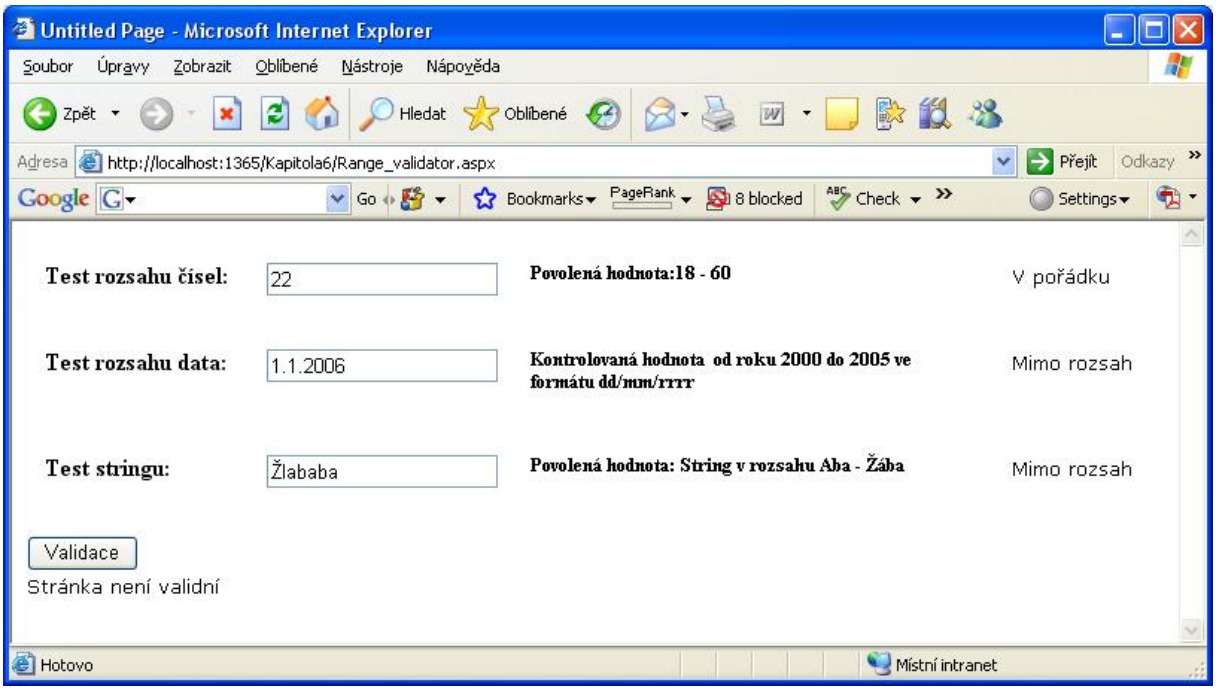
Podobným validátorem jako *CompareValidátor* je *RangeValidátor*, ten pracuje se stejnými datovými typy, jen místo relačních operátorů definuje rozsahy platných hodnot pro vstupní data.

Třetí příklad nebude souviset s naší virtuální cestovní kanceláří, ukážeme si test hodnot různých datových typů. Připravme si formulář, kde otestujeme rozsahy čísla, datumu a řetězce.



Obr. 6.9 Formulář pro validaci dat *RangeValidátorem*

Povolené hodnoty oznámíme uživateli v popisku. Po stisku validačního tlačítka *RangeValidátory* zkontrolují hodnoty a oznámí případnou chybu.



Obr. 6.10 Příklad nevyhovujících vstupních údajů

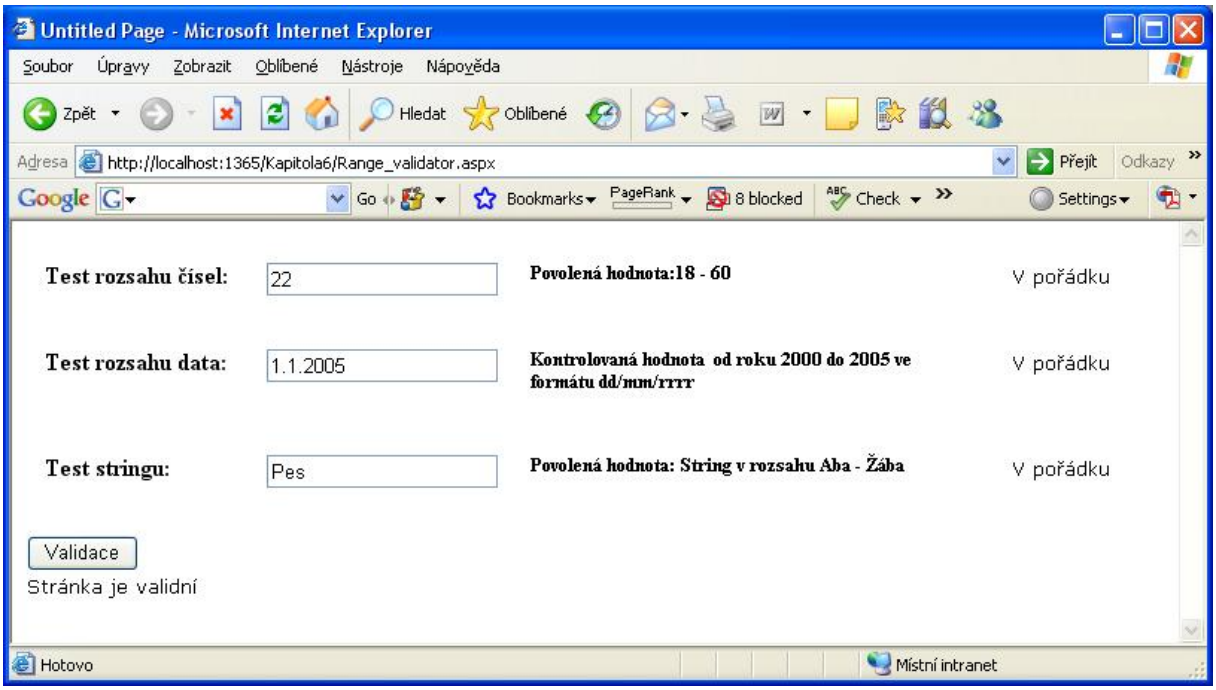
Pojem k zapamatování: Definice rozsahu platnosti u RangeValidátoru

Rozsah platnosti definujeme vlastnostmi *MinimumValue* a *MaximumValue*. Tyto hodnoty pak nadefinujeme tak, aby odpovídaly zvolenému datovému typu *Type*.

V uvedeném skriptu si všimněte, jakým způsobem definujeme minimální a maximální povolenou hodnotu pro jednotlivé datové typy.

Příklad – Formulář s RangeValidátory

```
<form id="Form1" runat="server">
  Test rozsahu čísel:
        <asp:TextBox id="tbCislo" runat="server"/>
        <h5> Povolená hodnota:18 - 60</h5>
      <asp:RequiredFieldValidator ID="RequiredFieldValidator1"
          runat="server" ControlToValidate="tbCislo"
          ErrorMessage="*"></asp:RequiredFieldValidator>
        <asp:Label id="lblInfo1" runat="server" />
   Test rozsahu data:
         <h5><asp:TextBox id="tbDatum" runat="server"/></h5>
         <h5>Kontrolovaná hodnota  od roku 2000 do 2005 ve
    formátu dd/mm/rrrr</h5>
        <asp:RequiredFieldValidator ID="RFV2" runat="server"
             ControlToValidate="tbDatum"
             ErrorMessage="*"></asp:RequiredFieldValidator>
             <asp:Label id="lblInfo2" runat="server" />
  Test stringu:
        <h5><asp:TextBox id="tbString" runat="server"/></h5>
        <h5>Povolená hodnota: String v rozsahu Aba - Žába</h5>
         <asp:RequiredFieldValidator ID="RFV3" runat="server"
             ControlToValidate="tbString"
              ErrorMessage="*"></asp:RequiredFieldValidator>
        <asp:Label id="lblInfo3" runat="server" />
        <asp:Button Text="Validace" ID="Bt1" onclick="Button1_Click"
         runat="server" />

     <asp:RangeValidator
        id="rangeValInteger"
        Type="Integer"
        ControlToValidate="tbCislo"
        MaximumValue="60"
        MinimumValue="18"
        runat="server" EnableClientScript="False"/>

     <asp:RangeValidator
        id="rangeValDate"
        Type="Date"
        ControlToValidate="tbDatum"
        MaximumValue="31/12/2005"
        MinimumValue="1/1/2000"
        runat="server" EnableClientScript="False"/>

        <asp:RangeValidator
        id="rangeValString"
        Type="String"
        ControlToValidate="tbString"
        MaximumValue="Žába"
        MinimumValue="Aba"
        runat="server" EnableClientScript="False"/>
     <br/>

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

    </form>
```



Obr. 6.11 Všechny vstupní hodnoty odpovídají požadavkům

Příklad – Obslužný skript pro příklad s RangeValidátory

```
<script runat="server">
    Sub Button1_Click(ByVal sender As Object, ByVal e As EventArgs)
        rangeValInteger.Validate()
        If (rangeValInteger.IsValid) And (tbCislo.Text <> "") Then
            lblInfo1.Text = "V pořádku"
        Else
            lblInfo1.Text = "Mimo rozsah"
        End If

        rangeValDate.Validate()
        If (rangeValDate.IsValid) And (tbDatum.Text <> "") Then
            lblInfo2.Text = "V pořádku"
        Else
            lblInfo2.Text = "Mimo rozsah"
        End If

        rangeValString.Validate()
        If (rangeValString.IsValid) And (tbString.Text <> "") Then
            lblInfo3.Text = "V pořádku"
        Else
            lblInfo3.Text = "Mimo rozsah"
        End If

        If (Page.IsValid) Then
            lblInfo.Text = "Stránka je validní"
        Else
            lblInfo.Text = "Stránka není validní"
        End If
    End Sub

</script>
```

6.3 Regulární výrazy a jejich použití ve validaci

Regulární výraz je předpis pro textové řetězce. Pomocí něj jsou specifikovány podmínky, kterým řetězce musí vyhovovat. Umožňují vytvářet pravidla pro komplikované tvary řetězců. Jsou jediným nástrojem pro možnost kontroly nad určitou posloupností znaků, které například charakterizují tvar e-mailové adresy, URL adresy, různých telefonních čísel a dalších řetězců, pro které platí určitá pravidla. Regulární výrazy jsou tvořeny metaznaky a kvantifikátory. Metaznaky určují různé zápisy libovolných znaků, které se mohou v řetězci vyskytovat.

Metaznaky	Charakteristika
.	Tečka reprezentuje jakýkoliv znak kromě \n.
[abc]	Vyhovuje jakýkoliv jediný znak, který je uveden v množině.
[^abc]	Vyhovuje jakýkoliv znak, který není uveden v množině.
[1-9a-zA-Z]	Vyhovující jsou znaky z uvedeného rozsahu.
\w	Vyhovuje jakýkoliv slovní znak, tj. písmena, číslice a podtržítko.
\W	Vyhovuje jakýkoliv jiný znak než písmeno, číslice a podtržítko.
\s	Vyhovuje jakýkoliv prázdný znak (mezera, tabulátor, nový řádek atd.).
\S	Vyhovuje jakýkoliv neprázdný znak.
\d	Vyhovuje jakýkoliv znak, který je číslicí.
\D	Vyhovuje libovolný znak, který není číslicí.

Tab. 6.3 Metaznaky regulárních výrazů

Pomocí metaznaků definujeme přípustné posloupnosti znaků. Aby byl nástroj regulárních výrazů ještě mocnější, jsou zavedeny tzv. kvantifikátory, které určují počet výskytů jednotlivých skupin metaznaků. Z vlastní zkušenosti víte, že například doména za poslední tečkou může mít pouze dva až čtyři znaky (*cz*, *com*, *info apod.*). Tuto podmínku vyjádříme prostřednictvím kvantifikátoru.

Kvantifikátor	Charakteristika
*	Vyhovuje jakýkoliv počet výskytů (i nulový).
+	Vyhovuje jeden nebo více výskytů.
?	Vyhovuje žádný nebo jeden výskyt.
{N}	Vyhovuje N výskytů.
{N,}	Vyhovuje N nebo více výskytů.
{N,M}	Vyhovuje N až M výskytů.

Tab. 6.4 Kvantifikátory regulárních výrazů

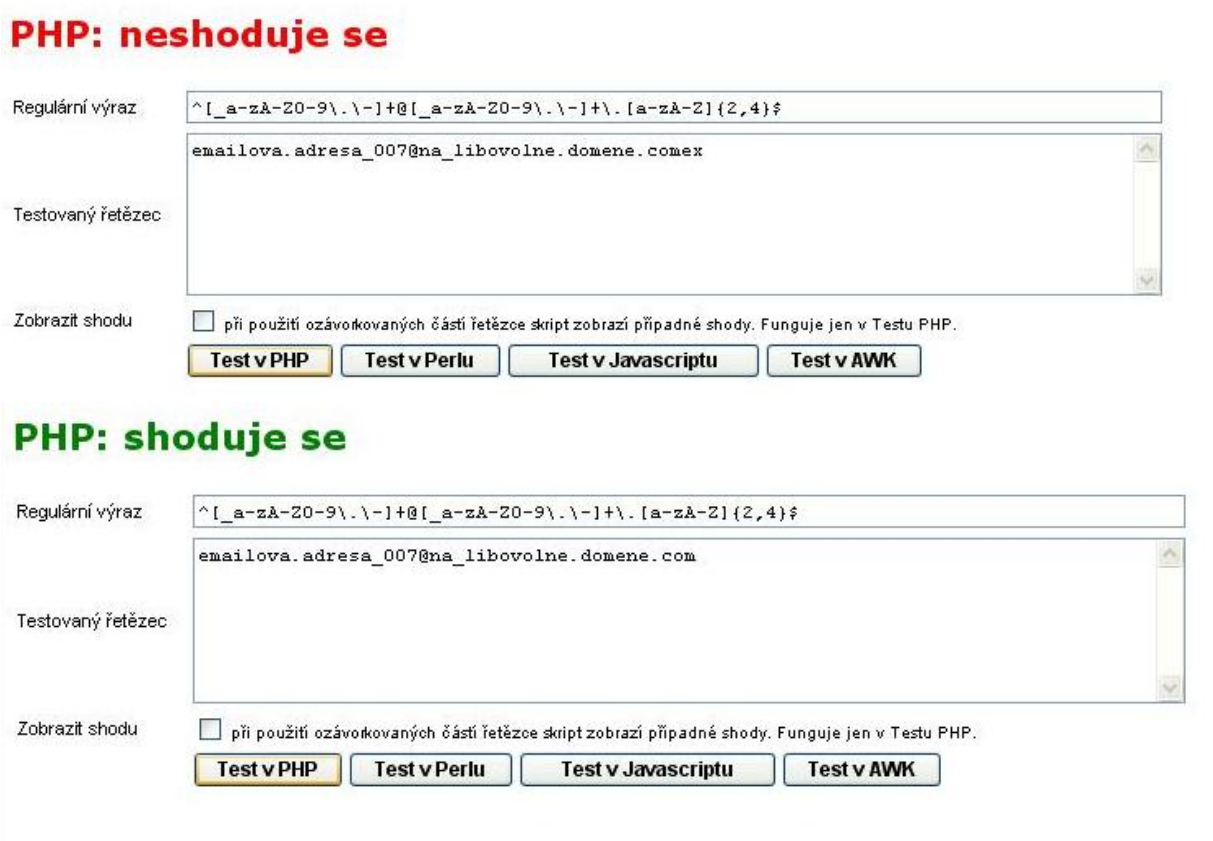
Mohli bychom nyní celou kapitolu věnovat různým regulárním výrazům a na množství příkladu si je procvičovat. To ale není náplní tohoto kurzu a proto bych vás rád odkázal na užitečné stránky:

http://www.regexp.cz, kde si můžete celou řadu regulárních výrazů procvičit.

Korespondenční úkol – Časté tvary regulárních výrazů

Procvičte si některé regulární výrazy jako např. na tvar e-mailové adresy, zadávání hesla složeného se znaků a číslic, PSČ, telefonní číslo apod.

Na uvedené stránce můžete testovat výrazy v různých skriptovacích jazycích. Můžete vyzkoušet, zda-li vstupní řetězce odpovídají příslušným regulárním výrazům. Na následujícím obrázku vidíme test e-mailové adresy.

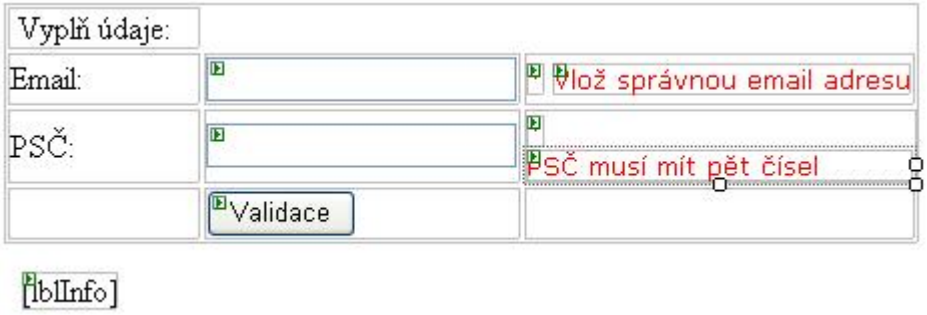


Obr. 6.12 Testy regulárních výrazů

Pojem k zapamatování: Použití regulárního výrazu ve validátoru

Robustní nástroj regulárních výrazů využívá *RegularExpressionValidátor*. Přesto, že se jedná o mocný validátor, jeho použití je jednoduché. Do vlastnosti *ValidationExpression* dosadíme vytvořený regulární výraz.

Vraťme se ale k Visual Studiu a k našim procvičovaným ovládacím prvkům. Využijme nově získané vědomosti a definujme formulář, který využije *RegularExpresionValidátoru* ke kontrole e-mailové adresy a poštovního směrovacího čísla.



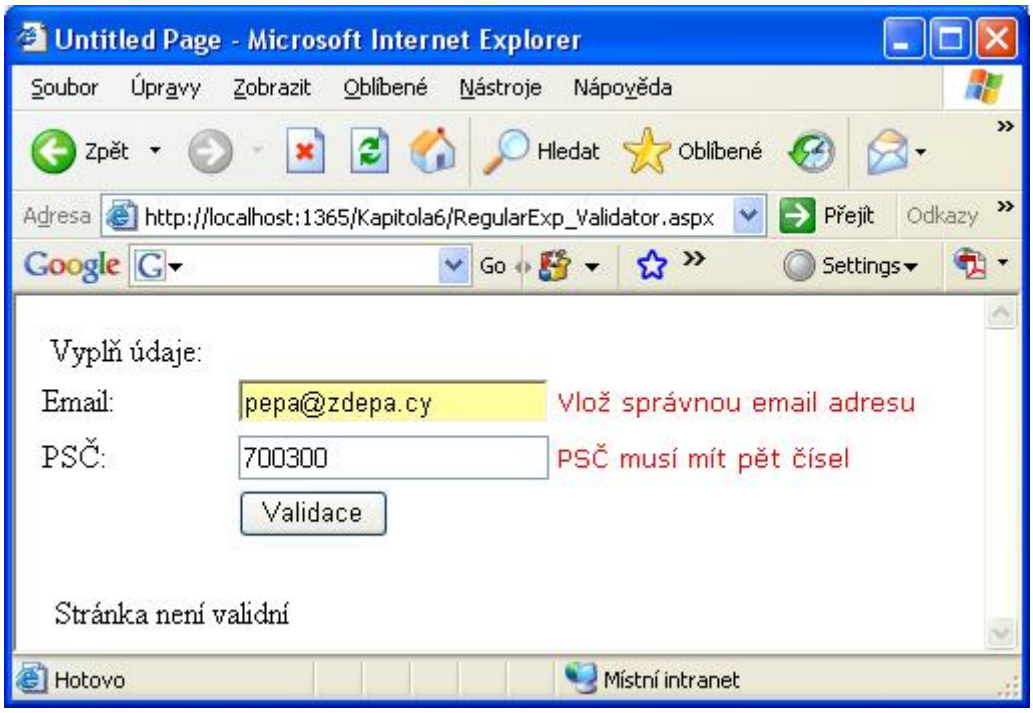
Obr. 6.13 Navržený formulář pro validaci údajů regulárními výrazy.

Nyní uveďme celý kód formuláře aplikace s důrazem na validátory regulárních výrazů. Regulární výraz pro kontrolu e-mailu může mít mnoho podob a každá aplikace může vyžadovat svůj konkrétní tvar. S poštovním směrovacím číslem to již tak neobvyklé není. Podle našich pravidel musí PSČ obsahovat pouze číslice a jeho délka je striktně pět.

Příklad – Formulář aplikace s validátory regulárních výrazů

```
<form id="Form1" runat="server">

    <table>
    <tr valign="top">
      <td style="width: 94px">
           Vyplň údaje:</td>
    </tr>

    <tr>
      <td style="width: 94px">
       Email:
      </td>
      <td>
        <ASP:TextBox id="TextBox1" runat="server" />
      </td>
      <td>
        <asp:RequiredFieldValidator id="RFV1" runat="server"
            ControlToValidate="TextBox1"
            Display="Dynamic"
            Font-Names="Verdana" Font-Size="10pt"
            EnableClientScript="False"
            >
            *
        </asp:RequiredFieldValidator>

        <asp:RegularExpressionValidator id="RExpV1" runat="server"
            ControlToValidate="TextBox1"
            ValidationExpression="^[\w-]+@[\w-
            ]+\.(cz|com|net|org|edu|mil)$"
            Display="Static"
            Font-Names="verdana" Font-Size="10pt"
            EnableClientScript="False">
            Vlož správnou email adresu
        </asp:RegularExpressionValidator>
      </td>
    </tr>
    <tr>
      <td style="width: 94px">
       PSČ:
      </td>
      <td>
        <ASP:TextBox id="TextBox3" runat="server" />
      </td>
      <td>
        <asp:RequiredFieldValidator id="RFV3" runat="server"
            ControlToValidate="TextBox3"
            Display="Dynamic"
            Font-Names="Verdana" Font-Size="10pt"
            EnableClientScript="False">
              *
        </asp:RequiredFieldValidator>

        <asp:RegularExpressionValidator id="RExpV3"
            ControlToValidate="TextBox3"
            ValidationExpression="^\d{5}$"
            Display="Static"
            Width="100%"
            Font-Names="verdana" Font-Size="10pt"
```

```
                runat="server" EnableClientScript="False">
                PSČ musí mít pět čísel
            </asp:RegularExpressionValidator>
        </td>
    </tr>
    <tr>
        <td style="width: 94px"></td>
        <td>
            <ASP:Button ID="Button1" text="Validace"
             OnClick="zkontroluj" runat="server" />
        </td>
        <td></td>
    </tr>
    </table>
        <asp:Label ID="lblInfo" runat="server" ></asp:Label>

</form>
```

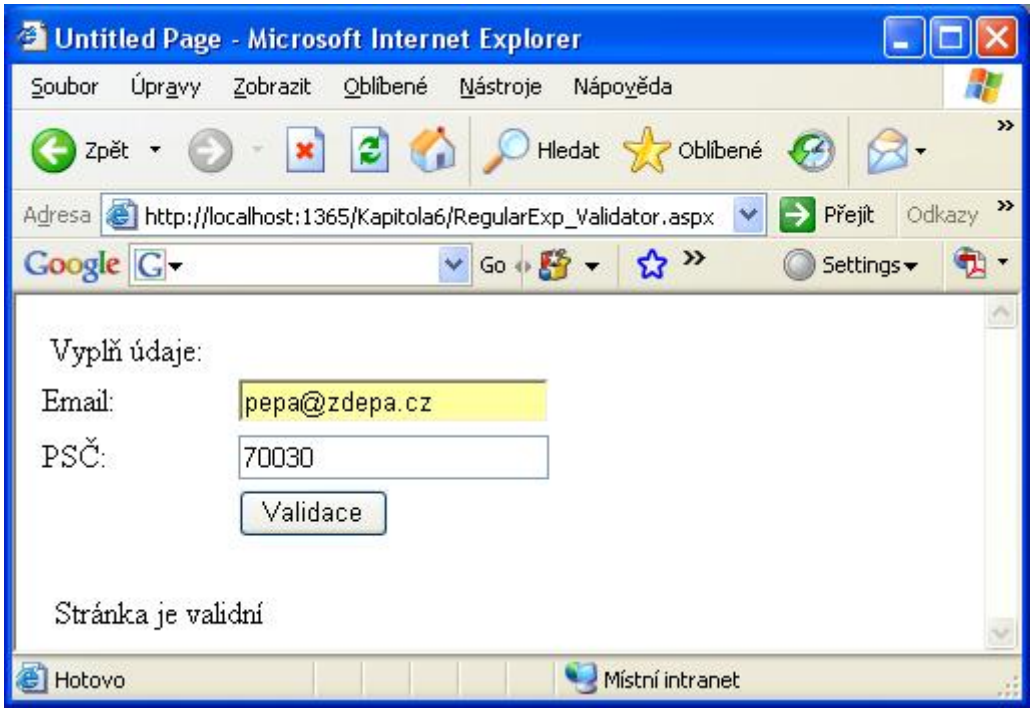


Obr. 6.14 Špatně vyplněný formulář

Na obrázku 6.14 vidíme údaje, které nejsou platné pro regulární výrazy. PSČ obsahuje jednu číslici navíc a doména *cy* není v našem regulárním výrazu povolena. Opravme údaje a odešleme formulář znovu, tentokrát je vše v pořádku (viz. obr. 6.15). Pro úplnost ještě uveďme skript aplikace:

Příklad – Obslužný skript aplikace s validátory regulárních výrazů

```
<script runat="server">
    Sub zkontroluj(ByVal sender As Object, ByVal e As EventArgs)
        If (Page.IsValid) Then
            lblInfo.Text = "Stránka je validní"
        Else
            lblInfo.Text = "Stránka není validní"
        End If
    End Sub
</script>
```



Obr. 6.15 Opravené údaje jsou již validní

6.4 Vlastní tvorba validace

Pokud potřebujeme, aby vstupní data vyhovovaly nějaké speciální podmínce, na kterou nám běžné validátory nestačí, máme k dispozici vlastní validátor. *CustomValidator* umožňuje definovat vlastní validační obsluhu jak na straně klienta, tak na straně serveru.

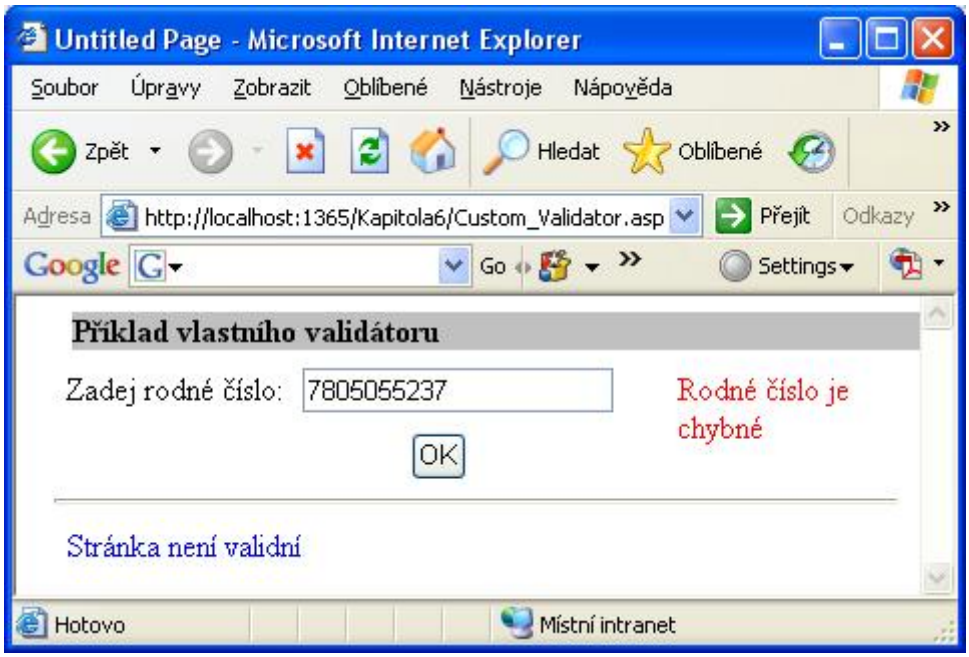
Jako příklad zvolíme formulář, do kterého uživatel vloží rodné číslo. Abychom měli jistotu, že rodné číslo není zadané náhodně, zkontrolujeme, zda-li je dělitelné číslem 11.

Obr. 6.16 Formulář s CustomValidátorem

Příklad – Formulář s vlastním validátorem

```
<form id="form1" runat="server">
   <asp:TextBox ID="tbRC" runat="server" ></asp:TextBox>
   <asp:Label ID="Label1" runat="server" Text="Zadej rodné číslo:">
</asp:Label>
  <asp:CustomValidator ID="CstV1" runat="server"
       OnServerValidate="ServerValidate"
       ControlToValidate="tbRC" ErrorMessage="Rodné číslo je chybné"
       Display="Dynamic"></asp:CustomValidator>
   <asp:Button ID="Button1" runat="server" Text="OK"
```

```
            OnClick="zkontroluj" />
        <asp:Label ID="Label2" runat="server" Text="Příklad vlastního
            validátoru"></asp:Label>
         <asp:Label ID="lblInfo" runat="server" ></asp:Label>
         <asp:RequiredFieldValidator ID="RFV1" runat="server"
           ControlToValidate="tbRC"
           ErrorMessage="*"></asp:RequiredFieldValidator>
    <hr/>
</form>
```



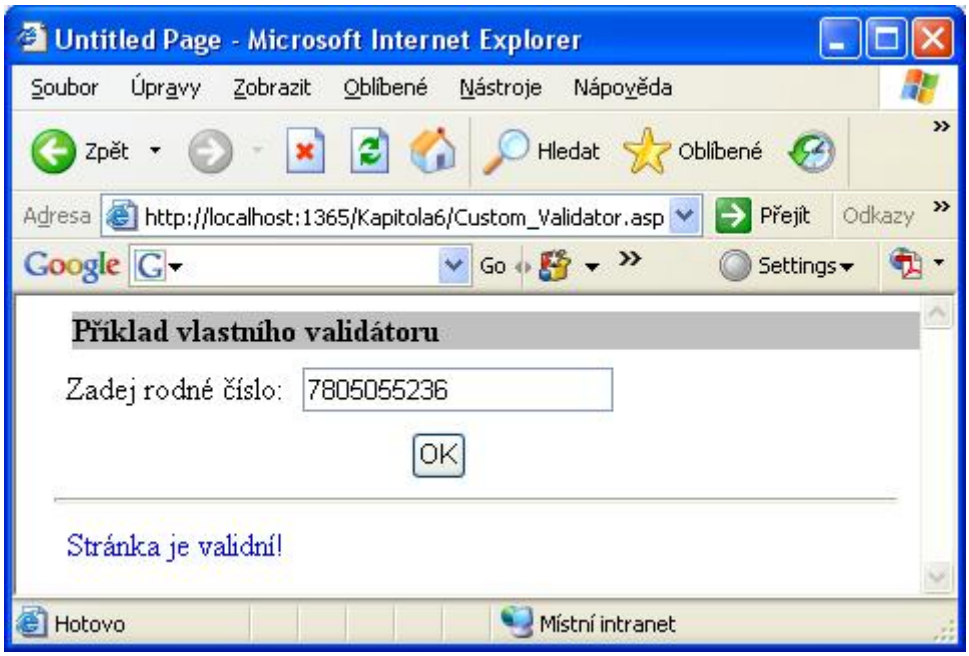
Obr. 6.17 Rodné číslo nevyhovující podmínce

Jistě jste si všimli vlastnosti ovládacího prvku *onServerValidate*. Musíme pro tuto událost vytvořit obslužný skript. Ten bude kontrolovat právě dělitelnost jedenácti. Nejprve ovšem musíme získanou hodnotu, která je zapsaná v textboxu, přetypovat na číslo.

Příklad – Skript pro obsluhu vlastní validace

```
<script runat="server">
    Sub zkontroluj(ByVal sender As Object, ByVal e As EventArgs)
        If (Page.IsValid) Then
            lblInfo.Text = "Stránka je validní!"
        Else
            lblInfo.Text = "Stránka není validní"
        End If
    End Sub

    Sub ServerValidate(ByVal sender As Object, ByVal value As
         ServerValidateEventArgs)
        Dim num As Long
        ' je rodne cislo delitelne 11?
        If Long.TryParse(value.Value, num) Then
            value.IsValid = (num Mod CLng(11) = 0)
               Else
            value.IsValid = False
        End If
    End Sub
</script>
```

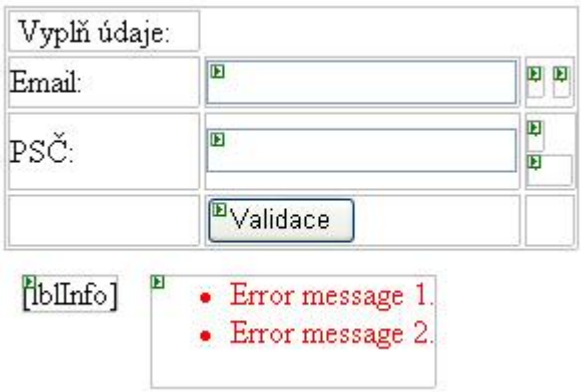
Protože rodné číslo je dlouhé a již mimo rozsah datového typu *Integer*, musíme řetězec přetypovat na *Long*. Ke zjištění dělitelnosti použijeme funkci modulo a porovnáme zbytek po dělení s nulou.



Obr. 6.18 Již správně vyplněný formulář

6.5 Validační souhrn

Posledním ovládacím prvkem z probírané kategorie je *ValidationSummary*. Tento prvek již neslouží k validaci dat, nýbrž k zobrazení validačního souhrnu. Souhrn hodnot vlastnosti *ErrorMessage* všech validátorů je zobrazen přehledně na stránce, kde umístíme ovládací prvek *ValidationSummary* nebo v novém okně, otevřeném pomocí javascriptu. Možné jsou i obě varianty zobrazení. Na obrázku 6.19 je zobrazen validační souhrn.

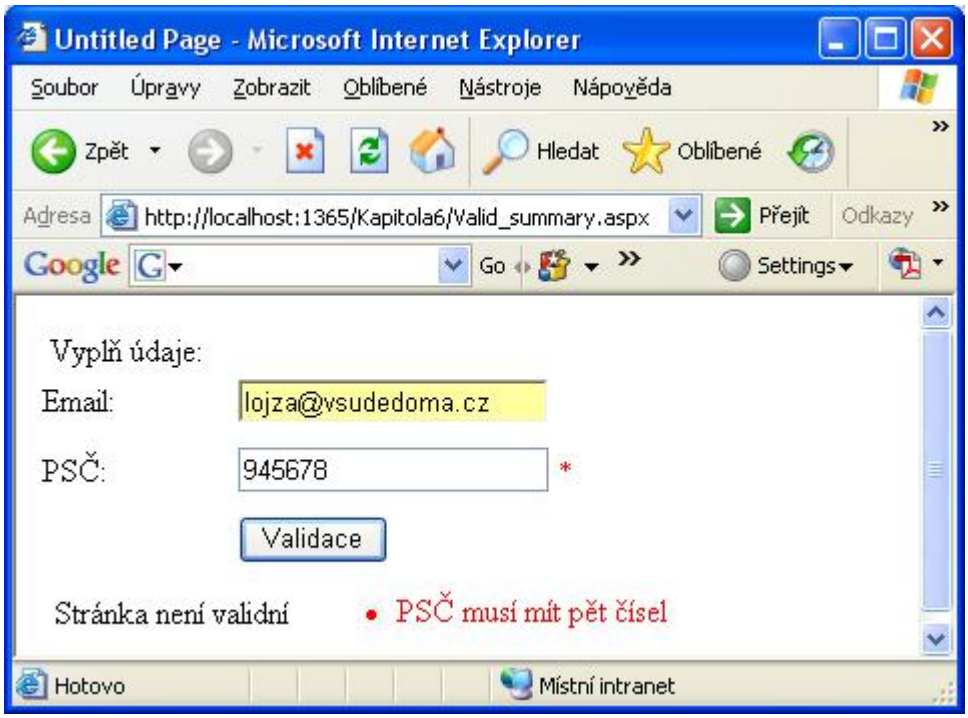


Obr. 6.19 Formulář s validačním souhrnem

Pojem k zapamatování: Typ zobrazení souhrnné validace.

Pokud chceme zobrazit validační souhrn v novém okně, nastavíme vlastnost *ShowMessageBox* na hodnotu True. Pokud chceme souhrn zobrazit ve stránce, nastavíme vlastnost *ShowSummary*.

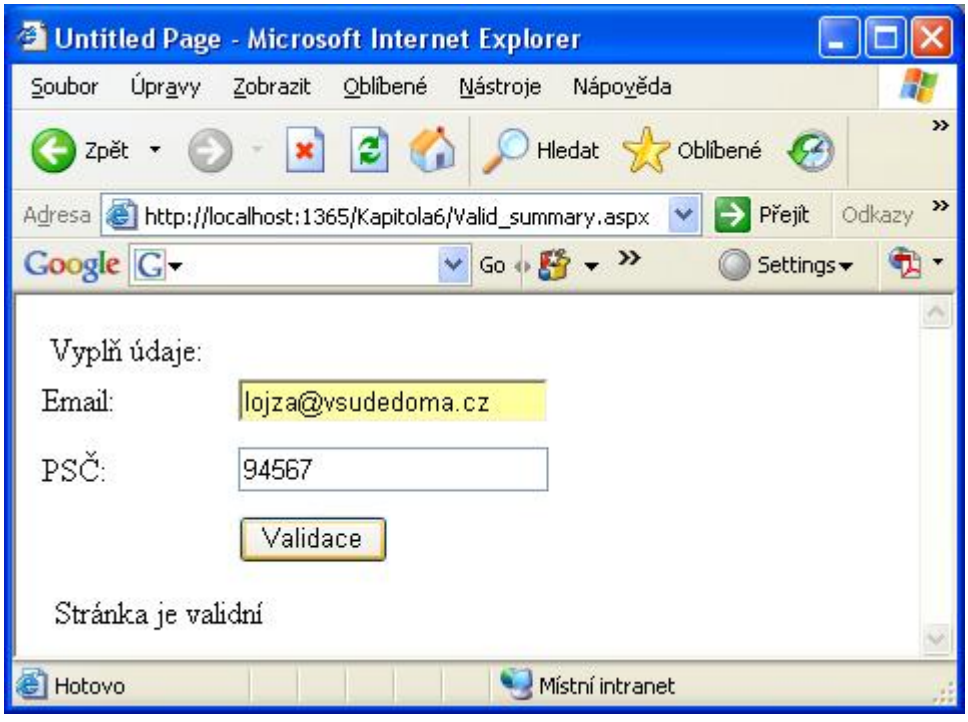
Validační souhrn jsme vložili do formuláře našeho příkladu s validátory regulárních výrazů. V ukázce zdrojového kódu vidíme u ovládacího prvku *ValidationSummary* pouze atribut *runat="server"*. Hodnota *ShowSummary = "True"* je nastavena defaultně, stejně tak jako *DisplayMode ="BulletList"*.



Obr. 6.20 Validační souhrn v akci

Vlastnost *DisplayMode* určuje styl zobrazení a může nabývat těchto tří hodnot:

- BulletList (zobrazení s puntíkem před položkou seznamu).
- List (zobrazení pouze seznamu položek).
- Single Paragraph (zobrazení všech chybových hlášení za sebou v jednom odstavci).



Obr. 6.21 Správně vyplněná stránka

Příklad – Jednoduchý validační souhrn

```
<asp:ValidationSummary ID="ValidationSummary1" runat="server"/>

<script runat="server">
    Sub zkontroluj(ByVal sender As Object, ByVal e As EventArgs)
        If (Page.IsValid) Then
            lblInfo.Text = "Stránka je validní"
        Else
            lblInfo.Text = "Stránka není validní"
        End If
    End Sub
</script>
```

Ke kapitole 6 je přiřazena demonstrační animace č. 5

Animace č. 5 obsahuje postup při vytváření řešených příkladů se všemi probíranými validačními ovládacími prvky.

Shrnutí kapitoly

V ASP.NET byly vyvinuty validační ovládací prvky, které je možné svázat s ovládacími prvky, které vyžadují nějaký vstup od uživatele a tyto validátory automaticky ověřují platnost na obou stranách, na straně klienta i serveru.

Validačních ovládacích prvků je šest: *<asp:RequiredFieldValidator>* zkontroluje, zda-li ovládací prvek, do kterého se vyplňují data není prázdný. *<asp:CompareValidator>* zkontroluje porovnáním, zda-li požadovaná hodnota vyhovuje předepsané hodnotě. *<asp:RangeValidator>* zkontroluje, zda-li vstupní hodnota vyhovuje určitému povolenému rozsahu hodnot. *<asp:RegularExpressionValidator>* zkontroluje, zda-li vstupní hodnota vyhovuje předepsanému regulárnímu výrazu. Vlastním validátorem *<asp:CustomValidator>* můžeme definovat svou validační logiku a na závěr máme ještě k dispozici souhrnou informaci o chybách ze všech validátorů pomocí *<asp:ValidationSummary>*.

Ověřujeme-li data od uživatele, mějme na paměti, že pokud kontrolujeme data jiným validátorem než je *RequiredFieldValidator* a uživatel ponechá pole prázdné, validace skončí úspěšně. Z výše uvedeného faktu plyne skutečnost, že budeme muset většinou kombinovat pro kontrolu vstupních dat *RequiredFieldValidátor* s dalším validátorem.

Pokud jsou všechny ovládací prvky platné, je hodnota vlastnosti *Page.isValid* rovna *True*. V jediném okamžiku se tak dovíme, zda-li jsou všechny ovládací prvky v pořádku.

ErrorMessage je vlastnost validátoru, která je zobrazena v případě chyby. Pokud tedy vstupní hodnota vázaného ovládacího prvku je prázdná, zobrazí se chybové hlášení. U *RequiredFieldValidátoru* se nejčastěji v praxi vypisovaná chyba oznamuje hvězdičkou.

Vlastnost *Type u CompareValidátoru* a *RangeValidátoru* určuje datový typ, který budeme porovnávat. Dalšími vlastnostmi *CompareValidátoru* jsou *ValueToCompare* pro porovnání s konstantní hodnotou nebo *ControlToCompare* pro porovnání s hodnotou jiného ovládacího prvku. Máme možnost použít pouze jednu z těchto možností. U *RangeValidátoru* definujeme rozsah platnosti vlastnostmi *MinimumValue*

a *MaximumValue*. Tyto hodnoty pak nadefinujeme tak, aby odpovídaly zvolenému datovému typu *Type*.

Regulární výraz je předpis pro textové řetězce. Pomocí něj jsou specifikovány podmínky, kterým řetězce musí vyhovovat. Umožňují vytvářet pravidla pro komplikované tvary řetězců. Jsou jediným nástrojem pro možnost kontroly nad určitou posloupností znaků, které například charakterizují tvar e-mailové adresy, URL adresy, různých telefonních čísel a dalších řetězců, pro které platí určitá pravidla. Regulární výrazy jsou tvořeny metaznaky a kvantifikátory.

Robustní nástroj regulárních výrazů využívá *RegularExpressionValidátor*. Přesto, že se jedná o mocný validátor, jeho použití je jednoduché. Do vlastnosti *ValidationExpression* dosadíme vytvořený regulární výraz.

Pokud potřebujeme, aby vstupní data vyhovovaly nějaké speciální podmínce, na kterou nám běžné validátory nestačí, máme k dispozici vlastní validátor. *CustomValidátor* umožňuje definovat vlastní validační obsluhu jak na straně klienta, tak na straně serveru.

Úkol k řešení 6.1 – Validace různých typů ovládacích prvků

Navrhněte formulář s ovládacími prvky, které lze validovat, ale které nebyly řešeny v ukázkových příkladech.

Úkol k řešení 6.2 – Validace formuláře

Definujte ve formuláři kromě tlačítka pro validaci a odeslání dat na server také tlačítko *Cancel*, kterým zrušíme vyplňování formuláře (U tohoto tlačítka vyřaďte validaci).

Úkol k řešení 6.3 – CompareValidator

Navrhněte vhodný formulář, ve kterém použijete *CompareValidátor* pro kontrolu vstupních dat různých datových typů.

Úkol k řešení 6.4 – CustomValidator

Definujte vlastní validátor pro zadání hodnot pouze lichých čísel do hodnoty 49.

Úkol k řešení 6.5 – RegularExpressionValidator

Definujte kontrolu nějakého vámi určeného silného hesla u *RegularExpressionValidátoru*.

Úkol k řešení 6.6 – Validation Summary

Zobrazte validační souhrn do okna typu *MessageBox*.

Kontrolní otázka 6.1

Které validátory znáte a k čemu slouží?

Kontrolní otázka 6.2

Které ovládací prvky lze a nelze validovat?

Kontrolní otázka 6.3

Jaké datové typy vstupních dat lze validovat a u jakých validátorů?

Kontrolní otázka 6.4

Co jsou to metaznaky a kvantifikátory regulárních výrazů, uveďte alespoň některé z nich.

Kontrolní otázka 6.5

Jaké operátory porovnání definujeme u CompareValidátoru?

Kontrolní otázka 6.6

Definujte pojmy *ControlToCompare, ValueToCompare, ErrorMessage, ValidationExpression* a *onServerValidate.*

7. UŽIVATELSKÉ OVLÁDACÍ PRVKY

Čas ke studiu: 2 hodiny

Cíl Po prostudování této kapitoly budete umět

- definovat důvody pro vytvoření uživatelského ovládacího prvku
- popsat nezbytné pojmy pro vytvoření uživatelského ovládacího prvku
- vytvářet vlastní jednoduché uživatelské ovládací prvky
- vytvářet uživatelské ovládací prvky obsahující skript nebo objekty
- využít uživatelských ovládacích prvků ve vlastní aplikaci

Výklad

Po třech kapitolách, kde jsme téměř vyčerpávajícím způsobem procvičili ovládací prvky, které nabízí Visual Studio, a také všechny validační ovládací prvky, nastala otázka, zda-li nemůžeme vytvořit vlastní ovládací prvek. Právě tomuto tématu je věnována dnešní lekce č. 7.

7.1 Co nás vede k vytvoření uživatelského prvku

Po procvičení téměř všech ovládacích prvků jste již pokročilými návrháři webových formulářů. Zajisté jste si položili otázku: Vyhovují mi opravdu všechny prvky, nepostrádám nějaký? Ano, právě odpovědí na tuto otázku je umožnění vytvoření vlastního ovládacího prvku. Druhým důvodem je jistý stereotyp při vytváření formulářů. Vždyť mnohokrát používáte tytéž prvky a tvoříte na vlas stejné formuláře. Nešlo by tedy nějak tuto činnost urychlit a zdokonalit? Odpovědí je opět možnost tvorby vlastního ovládacího prvku. Jeho výhodou je bezesporu možnost znovupoužití právě tam, kde jste si uvědomili, že tento formulář jste již stoprocentně někdy udělali.

Pojem k zapamatování: Uživatelský ovládací prvek

Uživatelský ovládací prvek je ve své podstatě část web stránky, která může obsahovat HTML kód, webové ovládací prvky i vlastnosti, metody a události. Je uložena zvlášť v souboru s příponou *ascx* a umožňuje opětovné použití ve více stránkách.

Zajímavým, ale jistě použitelným způsobem je vytvoření aspx stránky a poté zjišťování, které prvky budou znovupoužitelné. Ostatní jednoduše vymazat. Až budeme jisti, že zůstaly na stránce prvky, které jsou jistě pro nás vícekrát použitelné, převedeme stránku na ovládací prvek. Musíme ale dodržet následující postup:

1. Odstranit elementy <*html*>, <*body*> a <*form*>, protože tyto značky mohou být ve zdrojovém kódu stránky pouze jednou, takže je nemůže obsahovat uživatelský ovládací prvek, který pak budeme registrovat do hlavní stránky.
2. Odstranit direktivu *Page*, popřípadě ji změnit na direktivu *Control*, se všemi přípustnými atributy.
3. Zaměnit příponu *aspx* na *ascx*.

A jsme v podstatě hotovi. Důvodem proč můžeme uvedený postup použít je ten, že třída *Page* i *UserControl* se dědí ze stejné třídy *TemplateControl* a tudíž mají mnoho stejných metod a událostí.

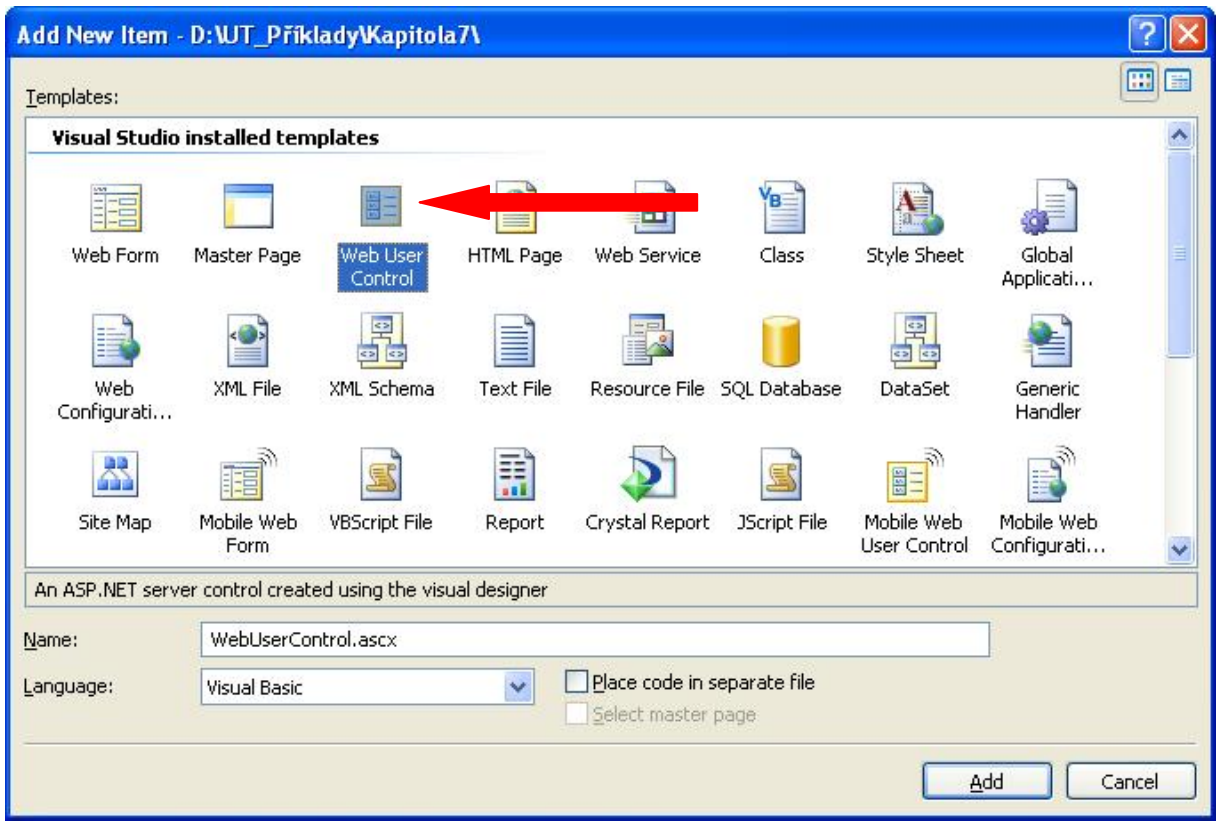
7.2 Příklady tvorby uživatelských ovládacích prvků

Základem stránky, ve které budeme chtít využít ovládací prvek, který jsme vytvořili, je direktiva *Register*. Ovládací prvek se svým názvem musíme zaregistrovat a vytvořit pro něj vlastní prefix.

Příklad – Registrace uživatelského ovládacího prvku

```
<%@ Page Language="VB"%>
<%@ Register TagPrefix="Můj_Definovaný_Prefix "
TagName="Muj_Definovaný_TagLabel" Src="Soubor_S_Mým_Prvkem.ascx" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Stránka s uživatelskými prvky</title>
```

Samotný uživatelský ovládací prvek vytvoříme jako nový soubor v projektu. Ve vzorových šablonách máme možnost zvolit ikonku *Web User Control*.



Obr. 7.1 Založení uživatelského ovládacího prvku

V našem projektu si vytvoříme celkem 4 vlastní ovládací prvky. Prvním prvkem je vlastní *label* naší Virtuální cestovní kanceláře. Abychom nemuseli na každé stránce, kde chceme použít popisek s cestovní kanceláří, pracně popisovat ovládací prvek *label*, nastavovat jeho rozměry, barvy a další vlastnosti, jednoduše jej vytvoříme jen jednou a uložíme jako *Web User Control*. To je nejjednodušší způsob vytvoření vlastního ovládacího prvku, neobsahuje žádný skript, pouze HTML kód.

Ovládací prvek obsahuje pouze direktivu *Control* a serverový ovládací prvek L*abel*.

Příklad – Jednoduchý uživatelský ovládací prvek

```
<%@ Control Language="VB" ClassName="Muj_label" %>

<asp:Label ID="Label1" runat="server" ForeColor="Blue" Height="18px"
Text="Cestovní kancelář Virtual Tour - Vesmír máte jako na dlani
" Width="407px" BackColor="#E0E0E0"></asp:Label>
<asp:Label ID="Label2" runat="server" BackColor="#E0E0E0"
ForeColor="Red"  Text="© Virtual Tour 2007" Font-
Bold="True"></asp:Label>
```

Druhým uživatelským prvkem, který si ukážeme, budou digitální hodiny. Máme-li například nějaký oblíbený skript, který často používáme ve svých stránkách, nic nám nebrání z něj udělat vlastní ovládací prvek. Co je na tom, že je třeba v javascriptu. Vytvoříme tedy *Label* a do něj zajistíme zobrazení aktuálního času, který se bude neustále obnovovat pomocí javascriptu.

Příklad – Uživatelský ovládací prvek se scriptem

```
<%@ Control Language="VB" ClassName="MujPrvek" %>

<script runat="server">

</script>
<script type="text/javascript">
      <!–
      function ShowTime(){
        // Zjistí se aktuální čas
        var D = new Date;
        // Z časového údaje se zjistí počet
        // hodin, minut a sekund
        var H = D.getHours();
        var M = D.getMinutes();
        var S = D.getSeconds();

        // Vytvoří se řetězec obsahující celý čas
        // U minut a sekund se přidají případné
        // uvozující nuly (pokud jich je méně jak 10)
        var StrTime = H + ":" +
          (M<10 ? "0"+M : M) + ":" +
          (S<10 ? "0"+S : S);
        // Změní se obsah značky daného jména
        Clock.innerText = StrTime;
        // Aktualizace hodin se provede za sekundu
        setTimeout ("ShowTime()", 1000);
      }
      // →
    </script>

<asp:Label ID="Label1" runat="server"  Text="Aktuální čas:">
</asp:Label>
<b id="Clock">00:00:00</b>
```

Líbí se vám analogové hodiny v prohlížeči? Je jich opravdu mnoho a jejich implementací se tedy nemusíme zabývat. Následující ovládací prvek obsahuje cestu ke vzdálenému javascriptu.

Příklad – Uživatelský ovládací prvek s odkazem na analogové hodiny

```
<%@ Control Language="VB" ClassName="Analog" %>

<script type="text/javascript">
var clocksize=100;
</script>
<script type="text/javascript"  src="http://gheos.net/js/clock.js">
</script>
```

Posledním vlastním prvkem, který si umístíme do stránky, bude popisek se zobrazením aktuálního data. Použijeme tedy přímo objekt *Date* a vhodnou metodou datum přepíšeme do ovládacího prvku *Label*.

Příklad – Uživatelský ovládací prvek s objektem Datum

```
<%@ Control Language="VB" ClassName="datum" %>

<script runat="server">
    Sub Page_Load(ByVal obj As Object, ByVal e As EventArgs)
        lblDatum.Text += Date.Now.Date.ToString("dddd dd MMMM yyyy")
    End Sub

</script>

<asp:Label ID="lblDatum" runat="server" Font-Bold="True"
ForeColor="Blue" Text="Dnes je "></asp:Label>
```

Máme tedy vytvořené 4 ovládací prvky, nyní je vložme do hlavní stránky aplikace. Nejprve musíme ovládací prvky zaregistrovat. Zvolíme jednotný prefix, dopíšeme názvy prvků a jejich zdrojové soubory.

Příklad – Direktivy stránky s uživatelskými ovládacími prvky

```
<%@ Page Language="VB"%>
<%@ Register TagPrefix="MyAsp" TagName="MujLabel"
Src="MujLabel.ascx" %>
<%@ Register TagPrefix="MyAsp" TagName="MujPrvek"
Src="MujPrvek.ascx" %>
<%@ Register TagPrefix="MyAsp" TagName="Analog" Src="Analog.ascx" %>
<%@ Register TagPrefix="MyAsp" TagName="Dnes" Src="datum.ascx" %>
```

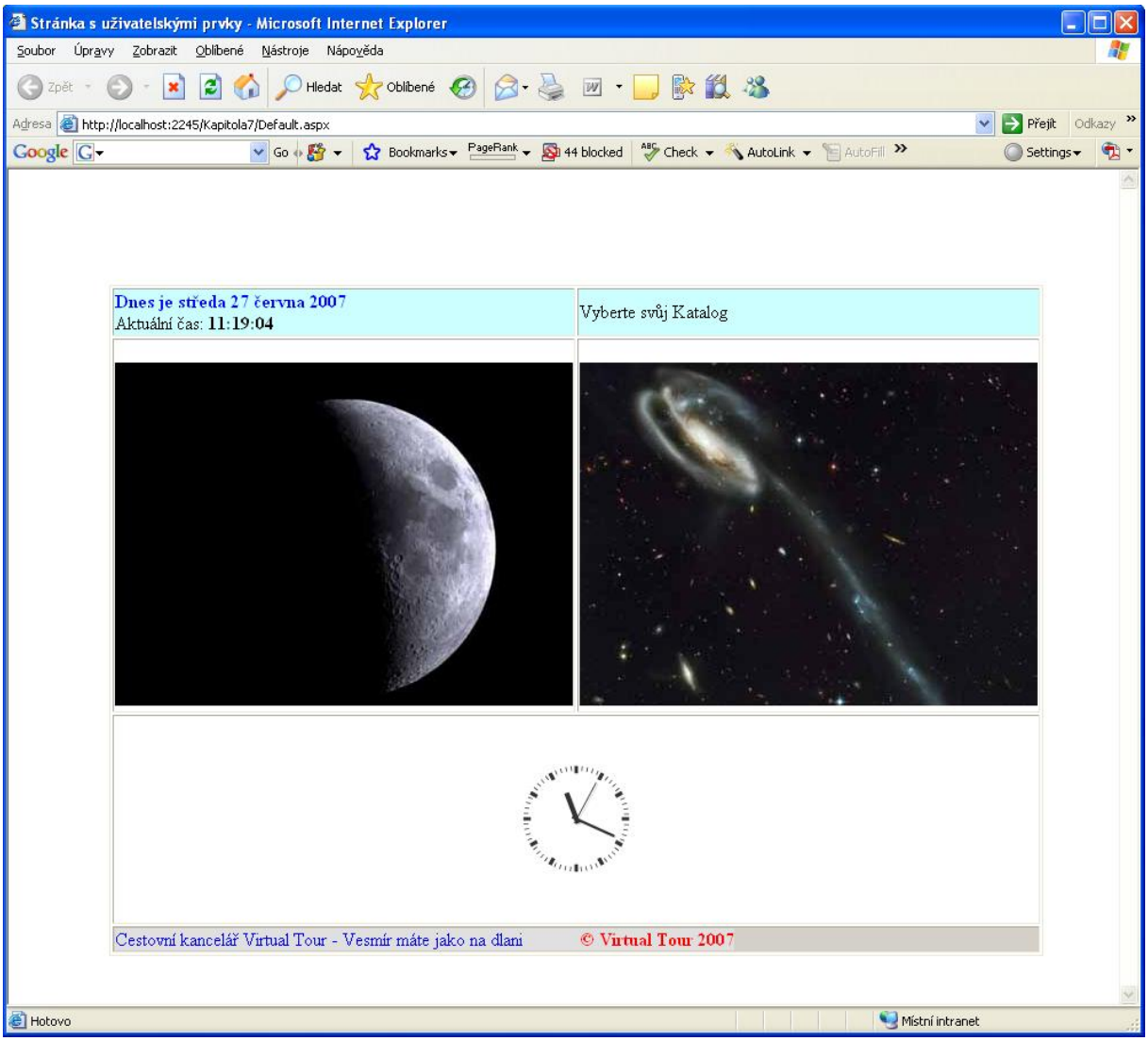
Výslednou aplikaci vytvoříme pouze ilustrativně, jistě si dokážete představit jakoukoliv stránku, která obsahuje elementy, jež jsou i na dalších stránkách aplikace, které představují uživatelské ovládací prvky. Proto vytvoříme pouze jednoduchou kostru aplikace s tabulkou, do níž kromě zmíněných uživatelských ovládacích prvků, o které nám jde především, umístíme ještě *ImageButtony*, reprezentující přechody na katalogy naší virtuální cestovní kanceláře. Formulář naší aplikace bude vypadat takto:

Příklad – Formulář stránky s uživatelskými ovládacími prvky

Zdrojový kód aplikace již můžeme vytvořit podle svého uvážení. V případě, že použijeme prefix `<MyAsp:`, IntelliSense nám nabídne 4 uživatelské ovládací prvky: `MujLabel`, `Analog`, `MujPrvek` a `Dnes`.

Příklad – Zdrojový kód aplikace s uživatelskými ovládacími prvky

```
<table border="1" frame="border">
 <tr>
 <td><MyAsp:Dnes ID="dnes1" runat="server" /><br /><MyAsp:MujPrvek
      ID="MP1" runat="server" /></td>
 <td> Vyberte svůj Katalog</td>
 </tr>
 <tr>
 <td><asp:ImageButton ID="IB1" runat="server"
ImageUrl="~/images/zima.jpg"  /> </td>
 <td><asp:ImageButton ID="IB2" runat="server"
ImageUrl="~/images/leto.jpg"  /></td>
 </tr>
 <tr><td align="center" colspan="2" ><MyAsp:Analog ID="A1"
runat="server" /></td>
 </tr>
 <tr>
 <td colspan="2"> <MyAsp:MujLabel ID="ML1" runat="server" /></td>
 </tr>
 </table>
```



Obr. 7.2 Výsledná aplikace se 4 uživatelskými ovládacími prvky

Ke kapitole 7 je přiřazena demonstrační animace č. 6

Animace č. 6 obsahuje tvorbu uživatelských ovládacích prvků, které byly vytvořeny v této kapitole.

Shrnutí kapitoly

Uživatelský ovládací prvek je ve své podstatě část web stránky, která může obsahovat HTML kód, webové ovládací prvky i vlastnosti, metody a události. Je uložena zvlášť v souboru s příponou *ascx* a umožňuje opětovné použití ve více stránkách.

Pokud budeme chtít vytvořit ovládací prvek z již existující *aspx* stránky, musíme dodržet následující postup:

1. Odstranit elementy <*html*>, <*body*> a <*form*>, protože tyto značky mohou být ve zdrojovém kódu stránky pouze jednou, takže je nemůže obsahovat uživatelský ovládací prvek, který pak budeme registrovat do hlavní stránky.

2. Odstranit direktivu *Page*, popřípadě ji změnit na direktivu *Control*, se všemi přípustnými atributy.

3. Zaměnit příponu *aspx* na *ascx*.

Základem stránky, ve které budeme chtít využít ovládací prvek, který jsme vytvořili, je direktiva *Register*. Ovládací prvek se svým názvem musíme zaregistrovat a vytvořit pro něj vlastní prefix. Dalším atributem direktivy je název ovládacího prvku a odkaz na soubor ascx s ovládacím prvkem.

```
<%@ Page Language="VB"%>
<%@ Register TagPrefix="Můj_Definovaný_Prefix"
TagName="Muj_Definovaný_TagLabel"
Src="Soubor_S_Mým_Prvkem.ascx" %>
```

Máme-li zaregistrovaný vlastní prefix, pokud jej použijeme, Visual Studio jej rozpozná a pomocí Intellisense nám nabídne všechny zaregistrované uživatelské ovládací prvky s tímto prefixem.

Příklad registrace a použití uživatelského ovládacího prvku:

```
<%@ Register TagPrefix="MyAsp" TagName="Dnes" Src="datum.ascx" %>

<MyAsp:Dnes ID="dnes1" runat="server" />
```

Uživatelské prvky často tvoříme jako triviální serverové ovládací prvky jako např. *Label*, protože mnohdy jej využijeme vícekrát a nemusíme pokaždé nastavovat vlastnosti jako velikost, font, barvy apod. Můžeme ale také vytvořit uživatelský ovládací prvek se skriptem, s využitím objektů ASP.NET nebo s vlastními metodami a událostmi.

Úkol k řešení 7.1 – Vytvoření aplikace s uživatelskými ovládacími prvky

Navrhněte aplikaci, která má na více stránkách shodné prvky. Kolekci těchto prvků předělejte na uživatelské ovládací prvky (např. část formuláře, navigační lištu či popisek).

Úkol k řešení 7.2 – Vytvoření uživatelského prvku

Vytvořte ovládací prvek konverzí z aspx stránky.

Úkol k řešení 7.3 – Vytváření uživatelských ovládacích prvků

Navrhněte nějaký užitečný uživatelský ovládací prvek a vložte jej do aplikace.

Kontrolní otázka 7.1

K čemu slouží uživatelské ovládací prvky?

Kontrolní otázka 7.2

Jak vytváříme uživatelské ovládací prvky?

Kontrolní otázka 7.3

Jak byste rozdělili uživatelské ovládací prvky?

Kontrolní otázka 7.4

Lze vytvořit uživatelský ovládací prvek nějakou konverzí aspx stránky?

Kontrolní otázka 7.5

Jakým způsobem vkládáme uživatelské ovládací prvky do aplikace?

8. PRÁCE S DATABÁZEMI V ASP.NET

Čas ke studiu: 3 hodiny

Cíl Po prostudování této kapitoly budete umět

- definovat pojem poskytovatele dat
- popsat základní třídy poskytovatele dat
- vytvořit v průzkumníku serveru nové datové připojení
- vytvářet databázové tabulky přímo ve Visual Studiu
- zobrazit v prohlížeči data z databázových tabulek
- charakterizovat ovládací prvek Grid View pro zobrazení dat
- srovnat práci s Grid View a předchozím Data Gridem z ASP.NET 1.0
- zajistit stránkování a řazení dat v zobrazené databázové tabulce
- zajistit editaci a mazání dat z tabulky v prohlížeči
- naimplementovat vkládání dat do databáze z web aplikace

Výklad

V této osmé kapitole se dostáváme k jednomu z témat, které je pokládáno za velice důležité, ne-li nejdůležitější, a to, jak zobrazit data z databázových tabulek do prohlížeče. S narůstajícími potřebami zákazníků se v dnešní době bez tohoto tématu neobejdeme. Přesto, že náš kurz je krátký a musíme toho zvládnout co nejvíce, budeme práci s databázemi věnovat tři kapitoly. V této kapitole začneme rychlým úvodem, jak vytvořit databázové připojení a jak co nejrychleji poskytnout webovému prohlížeči data, v dalších kapitolách si ukážeme ovládací prvky pro práci s daty a některé pokročilejší techniky jako parametry a uložené procedury.

8.1 Databázové připojení a poskytovatelé dat

Přístup k datům v aplikacích .NET je založen na vlastní technologii *.NET Frameworku* nazývané ADO.NET. Tato technologie zajišťuje komfortní připojení a správu dat ve všech typech aplikací, ať už desktopových či webových.
Poskytovatel dat neboli *provider* je propojovacím článkem mezi databázovým zdrojem dat a aplikací, která data zobrazuje. Úkolem *providera* je nejen navázat spojení s konkrétním zdrojem dat, ale také vykonávat na něm SQL příkazy a získávat data. *.NET Framework* disponuje čtyřmi základními poskytovateli:

- SQL Server Provider - Poskytuje přístup k databázím SQL Serveru 7.0 a novějším.
- OLE DB Provider - Poskytuje přístup zdroji dat, který obsahuje nějaký ovladač OLE DB, tedy i starší verze SQL Serveru.
- Oracle Provider - Poskytuje přísup k databázím Oracle (verze 8 a novější).
- ODBC Provider - Poskytuje přístup zdroji dat s ovladačem ODBC.

Poskytovatel dat je sadou tříd optimalizovaných pro každého providera, to znamená, že pro připojení k SQL serveru budeme používat *SQLConnection*, zatímco pro připojení ke zdroji dat s ovladačem OleDB, jako například Microsoft Access budeme používat *OleDBConnection*. Takovým způsobem budeme rozlišovat tyto základní třídy pro každého providera:

- Connection (připojení) – Nejprve se musíme ke zdroji dat připojit.
- Command (příkaz) – Na zdroji dat vykonáváme SQL příkaz.
- Data Reader (čtenář dat) – Získaná data SQL příkazem přečteme.
- Data Adapter (datový adaptér) – Naplňuje datové sady získanými daty a promítá změny do zdroje dat.

Třídy ADO.NET jsou samozřejmě rozděleny do jmenných prostorů podle svých poskytovatelů. Obecné třídy jsou uloženy ve jmenném prostoru *System.Data*. Na začátku každé *aspx* stránky, kde budeme třídy ADO.NET využívat, nesmíme zapomenout jmenné prostory naimportovat. To si samozřejmě ukážeme v řešených příkladech a přiložené animaci.

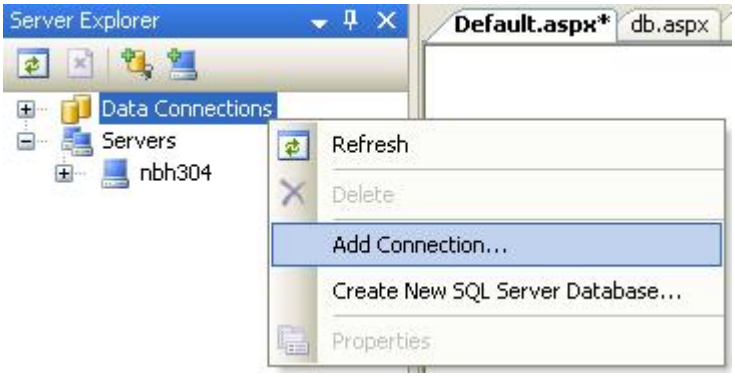
Jmenný prostor	Charakteristika
System.Data	Obsahuje třídy dat, které vytvářejí sloupce, řádky, tabulky, relace, sady dat, datové pohledy (views) a rozhraní.
System.Data.Common	Obsahuje základní třídy ADO.NET (ty jsou většinou abstraktní), definují základní funkcionality ADO.NET. Poskytovatelé dat z těchto tříd dědí.
System.Data.OleDb	Obsahuje třídy pro použití k připojení ke zdroji dat OleDB. Patří sem OleDbConnection, OleDbCommand, OleDbDataReader a OleDbDataAdapter.
System.Data.SqlClient	Obsahuje třídy pro použití k připojení k databázi Microsoft SQL Server. Patří sem SqlConnection, SqlCommand, SqlDataReader aSqlDataAdapter.
System.Data.OracleClient	Obsahuje třídy pro použití k připojení k databázi Oracle. Třídy, které sem patří, mají prefix Oracle.
System.Data.Odbc	Obsahuje třídy pro použití k připojení k datovému zdroji s ovladačem ODBC. Konfigurují se v ovládacích panelech prostřednictvím nástroje pro správu – Datové zdroje.
System.Data.SqlTypes	Obsahuje struktury, které odpovídají nativním datovým typům v SQL Serveru.

Tab. 8.1 Jmenné prostory ADO.NET

Pojem k zapamatování: Poskytovatelé dat a jmenné prostory

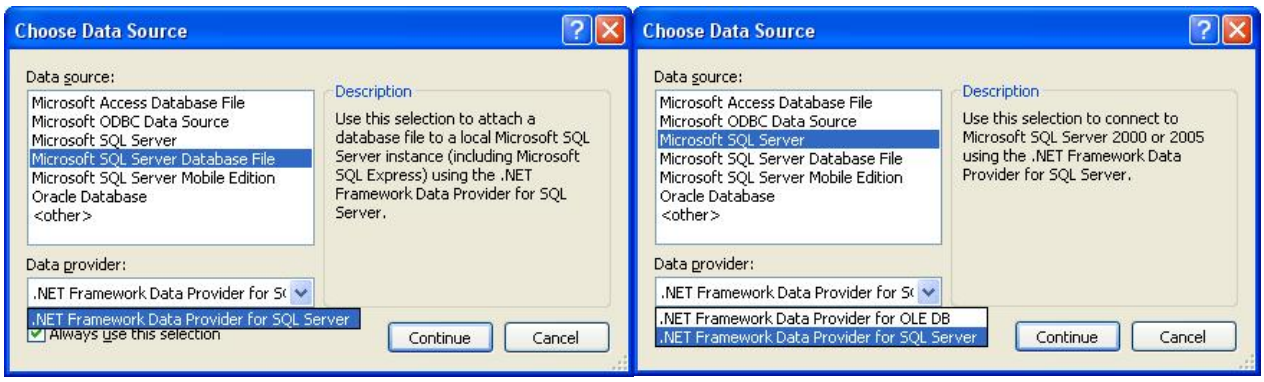
Každý poskytovatel má přidělen svůj jmenný prostor. Pokud budeme chtít využít např. databázi Microsoft Access, budeme využívat třídy OledbConnection, OleDbCommand a pravděpodobně OleDbDataAdapter. Musíme proto naimportovat jmenný prostor System.Data.OleDb.

Mnoho práce při vytváření databázového připojení a konfiguraci datových zdrojů za nás provedou průvodci ve Visual Studiu. Různými výběry z nabízených možností tak omezíme potenciální možnosti chyb a samozřejmě ušetříme mnoho času s psaním kódu. Přesto se někdy psaní kódu nevyhneme, ale nemůžeme přece předpokládat, že složitější aplikaci celou vyklikáme myší. Pro příklady v našem kurzu budeme využívat vestavěný SQL Expess 2005 Server a vystačíme si tak s tvorbou databázových aplikací s instalací Visual Studia, což jistě oceníme, protože odpadnou starosti a případné problémy s instalací SQL Serveru. V okně průzkumníka serveru (*Server Explorer*) vidíme rozbalovací strukturu datových připojení (*Data Connection*). Klikneme pravým tlačítkem myši a zvolíme volbu přidat připojení (viz obr. 8.1).



Obr. 8.1 Vytvoření databázového připojení v server exploreru

Otevře se průvodce, který nám nabízí volbu datového zdroje od databáze Access až po databázi Oracle. V našich příkladech budeme pracovat s databázovým SQL souborem *mdf*, který bude přímo součástí našeho projektu, ve kterém ho také vytvoříme. Pro každý typ datového zdroje máme k dispozici i konkrétního poskytovatele dat.

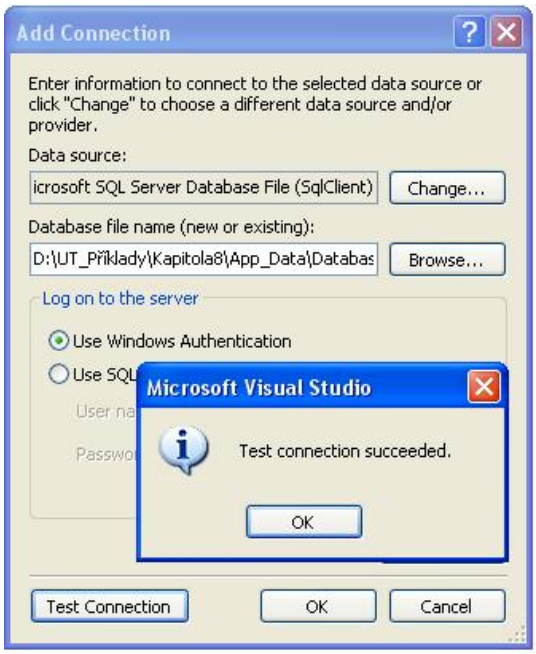


Obr. 8.2 Volba datového zdroje společně s providerem dat

Po výběru datového zdroje a potvrzení tlačítkem *Continue* máme možnost si vytvořené připojení otestovat (viz. obr. 8.3). Microsoft Visual Studio naváže spojení s datovým zdrojem a oznámí úspěšné připojení či potenciální chybu.

Pojem k zapamatování:
Postup při vytvoření datového připojení

1. V našem projektu vytvoříme SQL databázový mdf soubor.
2. V Server Exploreru klikneme na datová spojení (Data Connection) a zvolíme *Add Connection*.
3. Zvolíme datový zdroj z našeho projektu.
4. Otestujeme připojení.



Obr. 8.3 Datový zdroj vybereme z disku a otestujeme připojení

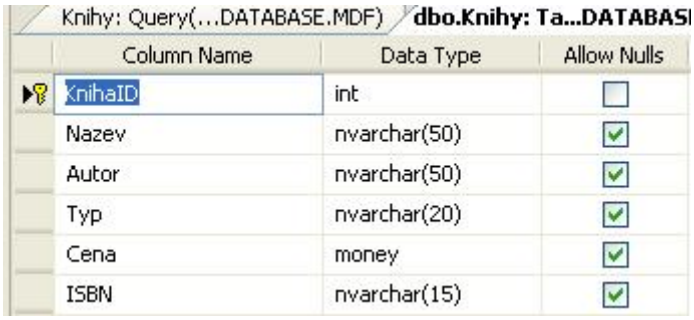
8.2 Vytvoření aplikace s databázovou tabulkou

V následujícím příkladu si ukážeme, jak během několika minut dokážeme zobrazit data z databázové tabulky a dokonce jak komfortním způsobem přizpůsobíme její vzhled a další funkce. To, co bychom v jiných technologiích pracně programovali desítkami řádků kódu uděláme za pomocí ASP.NET 2.0 v několika okamžicích. Nejprve vytvoříme strukturu tabulky, naplníme ji daty a poté několika technikami zajistíme zobrazení dat na webu a to bez napsání řádky kódu!
V předchozím odstavci jsme si vytvořili nové databázové připojení, nyní toto připojení otevřeme a zobrazí se nám v rozbalovací struktuře několik složek, z nichž nás v této chvíli nejvíce bude zajímat položka tabulky (*Tables*). Opět pravým tlačítkem na této položce máme několik možností, zvolíme vytvoření nové tabulky (*Add new Table*).

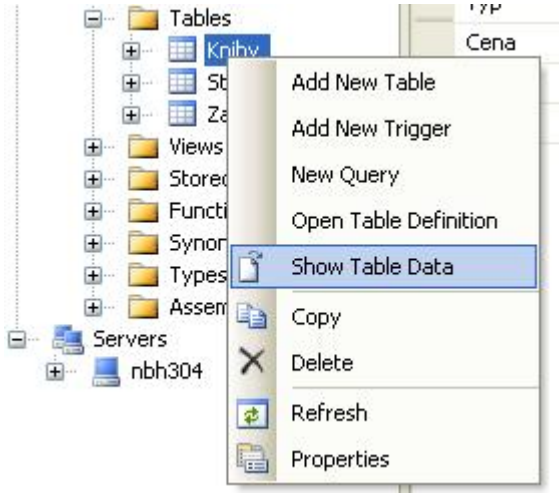
Obr. 8.4 Vytvoření nové databázové tabulky

Pojem k zapamatování:
Vytvoření databázové tabulky

Databázovou tabulku vytváříme přímo ve Visual Studiu. V průzkumníku serveru rozbalíme konkrétní datové propojení a můžeme vytvářet databázové tabulky. Nejprve jejich strukturu, poté je naplníme daty.



Obr. 8.5 Vytvoření položek databáze



Obr. 8.6 Zobrazení či naplnění databázové tabulky

Databázová tabulka se otevře v návrhovém zobrazení třech sloupců, kde zadáváme název sloupce, datový typ a zatrhávacím políčkem volíme možnost, zdali může být položka prázdná či nikoliv. Pravým tlačítkem myši na příslušném řádku tabulky volíme primární klíč. Pokud máte alespoň základní znalosti z oblasti databází, tato fakta znáte, stejně tak jako datové typy, proto se jimi nebudeme zabývat.

Korespondenční úkol – Datové typy

Zopakujte si základní datové typy, se kterými se můžeme setkat při tvorbě databázových tabulek.

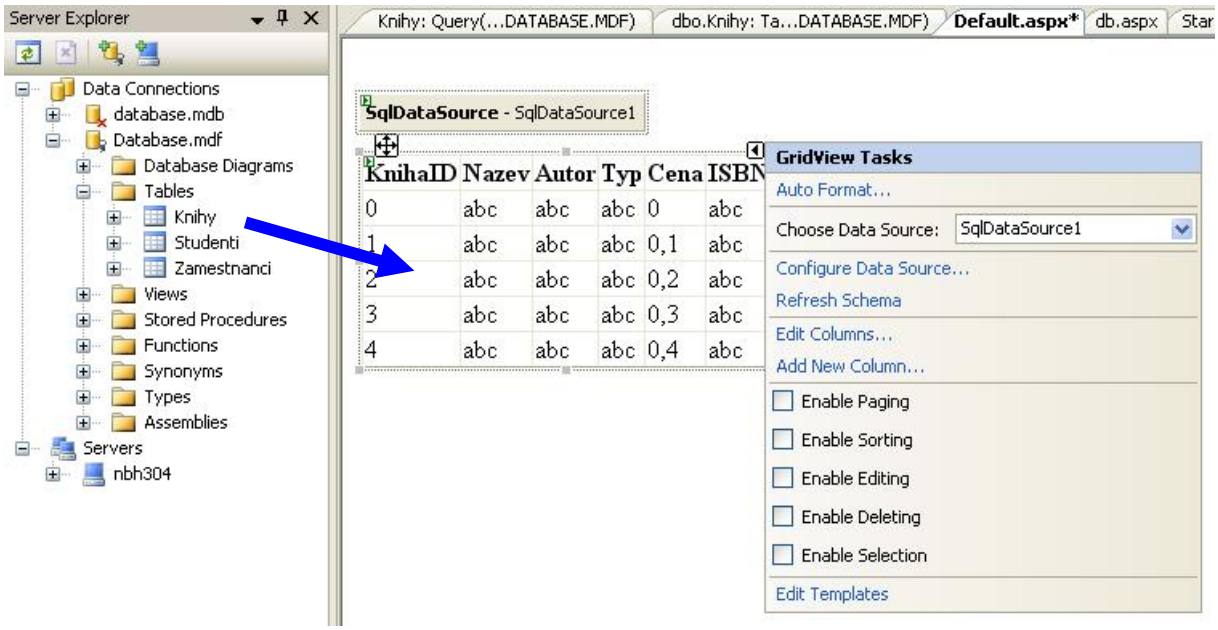
V našem příkladu si vytvoříme tabulku *knihy* se strukturou z obrázku 8.5. Po uložení tabulky již složka *Tables* obsahuje nově vytvořenou tabulku. Pokud budeme chtít tabulku naplnit daty, zvolíme nabídku *Show Table Data* po kliknutí pravým tlačítkem myši (viz. obr. 8.6). Tabulku naplníme daty. Naše pracovní tabulka obsahuje 9 knih a je zobrazena na následujícím obrázku.

Knihy: Query(...DATABASE.MDF) | dbo.Knihy: Ta...DATABASE.MDF) | Default.aspx* | db.aspx | Start Page | Object Browser

KnihaID	Nazev	Autor	Typ	Cena	ISBN
1	Ferda cvičí mraveniště	Ondřej Sekora	Dětská	151,0000	80-88545621
2	Internet pro každého	Pavel Vlk	Počítačová	180,0000	80-88545458
3	Encyklopedie jedovat...	Petr Kozák	Kuchařka	400,0000	80-88578621
4	Jak stloustnout do tý...	Kateřina Buchtová	Kuchařka	250,0000	80-8785621x
5	Honzíkova cesta	Bohumil Říha	Dětská	199,0000	87-88547651
6	Velká kniha hlavolamů	Petr Ježek	Dětská	240,0000	80-88543654
7	Viry jsou všude	Karel Strašák	Počítačová	800,0000	88-88787625
8	Jak zabezpečit domá...	Jiří Lubina	Počítačová	670,0000	87-7896212x
9	Mumínci	Tove Jansen	Dětská	290,0000	80-8986565x

Obr. 8.7 Naše tabulka knih obsahuje tyto položky

Jakým způsobem tuto tabulku zobrazíme v prohlížeči? Odpověď je snadná. Jednoduše přetáhneme tabulku z průzkumníka serveru na stránku v design módu. Tohle velmi rychlé přetažení myší má ale za následek několik akcí s téměř desítkami řádků zdrojového kódu, který bychom jinak museli programovat. Automaticky se vygeneruje ovládací prvek *Grid View* pro zobrazení databázové tabulky, jehož základní neupravená podoba obsahuje sloupce tabulky. Společně s touto podobou vidíme ještě tzv. *tasks* ovládacího prvku, které známe i z předchozích kapitol. U *Grid View* však mají mocné funkce a průvodce, které se netýkají pouze vzhledu prvku.



Obr. 8.8 Tabulku přetáhneme přímo ze server Explorer do aspx stránky v design módu

Pojem k zapamatování: Nejrychlejší zobrazení databázové tabulky

Pouhým přetažením tabulky z průzkumníka serveru do návrhu stránky vygenerujeme ovládací prvek pro zobrazení databázové tabulky v prohlížeči.

Naše velmi rychlé táhnutí myši vygenerovalo zdrojový kód, který bychom mohli psát i ručně a na jeho podobu často budeme brát zřetel při editaci ovládacího prvku, protože přeci jen editovat zdrojový kód je někdy rychlejší, než proklikávání se ve vlastnostech ovládacích prvků. Proto je důležité podobu zdrojového kódu ovládacího prvku *Grid View* znát.

Příklad – Vygenerovaný zdrojový kód ovládacího prvku Grid View

```
<asp:GridView ID="GridView1"
      runat="server"
      AutoGenerateColumns="False"
      BackColor="White"
      BorderColor="#3366CC"
      BorderStyle="None"
      BorderWidth="1px"
      CellPadding="4"
      DataKeyNames="KnihaID"
      DataSourceID="SqlDataSource1"
      EmptyDataText="There are no data records to display.">
<FooterStyle BackColor="#99CCCC" ForeColor="#003399" />
  <Columns>
       <asp:BoundField DataField="KnihaID" HeaderText="KnihaID"/>
       <asp:BoundField DataField="Nazev" HeaderText="Nazev" />
       <asp:BoundField DataField="Autor" HeaderText="Autor" />
       <asp:BoundField DataField="Typ" HeaderText="Typ" />
       <asp:BoundField DataField="Cena" HeaderText="Cena" />
       <asp:BoundField DataField="ISBN" HeaderText="ISBN" />
  </Columns>
  <RowStyle BackColor="White" ForeColor="#003399" />
  <PagerStyle BackColor="#99CCCC" ForeColor="#003399" />
  <HeaderStyle BackColor="#003399" ForeColor="#CCCCFF" />
</asp:GridView>
```

První a velice užitečnou pomůckou v *Grid View Tasks* je položka *Auto Format*. Můžeme zvolit z množství předem naformátovaných tabulek různého barevného vzhledu a samotného provedení. Tato skutečnost je velice cenná, nemusíme pracně formátovat všechny možné položky ovládacího prvku a že jich není málo.

KnihaID	Nazev	Autor	Typ	Cena	ISBN
1	Ferda cvičí mraveniště	Ondřej Sekora	Dětská	151,0000	80-88545621
2	Internet pro každého	Pavel Vlk	Počítačová	180,0000	80-88545458
3	Encyklopedie jedovatých hub	Petr Kozák	Kuchařka	400,0000	80-88578621
4	Jak stloustnout do týdne	Kateřina Buchtová	Kuchařka	250,0000	80-8785621x
5	Honzíkova cesta	Bohumil Říha	Dětská	199,0000	87-88547651
6	Velká kniha hlavolamů	Petr Ježek	Dětská	240,0000	80-88543654
7	Viry jsou všude	Karel Strašák	Počítačová	800,0000	88-88787625
8	Jak zabezpečit domácí síť	Jiří Lubina	Počítačová	670,0000	87-7896212x
9	Mumínci	Tove Jansen	Dětská	290,0000	80-8986565x

Obr. 8.9 Výsledná tabulka v ovládacím prvku Grid View

Při přetažení tabulky v návrhovém zobrazení se nevytvořil pouze ovládací prvek *Grid View*, ale také nezbytný zdroj dat se všemi svými implicitními parametry, které převzal z úvodu příkladu, kdy jsme vytvářeli datové spojení a také samotnou strukturu databázové tabulky. Rovněž tento zdrojový kód je třeba znát, i když byl vygenerován automaticky. Ne všechny atributy zdroje dat ihned použijeme, bez tzv. *Connection Stringu* a *SelectedCommandu* bychom se však neobešli. Jejich funkčnost je patrná již

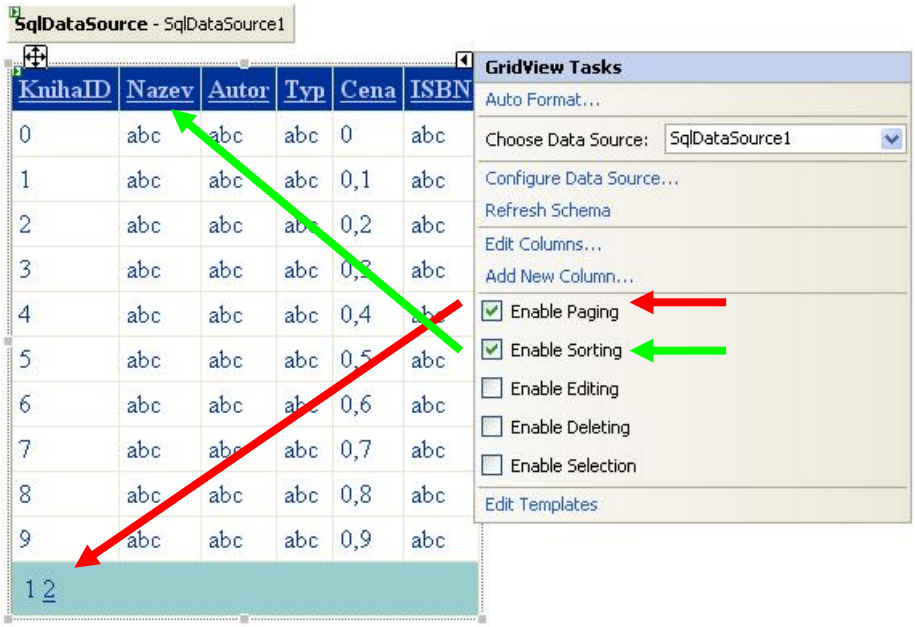
z názvu. Samotný zdroj dat můžeme také editovat ručně, což je rovněž mnohdy rychlejší než samotné spouštění průvodců.

Příklad – Vygenerovaný zdroj dat

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString="<%$ ConnectionStrings:DbaseConnStr1 %>"
DeleteCommand="DELETE FROM [Knihy] WHERE [KnihaID] = @KnihaID"
InsertCommand="INSERT INTO [Knihy] ([KnihaID], [Nazev], [Autor],
[Typ], [Cena], [ISBN]) VALUES (@KnihaID, @Nazev, @Autor, @Typ,
@Cena, @ISBN)"
ProviderName="<%$ ConnectionStrings:DbaseConnStr1.ProviderName %>"
SelectCommand="SELECT [KnihaID], [Nazev], [Autor], [Typ], [Cena],
[ISBN] FROM [Knihy]"
UpdateCommand="UPDATE [Knihy] SET [Nazev] = @Nazev, [Autor] =
@Autor, [Typ] = @Typ, [Cena] = @Cena, [ISBN] = @ISBN WHERE [KnihaID]
= @KnihaID">
</asp:SqlDataSource>
```

8.3 Možnosti stránkování a řazení záznamů

Zobrazením požadované tabulky možnosti ovládacího prvku nekončí. Uživatelé potřebují data v tabulce řadit, stránkovat, vybírat, mazat a editovat. Tyto možnosti rovněž nabízí *Grid View Tasks*. V předchozí verzi ASP.NET 1.0 ovládací prvek *Grid View* nebyl, jeho omezené schopnosti nabízel *Data Grid*, ale tyto funkcionality bylo nutné obsloužit ručně psaným kódem. V posledním odstavci a také v animaci č. 7 si *Data Grid* ukážeme, neboť kompatibilita směrem dolů je zajištěna a s ovládacím prvkem *Data Grid* se jistě mnohdy setkáte. V *Grid View Tasks* vidíme zaškrtávací políčka *Enable Paging* a *Enable Sorting* (viz. obr. 8.10).



Obr. 8.10 Nastavení stránkování a řazení

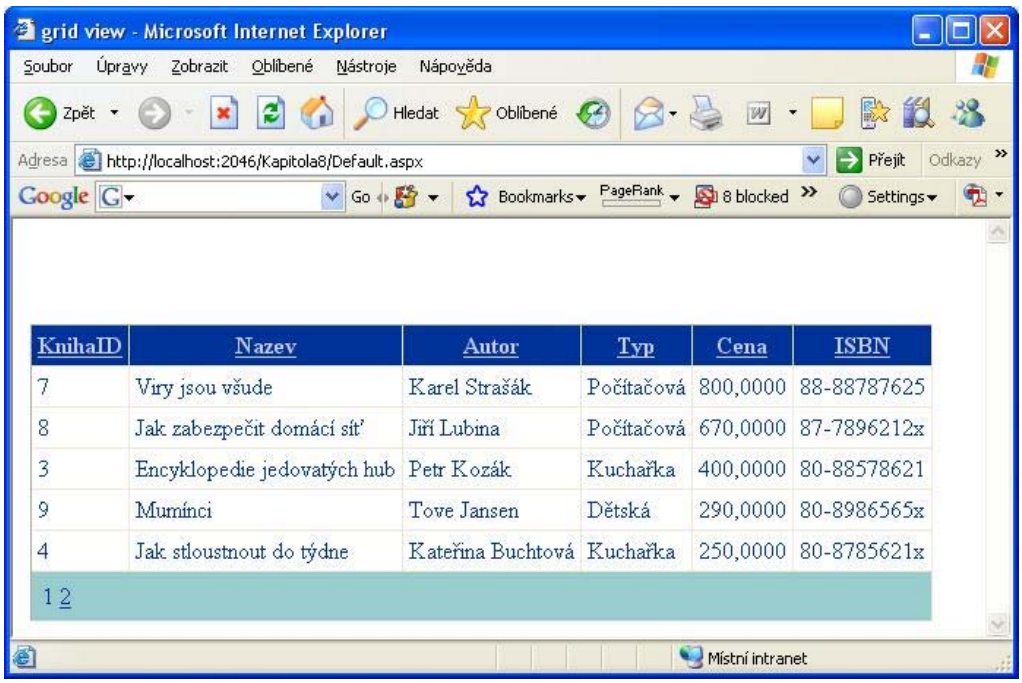
Tyto zaškrtávací políčka označme a podívejme se na atributy prvku `<asp:GridView>`. V podstatě jsme nastavili hodnoty stránkování a řazení na *True* a v atributu *PageSize* počet zobrazovaných záznamů na jedné stránce.

Příklad – Stránkování a řazení Grid View

```
<asp:GridView ID="GridView1"
runat="server"
AutoGenerateColumns="False"
BackColor="White"
BorderColor="#3366CC"
BorderStyle="None"
BorderWidth="1px" CellPadding="4"
DataKeyNames="KnihaID"
DataSourceID="SqlDataSource1"
EmptyDataText="There are no data records to display."
AllowPaging="True" AllowSorting="True" PageSize="5">
```

Korespondenční úkol – Formát stránkovacích tlačítek

Stránkování poznáte podle čísel v patičce tabulky. Zjistěte jakým způsobem můžeme změnit počet zobrazených čísel stránek a jak změníme numerické číslování na šipky.



KnihaID	Nazev	Autor	Typ	Cena	ISBN
7	Viry jsou všude	Karel Strašák	Počítačová	800,0000	88-88787625
8	Jak zabezpečit domácí síť	Jiří Lubina	Počítačová	670,0000	87-7896212x
3	Encyklopedie jedovatých hub	Petr Kozák	Kuchařka	400,0000	80-88578621
9	Mumínci	Tove Jansen	Dětská	290,0000	80-8986565x
4	Jak stloustnout do týdne	Kateřina Buchtová	Kuchařka	250,0000	80-8785621x

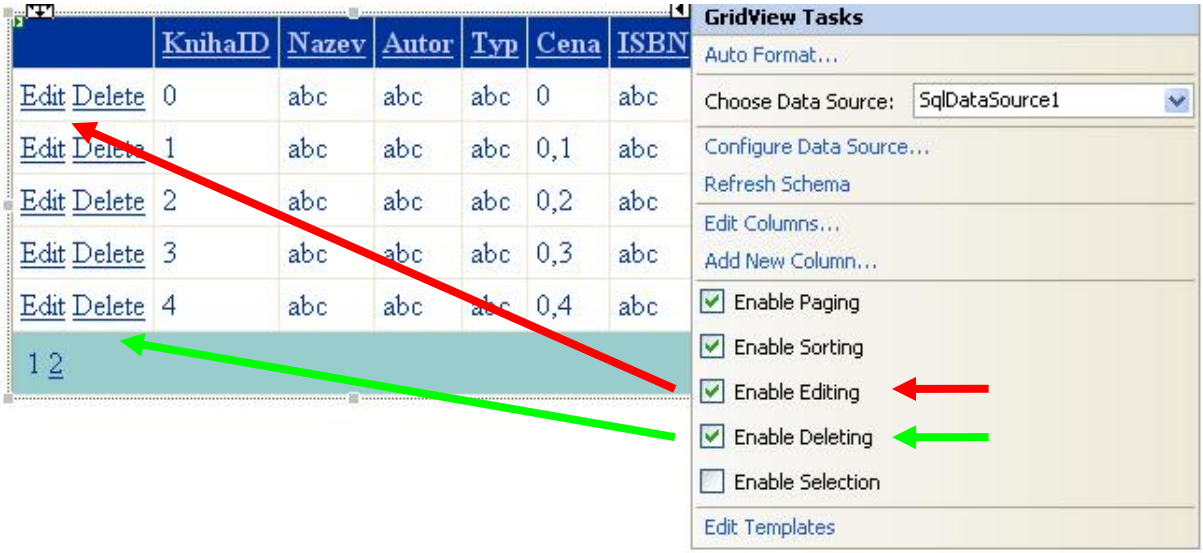
1 2

Obr. 8.11 Tabulku lze stránkovat a řadit data podle sloupců

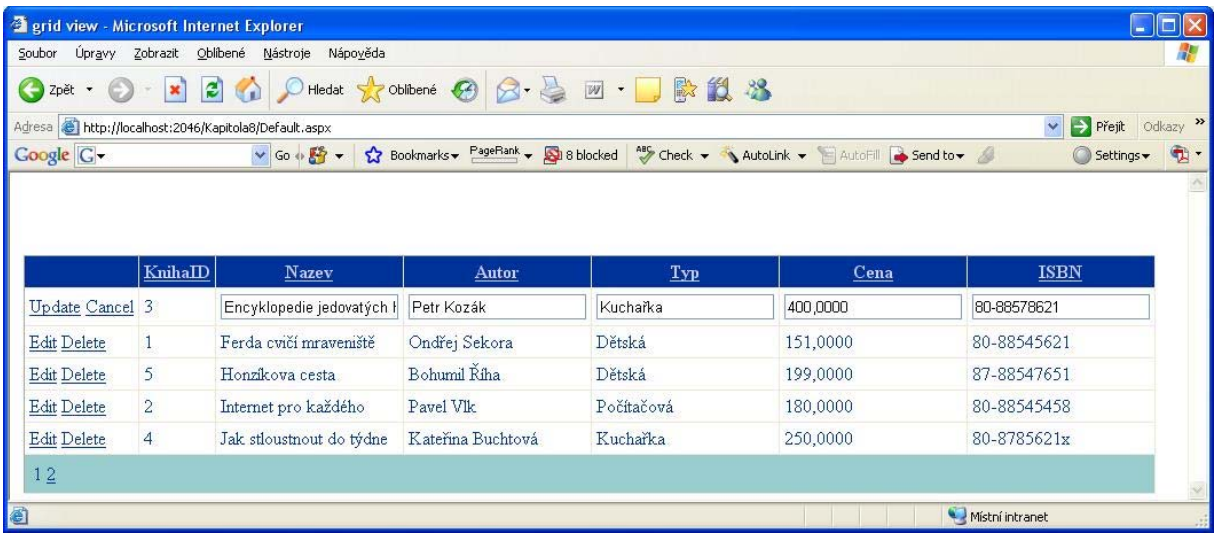
8.4 Editace a mazání dat z databáze

Editace a mazání záznamů nám rovněž nabízí *Grid View Tasks*. Po zatržení příslušných políček se v prvním sloupci tabulky vygenerují *link butony Edit* a *Delete*. Při volbě Edit se editovaný záznam otevře a jednotlivé položky záznamu jsou přístupné v textboxech. Rovněž tlačítko *Edit* se změnilo na dvojici možností *Update* a *Cancel* pro potvrzení změny záznamu či krok zpět. Popsané skutečnosti si rovněž procvičíme v přiložené animaci a výsledek vidíme na obrázku 8.13. Jistě jste si všimli, že vygenerovaný zdroj dat `<asp:SqlDataSource ID="SqlDataSource1">` obsahoval *UpdateCommand* a *DeleteCommand* s položkami začínajícími znakem @. Tímto znakem označujeme parametry a ještě se jim budeme věnovat. Všimněte si vygenerovaných parametrů *UpdateCommand* a

DeleteCommand v příkladu zdrojového kódu. Tyto parametry datového zdroje jsou předány v textboxech ovládacího prvku právě do SQL dotazu, jenž je atributem *Grid View*.



Obr. 8.12 Nastavení editace a smazání záznamů



Obr. 8.13 Záznamy je možno pohodlně editovat i mazat

Příklad – Parametry pro editaci a smazání záznamu

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server" ..atd.
            <UpdateParameters>
                <asp:Parameter Name="Nazev" Type="String" />
                <asp:Parameter Name="Autor" Type="String" />
                <asp:Parameter Name="Typ" Type="String" />
                <asp:Parameter Name="Cena" Type="Decimal" />
                <asp:Parameter Name="ISBN" Type="String" />
                <asp:Parameter Name="KnihaID" Type="Int32" />
            </UpdateParameters>
            <DeleteParameters>
                <asp:Parameter Name="KnihaID" Type="Int32" />
            </DeleteParameters>
        </asp:SqlDataSource>
```

8.5 Vkládání nového záznamu do databáze

Asi netrpělivě čekáte, jak je to s vložením nového záznamu do tabulky. Tohle ovládací prvek *Grid View* dosud bohužel neumí a uvedenou situaci budeme muset vyřešit sami. Mohli bychom se různými způsoby pokoušet o implementaci vkládacího řádku přímo do *Grid View*. Tahle varianta je samozřejmě možná, ale ponecháme ji zvídavějším studentům se zájmem o hlubší problematiku ASP.NET. V našem příkladu si vytvoříme formulář, kde novou položku založíme a pokusíme se ji vložit do databáze po kliknutí na tlačítko.

Příklad – Vytvoření formuláře vložení záznamu

Uvedeným postupem si zopakujeme všechny pojmy této kapitoly. Nejprve tedy vytvořme formulář pro novou knihu (viz. příklad). Většinu položek budeme zadávat z textboxů, pouze pro typ knihy připravíme *Drop Down List*. Budeme pracovat s SQL datovým připojením a množinou *DataSet*, proto musíme importovat potřebné jmenné prostory.

Příklad – Importování jmených prostorů

```
<%@ Page Language="VB" %>
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="system.data.sqlclient" %>
```

Nejprve nadefinujeme proměnnou datového spojení. Musíme si uvědomit, že nemůžeme do tabulky vložit knihu s hodnotou KnihaID, která již v tabulce existuje. Proto musíme definovat výběrový dotaz, kterým uvedenou skutečnost zkontrolujeme. Definujeme také datový adaptér. To vše je součástí procedury stisku tlačítka, které má vložit nový záznam do databáze.

Příklad – Založení datového spojení

```
Dim MyConnection As New SqlConnection("Data Source=.\SQLEXPRESS;
AttachDbFilename=D:\UT_Příklady\Kapitola8\App_Data\Database.mdf;
Integrated Security=True;Connect Timeout=30;User Instance=True")

Dim searchIsbnString = "Select * from Knihy where KnihaID = '" &
tbID.Text & "'"
Dim MyCommand As New SqlDataAdapter(searchIsbnString, MyConnection)
```

Vytvoříme datový pohled a pokud v něm najdeme knihu, kterou chce uživatel vložit, tak to uživateli oznámíme.

Příklad – Kontrola existujícího záznamu

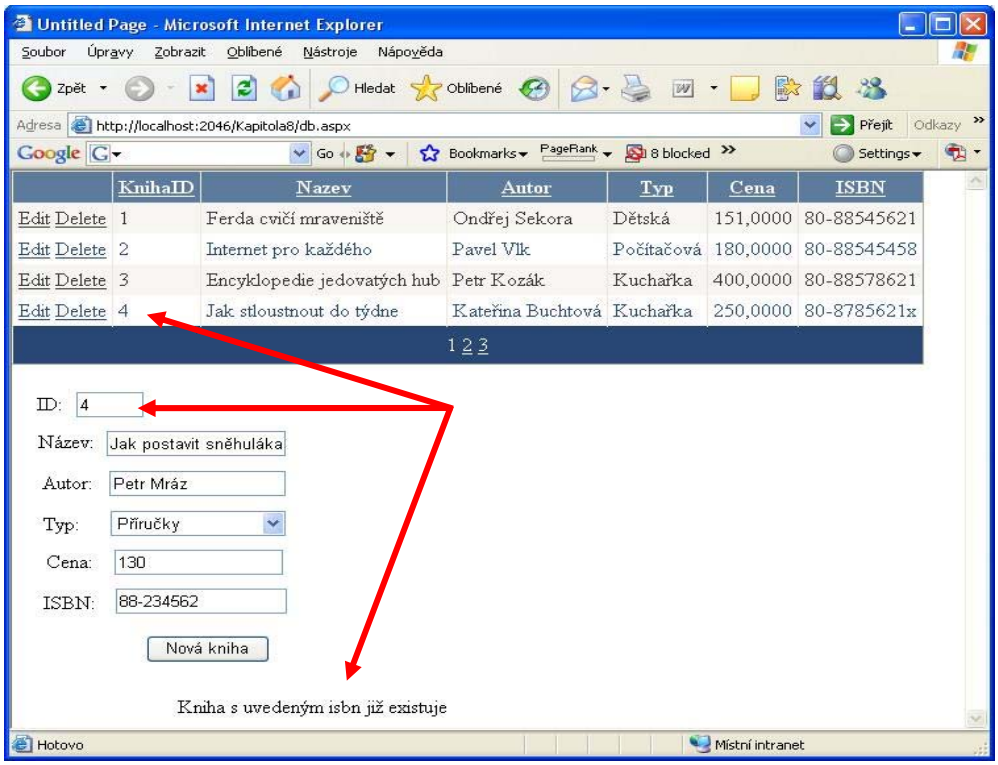
```
Sub klik(ByVal obj As Object, ByVal e As EventArgs)
Dim MyConnection As New SqlConnection("Data Source=.\SQLEXPRESS;
AttachDbFilename=D:\UT_Příklady\Kapitola8\App_Data\Database.mdf;
Integrated Security=True;Connect Timeout=30;User Instance=True")
Dim searchIsbnString = "Select * from Knihy where KnihaID = '" &
tbID.Text & "'"
Dim MyCommand As New SqlDataAdapter(searchIsbnString, MyConnection)
        'naplneni datove mnoziny
        Dim dSet As DataSet = New DataSet()
        MyCommand.Fill(dSet, "Knihy")
        Dim dTable As DataTable
        Dim tableView As DataView
        dTable = dSet.Tables("Knihy")
        tableView = New DataView(dTable)
        If tableView.Count = 1 Then
            lblZprava.text = "Kniha s uvedeným isbn již existuje"
            Exit Sub
        End If
```

Procedura kliknutí tlačítka bude pokračovat nadefinováním vkládacího dotazu. Posbíráme hodnoty z *textboxů* a *DropDownListu* a vložíme je do tabulky. Nejprve spojení otevřeme, poté provedeme SQL dotaz a spojení uzavřeme. Uvedený postup máme navržen se zachycením vyjímky, tak aby v případě problému neviděl uživatel ošklivé chybové hlášení, které není vhodné ani bezpečné, ale aby se zobrazilo hlášení v *labelu*.

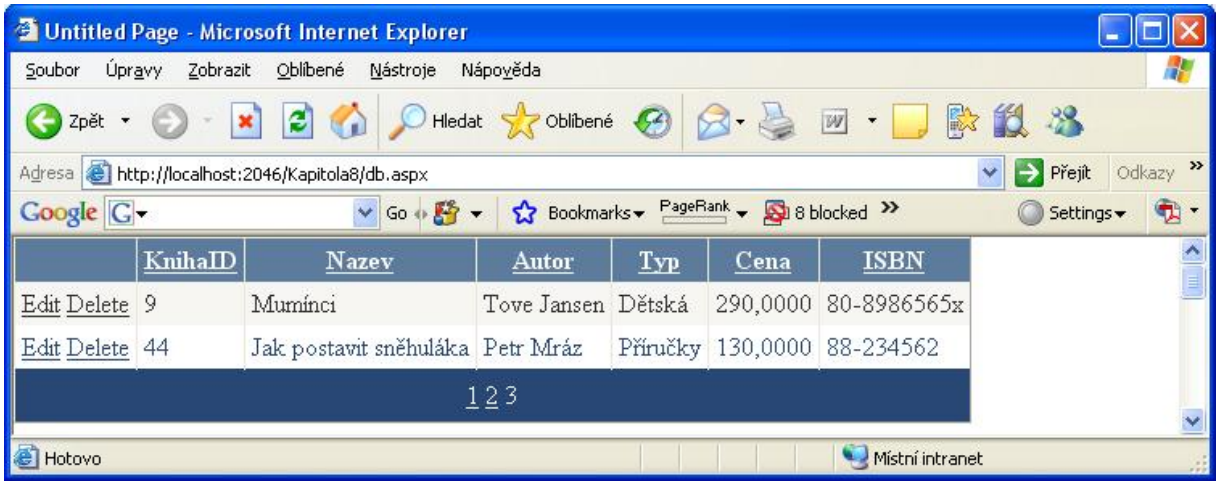
Příklad – Vložení nového záznamu do databázové tabulky

```
Dim sqlString = "Insert Into knihy (KnihaID, Nazev, Autor, typ,
cena, ISBN) Values ('" & tbID.Text & "', '" & tbNazev.Text & "', '"
& tbAutor.Text & "', '" & ddlTyp.SelectedItem.Text & "', '" &
TbCena.Text & "', '" & tbIsbn.Text & "')"
Dim objCmd As New SqlCommand(sqlString, MyConnection)
        Try
            objCmd.Connection.Open()
            objCmd.ExecuteNonQuery()
        Catch ex As Exception
            lblZprava.text = "databázová operace se nezdařila."
        End Try
        objCmd.Connection.Close()
    End Sub
```

Na následujících obrázcích vidíme chod aplikace. V případě zadání již existujícího ID se kniha nevloží a objeví se hlášení, pokud vše proběhne v pořádku, kniha bude vložena na tabulky.



Obr. 8.14 Musí být zajištěna kontrola jedinečnosti ID

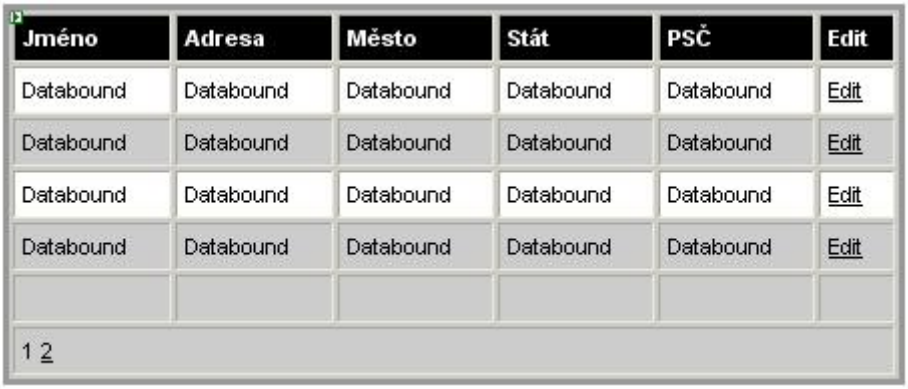


Obr. 8.15 Vložení nového záznamu proběhlo úspěšně

V předchozích odstavcích jsme se zmínili o *Data Gridu*, jenž byl předchůdcem ovládacího prvku *Grid View*. Tento prvek se stále poměrně často vyskytuje v řešených příkladech různých knih a internetových zdrojů a proto si jej představme v ukázce načtení dat z databáze MS Access. Tato databáze byla vybrána záměrně, abychom demonstrovali obdobný postup pouze s odlišnými objekty. Nejprve musíme naimportovat potřebný jmenný prostor direktivou:

```
<%@ Import Namespace="System.Data.OleDb" %>
```

Příklad – Formulář datagridu



Jméno	Adresa	Město	Stát	PSČ	Edit
Databound	Databound	Databound	Databound	Databound	Edit
Databound	Databound	Databound	Databound	Databound	Edit
Databound	Databound	Databound	Databound	Databound	Edit
Databound	Databound	Databound	Databound	Databound	Edit

1 2

V ukázce příkladu vidíme obdobný postup, místo *SqlConnection* používáme *OleDbConnection* s příslušným providerem dat.

Příklad – Vytvoření datového spojení

```
sub Page_Load(obj as object, e as eventArgs)

'vytvoreni databazoveho pripojeni
Dim myconnection As New
OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0; " &
"Data Source=D:\UT_Příklady\Kapitola9\App_Data\database.mdb")

'otevreni pripojeni
Dim MyCommand As New OleDbDataAdapter("select * from tblZam",
myconnection)
```

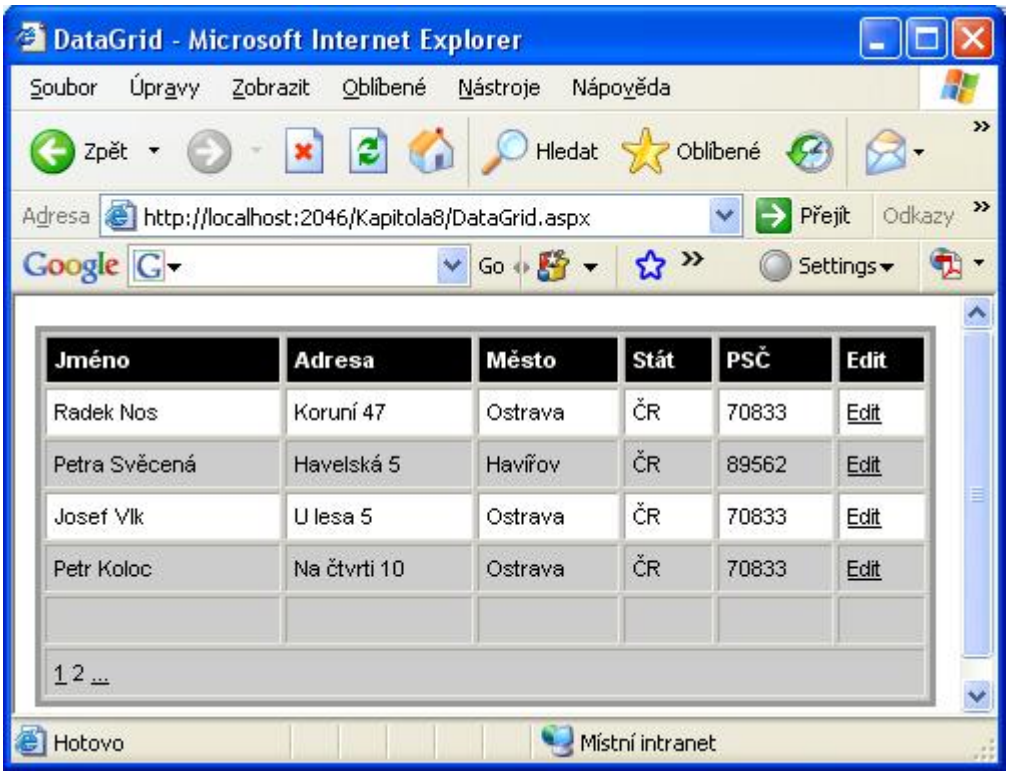
DataGrid je tedy obdobou *Grid View*, jak jde vidět ze zdrojového kódu. Bohužel mnoho funkcionalit musíme obhospodařit ručně, což znamená psaní množství kódu. Na jedné straně nás to donutí porozumět všem možným úskalím, které u *Grid View* neřešíme, jako např. procedura změny indexu stránky tabulky po vyvolání události `OnPageIndexchanged`, na druhou stranu je to však vykoupeno časovou náročností při psaní kódu.

Příklad – Datagrid

```
<asp:DataGrid id="DataGrid1" runat="server" width="450px" Font-
Names="Arial" Font-Size="8pt" ShowFooter="True" HeaderStyle-
BackColor="#cccc99" FooterStyle-BackColor="#cccc99" ItemStyle-
BackColor="#ffffff" AlternatingItemStyle-BackColor="#cccccc"
AutoGenerateColumns="False" OnEditCommand="DataGrid1_Edit"
OnCancelCommand="DataGrid1_Cancel"
OnUpdateCommand="DataGrid1_Update" AllowPaging="True" PageSize="4"
PagerStyle-Mode="NextPrev" PagerStyle-PagebuttonCount="2"
OnPageIndexchanged="Datagrid1_PageIndexchanged"
BorderColor="#999999" BorderStyle="Solid" BorderWidth="3px"
BackColor="#CCCCCC" CellPadding="4" CellSpacing="2"
ForeColor="Black">

<FooterStyle BackColor="#CCCCCC"></FooterStyle>
<HeaderStyle Font-Bold="True" ForeColor="White"
BackColor="Black"></HeaderStyle>
<PagerStyle HorizontalAlign="Left" ForeColor="Black"
```

```
BackColor="#CCCCCC" PageButtonCount="2"
Mode="NumericPages"></PagerStyle>
<SelectedItemStyle Font-Bold="True" ForeColor="White"
BackColor="#000099"></SelectedItemStyle>
<ItemStyle BackColor="White"></ItemStyle>

<Columns>
      <asp:TemplateColumn HeaderText="Jméno">
      <ItemTemplate>
      <asp:Label id="Name" runat="server" Text='<%#
Container.DataItem("FirstName") & " " &
Container.DataItem("LastName") %> '/>
      </ItemTemplate>
      </asp:TemplateColumn>
<asp:BoundColumn DataField="Address" SortExpression="Address"
HeaderText="Adresa"></asp:BoundColumn>
<asp:BoundColumn DataField="City"
HeaderText="Město"></asp:BoundColumn>
<asp:BoundColumn DataField="State"
HeaderText="Stát"></asp:BoundColumn>
<asp:BoundColumn DataField="Zip" HeaderText="PSČ"></asp:BoundColumn>
<asp:EditCommandColumn ButtonType="LinkButton"
UpdateText="Aktualizace" HeaderText="Edit" CancelText="Storno"
EditText="Edit">
<ItemStyle Wrap="False"></ItemStyle>
</asp:EditCommandColumn>
</Columns>
</asp:DataGrid></form>
```

Na obrázku 8.16 vidíme výslednou aplikaci zobrazení databázové tabulky v *Data Gridu* i s editací a stránkováním, tak jako u *Grid View*. Nemusíme ale pokládat triviální otázku, který prvek zabral daleko více času a námahy.



Obr. 8.16 Aplikace Data Gridu

Ke kapitole 8 je přiřazena demonstrační animace č.7

Animace č. 7 obsahuje vytvoření datového spojení, založení databázové tabulky v projektu, její zobrazení v prohlížeči a nastavení možnosti editace, mazání, vkládání nového záznamu, stránkování a řazení.

Shrnutí kapitoly

Přístup k datům v aplikacích .NET je založen na vlastní technologii *.NET Frameworku* nazývané ADO.NET. Tato technologie zajišťuje komfortní připojení a správu dat ve všech typech aplikací, ať už desktopových či webových.

Poskytovatel dat neboli provider je propojovacím článkem mezi databázovým zdrojem dat a aplikací, která data zobrazuje. Úkolem providera je nejen navázat spojení s konkrétním zdrojem dat, ale také vykonávat na něm SQL příkazy a získávat data. *.NET Framework* disponuje čtyřmi základními poskytovateli:

• SQL Server Provider - Poskytuje přístup k databázím SQL Serveru 7.0 a novějším.

• OLE DB Provider - Poskytuje přístup zdroji dat, který obsahuje nějaký ovladač OLE DB, tedy i starší verze SQL Serveru.

• Oracle Provider - Poskytuje přísup k databázím Oracle (verze 8 a novější).

• ODBC Provider - Poskytuje přístup zdroji dat s ovladačem ODBC.

Pro každého providera máme k dispozici tyto třídy:

• Connection (připojení) – Nejprve se musíme ke zdroji dat připojit.

• Command (příkaz) – Na zdroji dat vykonáváme SQL příkaz.

• Data Reader (čtenář dat) – Získaná data SQL příkazem přečteme.

• Data Adapter (datový adaptér) – Naplňuje datové sady a promítá změny do zdroje dat.

Postup při vytvoření datového připojení:

1. V našem projektu vytvoříme SQL databázový mdf soubor.

2. V Server Exploreru klikneme na datová spojení a zvolíme *Add Connection*.

3. Zvolíme datový zdroj z našeho projektu.

4. Otestujeme připojení.

Databázovou tabulku vytváříme přímo ve Visual Studiu. V průzkumníku serveru rozbalíme konkrétní datové propojení a můžeme vytvářet databázové tabulky. Nejprve jejich strukturu, poté je naplníme daty. Tabulku zobrazíme v prohlížeči jednoduchým přetáhnutím tabulky z průzkumníka serveru na stránku v design módu. Tohle velmi rychlé přetažení myší má ale za následek několik akcí s téměř desítkami řádků zdrojového kódu, který bychom jinak museli programovat. Automaticky se vygeneruje ovládací prvek *Grid View* pro zobrazení databázové tabulky, jehož základní neupravená podoba obsahuje sloupce tabulky. Společně s touto podobou vidíme ještě tzv. tasks ovládacího prvku, které známe i z předchozích kapitol. U *Grid View* však mají mocné funkce a průvodce, které se netýkají pouze vzhledu prvku, máme možnost data v tabulce řadit, stránkovat, vybírat, mazat a editovat. Avšak vložení nového záznamu do tabulky ovládací prvek *Grid View* dosud bohužel neumí a uvedenou situaci budeme muset vyřešit sami.

Úkol k řešení 8.1 – Vytvoření databázové tabulky

Vytvořte datové připojení a vlastní tabulku např. hudebních CD disků, zobrazte tabulku v prohlížeči, umožněte editaci, mazání, stránkování a řazení jednotlivých záznamů.

Úkol k řešení 8.2 – Práce s databázovou tabulkou

Zajistěte vložení nového záznamu do tabulky z předchozího příkladu.

Kontrolní otázka 8.1

Jakým způsobem zajistíme stránkování tabulky a jak nastavíme počet zobrazených záznamů na stránce?

Kontrolní otázka 8.2

Jaké typy datových zdrojů mohu použít?

Kontrolní otázka 8.3

Co je to datový pohled?

Kontrolní otázka 8.4

Co je to připojovací řetězec a kde ho najdeme?

Kontrolní otázka 8.5

Jaký jmenný prostor musíme naimportovat, pokud budeme chtít využít třídu *DataSet*?

Kontrolní otázka 8.6

Charakterizujte hlavní rozdíly mezi *Data Gridem* a *Grid View*.

9. TECHNIKA VÁZÁNÍ DAT

Čas ke studiu: 3 hodiny

Cíl Po prostudování této kapitoly budete umět

- vysvětlit princip vázání dat
- charakterizovat ovládací prvky pro data
- používat ovládací prvky pro vázání dat
- programovat širokou škálu příkladů, které váží data z databázových tabulek
- implementovat hierarchické vázání dat z XML souboru pomocí Tree View

Výklad

Webové aplikace vyžadující nějaká data z různých datových zdrojů jsou stále žádanější. Pokud chceme zpracovávat data z databázových tabulek, musíme zajistit jejich vázání na určité serverové ovládací prvky. Technologie ASP.NET je na tento požadavek dobře připravena a nabízí bohatý model pro vázání dat, tzv. Data-binding. V této kapitole se seznámíme s ovládacími prvky, které data-binding využívají a na názorných příkladech si tyto prvky procvičíme.

9.1 Vázání dat

Vázání dat je mechanismus implementovaný v ASP.NET, který dokáže automaticky zobrazovat data. Máme několik možností jak vázat data z datových zdrojů. Jednoduché vázání dat můžeme zajistit i s klasickými ovládacími prvky jako je *textbox*, *label*, *linkButton* a další. Složitější vázání dat podporují jednak prvky pro seznamy, které již dokáží svázat ucelenou sadu dat, ale hlavní hybnou silou vázání dat jsou vestavěné ovládací prvky pro data, jako např. *Grid View*. Jednoduché vázání dat je realizováno oddělovači `<%#` a `%>`. Pozor nejedná se o blok skriptu. Můžeme tak v HTML kódu svázat s textem proměnnou typu *String*:

```
Příklad vázání dat: <%# retezec%>
```

Jednoduché vázání dat můžeme provést i se zmíněným ovládacím prvkem textbox, který svážeme opět s proměnnou typu *String*, složené vázání dat si ukážeme na *ListBoxu*:

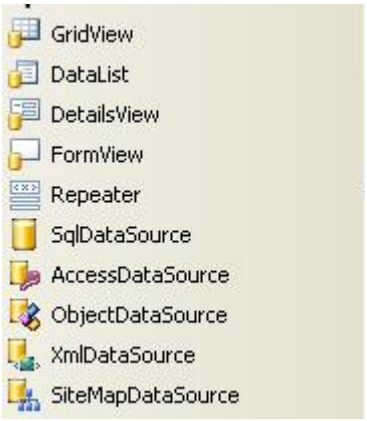
```
Dim retezec2 As String = " vázaný řetězec do textboxu"
Dim PolePolozek() As String = {"Maso", "Mléko", "Pečivo", "Chléb", "Voda"}

<asp:textbox ID="tb1" runat="server" Text='<%# retezec2 %>'/>
<asp:listbox ID="lb1" runat="server" DataSource='<%# PolePolozek %>' />
```

Aby tyto vázané bloky byly funkční, musíme zavolat metodu *DataBind()*. Bez této metody by ovládací prvky neměly co navázat

```
Sub Page_load(ByVal obj As Object, ByVal e As EventArgs)
        Page.DataBind()
End Sub
```

9.2 Ovládací prvky pro práci s daty

S představením ovládacích prvků pro práci s daty jsme již začali v předchozí kapitole, kde jsme pracovali s *Grid View* a jeho předchůdcem *Data Grid*. Ovládací prvky pro data již byly implementovány v předchozích verzích ASP.NET 1.x a mimo představeného *Data Gridu* to jsou ještě *Repeater* a *DataList*. *Repeater* je nejjednodušším prvkem, který nemá žádné předdefinované zobrazení a pomocí šablon si ho přizpůsobujeme své potřebě. S *Data Listem* se pracuje obdobně jako s *Grid View*, má jednodušší model, nenabízí tolik možností, ale oproti *Grid View* umožňuje vícesloupcovou tabulku záznamů. *Grid View* vyžaduje pro jeden záznam samostatný řádek. V ASP.NET 2.0 byly ještě doimplementovány prvky *Details View* a *Form View*. Ty vycházejí z *Grid View* a jejich použití tkví v možnosti prohlížet podrobněji jeden konkrétní záznam. V této kapitole si ještě ukážeme možnosti hierarchického vázání dat, které nám nabízí ovládací prvek *Tree View*.



Pojem k zapamatování: Zdroje dat

S pokročilými ovládacími prvky pro práci s daty úzce souvisí ovládací prvky pro zdroje dat. Použijeme-li ovládací prvek *SqlDataSource* či *AccessDataSource*, nebudeme muset psát kód pro přístup k datům. Pokud je zdrojem dat XML soubor, použijeme *XmlDataSource*. *ObjectDataSource* slouží pro přístup k datům vlastní třídy a *SiteMapDataSource* umožňuje přísup k navigačnímu zdroji dat umístěnému v souboru *Web.SiteMap*.

Obr. 9.1 Ovládací prvky pro práci s daty

Jistě jste si všimli, že přetažením tabulky do stránky v návrhovém módu se nevygeneroval automaticky pouze ovládací prvek *Grid View*, ale také zdroj dat *SqlDataSource*. Ten umožňuje připojení k jakémukoliv zdroji dat, který má svého poskytovatele. Jedná se tedy i o Oracle, OleDB nebo ODBC.

9.3 Ovládací prvek Grid View

Pomocí tohoto prvku jsme dokázali zobrazit databázovou tabulku s možností stránkování, řazení, editace a mazání bez toho, aniž bychom museli napsat jediný řádek zdrojového kódu. Ovládací prvek *Grid View* obsahuje mimo své formátovací vlastnosti také důležitou část, jíž jsou sloupce datové tabulky, které zobrazují data z datového zdroje.

Pojem k zapamatování: AutoGenerateColumns

Přetažením tabulky z průzkumníka serveru do návrhu stránky zobrazíme databázovou tabulku se všemi sloupci tak, jak je nadefinována. To zajistí vlastnost *AutoGenerateColumns* nastavena na *True*. Pokud chceme zobrazit pouze některé sloupce, nebo je zobrazit v jiném pořadí, nastavíme tuto vlastnost na *False*. Poté nadefinujeme v sekci <columns> vlastní pořadí vázaných sloupců.

Dosud jsme se setkali s nejpoužívanějším sloupcem a to je BoundField:

```
<asp:BoundField DataField="Cena" HeaderText="Cena" />
```

Kromě tohoto sloupce však máme i další možnosti tvorby sloupců. Jejich výčet společně s popisem uvádí následující tabulka.

Sloupec	Charakteristika
BoundField	Zobrazí sloupec ze zdroje dat.
ButtonField	Zobrazí tlačítko pro každý řádek.
CheckBoxField	Zobrazí zaškrtávací pole pro každý řádek.
CommandField	Tlačítko pro výběr či editaci.
HyperLinkField	Zobrazení sloupce jako hypertextový obsah.
ImageField	Zobrazení obrázku.
TemplateField	Definuje vlastní šablonu.

Tab. 9.1 Sloupce v Grid View

Ovládací prvek *Grid View* je založený na stylech. Tyto styly určují celkový formát tabulky a to pro každý řádek, i pro každý druhý řádek, hlavičku, zápatí, editaci, stránkování atd. Tyto styly definují na příslušných řádcích barvy, velikosti, zarovnání, rámování a další známé vlastnosti stylů.

Styl	Charakteristika
HeaderStyle	Určuje vzhled řádku hlavičky.
RowStyle	Určuje vzhled každého řádku.
AlternatingRowStyle	Určuje odlišný vzhled lichých řádků.
SelectedRowStyle	Určuje vzhled vybraného řádku.
EditRowStyle	Určuje vzhled řádku v režimu editace.
EmptyDataRowStyle	Určuje vzhled řádku, pokud je tabulka prázdná.
FooterStyle	Určuje vzhled zápatí.
PagerStyle	Určuje vzhled stránkování.

Tab. 9.2 Styly Grid View

Pokud chceme přizpůsobit obsah zobrazovaných buněk svým potřebám, přidávat do nich různý HTML kód či ovládací prvky, nevystačíme s vázanými sloupci a musíme definovat svou vlastní šablonu ve sloupci *TemplateField*.

Příklad – Sloupec se šablonou v Grid View

```
<Columns>
 <asp:TemplateField  HeaderText="Kniha">
    <ItemTemplate>
         <%# Eval("Nazev") %>, autor: <%#Eval("Autor")%> <br />
         Kniha stojí <%# Eval("Cena") %> Kč
    </ItemTemplate>
 </asp:TemplateField>

/Columns>
```

Pro vázání dat pak potřebujeme metodu *Eval()*. Tato metoda má výhodu v možnosti použití formátovacích řetězců přímo v kódu:

```
Kniha stojí <%# Eval("Cena","{0:C}") %>
```

Korespondenční úkol – Formátovací řetězce

Vyhledejte tvary formátovacích řetězců pro čísla, datum a čas.

Šablona *ItemTemplate* není jedinou šablonou, která se dá využít pro tvorbu vlastního vzhledu *GridView*. Pokud v *GridView Tasks* klikneme na položku *Edit Templates*, máme další volby pro editaci zbylých šablon. Tuto editaci však můžeme provést přímo ve zdrojovém kódu za použití IntelliSense. Šablony mají obdobnou funkci jako vázané sloupce a jejich přehled uvádí tabulka.

Šablona	Charakteristika
HeaderTemplate	Určuje vzhled hlavičky
FooterTemplate	Určuje vzhlad zápatí
ItemTemplate	Určuje vzhled jednotlivých buněk tabulky
AlternatingItemTemplate	Určuje odlišný vzhled lichých řádků
EditItemTemplate	Určuje vzhled editace Grid View

Tab. 9.3 Šablony Grid View

9.4 Ovládací prvek Repeater

Repeater je kontejner určený k cyklickému procházení dat. Je definován bez defaultního vzhledu, vše zajišťujeme sami pomocí šablon. Tyto šablony určí výslednou podobu zobrazených dat. Pokud žádné zobrazení šablon neurčíme, ovládací prvek žádná data nezobrazí. Protože jsme se v minulém odstavci se šablonami setkali, již víme, co bychom mohli od *Repeatru* očekávat. Šablon má *Repeater* pět:

- *ItemTemplate* – Povinná šablona *Repeatru* vytvářející pro datový záznam jeden řádek výstupu. Vlastní data jsou vázána podle popsaného mechanismu výše.
- *AlternatingItemTemplate* – Šablona pro odlišení lichých a sudých řádků.
- *SeparatorTemplate* – Zobrazuje oddělovač mezi jednotlivými řádky dat.
- *HeaderTemplate* a *FooterTemplate* – Zajišťuje zobrazení hlavičky a zápatí tabulky.

Repeater si předvedeme ihned v ukázce. Nejprve si vytvoříme databázové připojení, následně ho otevřeme, naplníme datovou množinu a svážeme datový pohled s ovládacím prvkem *Repeater*.

Příklad – Skript k příkladu demonstrující činnost repeatru

```
sub Page_Load(obj as object, e as eventArgs)

'vytvoreni databazoveho pripojeni
Dim myconnection As New
OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;" &
"Data Source=D:\UT_Příklady\Kapitola9\App_Data\database.mdb")

'otevreni pripojeni
Dim MyCommand As New OleDbDataAdapter("select * from tblZam",
myconnection)

'naplneni datove mnoziny
dim ds as DataSet = new DataSet()
 MyCommand.Fill(ds, "tblZam")

'vyber datoveho pohledu a svazani se serverovym ovladacim prvkem
Rep.DataSource = ds.Tables("tblZam").DefaultView
DataBind()
```

Repeater bude cyklicky zobrazovat do dvou sloupců tabulky jméno a příjmení a telefon z databázové tabulky. Šablony *Header* a *Footer Template* budou obsahovat pouze začátek a konec tabulky, položka *Item* a *AlternatingItemTemplate* bude obsahovat stejná vázaná data, rozdíl bude pouze v barvě pozadí buňky. Jako oddělovač zvolíme pouze ilustrativně tři znaménka mínus.

Příklad – Zdrojový kód ovládacího prvku Repeater

```
<asp:Repeater ID="Rep" runat="server">
      <HeaderTemplate>
      <table><tr>
      <td style="background-color:#cccc99; width:200"><b>Jméno</b>
</td>
      <td style="background-color:#cccc99; width:200"><b>Telefon</b>
</td> </tr>
      </HeaderTemplate>
      <ItemTemplate>
      <tr><td><%# Container.DataItem("FirstName")%>
              <%# Container.DataItem("LastName")%>    </td>
          <td><%# Container.DataItem("Phone")%></td></tr>
      </ItemTemplate>
      <AlternatingItemTemplate>
      <tr><td style="background-color:#cccc99">
          <%# Container.DataItem("FirstName")%>
          <%# Container.DataItem("LastName")%>  </td>
          <td style="background-color:#cccc99">
          <%# Container.DataItem("Phone")%></td></tr>
      </AlternatingItemTemplate>
      <SeparatorTemplate>
      <tr><td colspan="2" align="center">  - - -  </td></tr>
      </SeparatorTemplate>
      <FooterTemplate>
      </table>
      </FooterTemplate>
</asp:Repeater>
```

Pokud napíšeme tento zdrojový kód a přepneme se do návrhového módu, uvidíme výsledný ovládací prvek, tak jak jsme jej nadefinovali podle šablon:

Příklad – Ovládací prvek Repeater v návrhovém zobrazení

Jméno	Telefon
Databound Databound	Databound
- - -	
Databound Databound	Databound
- - -	
Databound Databound	Databound
- - -	
Databound Databound	Databound
- - -	
Databound Databound	Databound

Po spuštění aplikace můžeme v prohlížeči vidět výsledný *Repeater* (viz. obr. 9.2).

Pojem k zapamatování: Repeater

Repeater je cyklický kontejner dat založený na šablonách. Nečekejme od něj nějaké pokročilé vestavěné funkce, avšak tento prvek má své místo mezi prvky pro zobrazování dat a použití si určitě najde.

První příklad procvičující *Repeater* byl názornou ukázkou vázání dat. Ve skriptu aplikace jsme museli obsloužit celou řadu akcí. Druhý příklad již věnujeme vázání dat na datový zdroj *SqlDataSource*. To znamená, že nebudeme muset psát kód pro přístup k datům.
Jako zdroj dat zvolíme databázovou tabulku knih v *mdf* souboru, připravenou již v minulé kapitole.
Po spuštění aplikace uvidíme výslednou tabulku v prohlížeči (viz. obr. 9.3). Na první pohled nepoznáme, o jaký ovládací prvek se jedná, neboť ne vždy potřebujeme využít bohatých možností *Grid View*.

Jméno	Telefon
Jaromír Altner	602568778
Karel Větroň	603455557
Petr Plachý	608566666
Simona Okrouhlíková	703569987
Radek Nos	608454545
Petra Svěcená	602587888
Josef Vlk	777589654
Petr Koloc	776568985
Petra Kolocová	771456254

Obr. 9.2 Ovládací prvek Repeater

Příklad – Kostra ovládacího prvku Repeater

```
<ASP:Repeater id="MyRepeater" runat="server"
DataSourceID="SqlDataSource1">
<HeaderTemplate>
   <table width="100%" style="font: 8pt verdana">
   <tr style="background-color:Gray">
    <th> Název </th>
    <th> Autor </th>
    <th> Cena  </th>
    <th> Typ   </th>
    <th> ISBN  </th>
   </tr>
</HeaderTemplate>
<ItemTemplate>
 <tr style="background-color:Silver">
 <td> <%#DataBinder.Eval(Container.DataItem, "Nazev")%> </td>
 <td> <%#DataBinder.Eval(Container.DataItem, "Autor")%> </td>
 <td> <%#DataBinder.Eval(Container.DataItem, "Cena", "{0} Kč")%>
</td>
 <td>  <%# DataBinder.Eval(Container.DataItem, "Typ") %></td>
 <td>  <%#DataBinder.Eval(Container.DataItem, "ISBN")%> </td></tr>
</ItemTemplate>
<FooterTemplate>
 </table>
</FooterTemplate>
</ASP:Repeater>
```

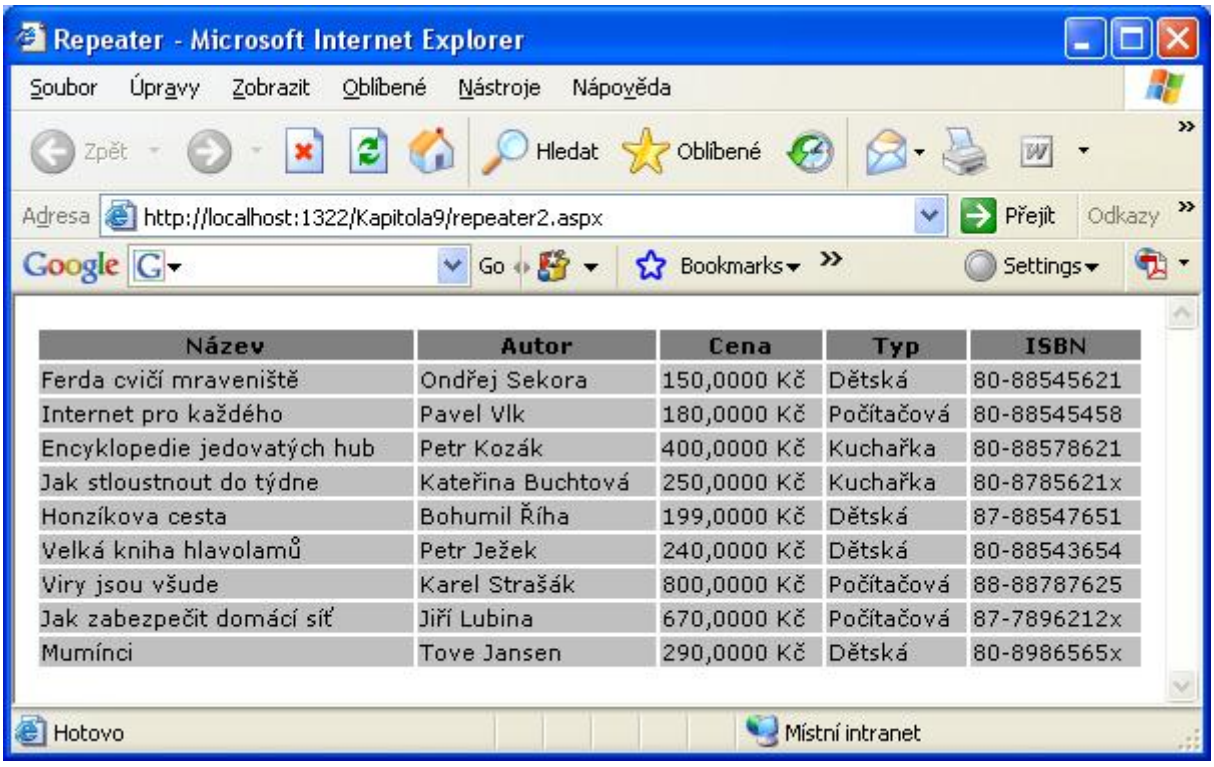
Příklad – SqlDataSource pro Repeater

```
<asp:SqlDataSource ConnectionString="<%$
ConnectionStrings:DatabaseConnectionString1 %>" ID="SqlDataSource1"
runat="server" SelectCommand="SELECT * FROM[Knihy]">
</asp:SqlDataSource>
```

Příklad – Ovládací prvek Repeater v návrhovém zobrazení

Název	Autor	Cena	Typ	ISBN
Databound	Databound	Databound	Databound	Databound
Databound	Databound	Databound	Databound	Databound
Databound	Databound	Databound	Databound	Databound
Databound	Databound	Databound	Databound	Databound
Databound	Databound	Databound	Databound	Databound

SqlDataSource - SqlDataSource1



Název	Autor	Cena	Typ	ISBN
Ferda cvičí mraveniště	Ondřej Sekora	150,0000 Kč	Dětská	80-88545621
Internet pro každého	Pavel Vlk	180,0000 Kč	Počítačová	80-88545458
Encyklopedie jedovatých hub	Petr Kozák	400,0000 Kč	Kuchařka	80-88578621
Jak stloustnout do týdne	Kateřina Buchtová	250,0000 Kč	Kuchařka	80-8785621x
Honzíkova cesta	Bohumil Říha	199,0000 Kč	Dětská	87-88547651
Velká kniha hlavolamů	Petr Ježek	240,0000 Kč	Dětská	80-88543654
Viry jsou všude	Karel Strašák	800,0000 Kč	Počítačová	88-88787625
Jak zabezpečit domácí síť	Jiří Lubina	670,0000 Kč	Počítačová	87-7896212x
Mumínci	Tove Jansen	290,0000 Kč	Dětská	80-8986565x

Obr. 9.3 Druhá ukázka ovládacího prvku Repeater

9.5 Ovládací prvek Data List

Ovládací prvek *Data List* je mezistupněm mezi *Repeatrem* a *Grid View*. Do jisté míry se podobá *Repeatru*, je také založen na šablonách, obsahuje navíc možnosti komunikace s uživatelem při modifikaci záznamů. Základní šablony jsou shodné s *Repeatrem*, obsahuje však navíc dvě šablony:

- *SelectedItemTemplate* – Obsahuje další elementy pro výběr položky.
- *EditItemTemplate* – Definuje rozložení *Data Listu* v editačním módu.

V našem kurzu se snažíme o to, abychom si na různých příkladech procvičili co nejvíce možností vázání dat. Zobrazili jsme již data z různých databází, rovněž různými způsoby, ať už naplněním datové množiny a svázání s datovým pohledem a nebo přes *SqlDataSource* bez nutnosti psaní zdrojového kódu. Na prvním příkladu s *Data Listem* si ukážeme vázání dat z vlastního datového zdroje. Nejprve tedy vytvořme ovládací prvek *Data List*:

Příklad – Zdrojový kód Data Listu

```
<asp:DataList id="DL1" runat="server"
              BorderColor="#3366CC"
              BorderWidth="1px"
              GridLines="Both"
              CellPadding="4"
              Font-Names="Verdana"
              Font-Size="8pt"
              Width="200px"
              HeaderStyle-BackColor="#aaaadd"
              AlternatingItemStyle-BackColor="Gainsboro"
              SelectedItemStyle-BackColor="yellow"
              OnItemCommand="DataList_ItemCommand" BackColor="White"
               BorderStyle="None">
<HeaderTemplate> Sklad součástek </HeaderTemplate>
<ItemTemplate>
<asp:LinkButton id="button1" runat="server" Text="Detail součástky"
    CommandName="select" />
<%# DataBinder.Eval(Container.DataItem, "StringValue") %>
</ItemTemplate>
<SelectedItemTemplate>
Položka:
<%# Container.DataItem("StringValue")  %><br>
Datum uložení:
<%# DataBinder.Eval(Container.DataItem, "DateTimeValue", "{0:d}")%>
<br>Počet:
<%# DataBinder.Eval(Container.DataItem, "IntegerValue", "{0:N1}") %>
</SelectedItemTemplate>
<FooterStyle BackColor="#99CCCC" ForeColor="#003399" />
<SelectedItemStyle BackColor="#009999" Font-Bold="True"
    ForeColor="#CCFF99" />
<ItemStyle BackColor="White" ForeColor="#003399" />
<HeaderStyle BackColor="#003399" Font-Bold="True"
    ForeColor="#CCCCFF" />
<AlternatingItemStyle BackColor="Gainsboro" />
</asp:DataList>
```

Ve zdrojovém kódu vidíme nejdůležitější šablonu *ItemTemplate* pro zobrazení položek a také výběrovou šablonu *SelectedItemTemplate*, která zajistí zobrazení detailu položky.

Nyní vytvořme jednoduchý vlastní zdroj dat. Zdrojem dat bude tabulka, kterou jednoduše naplníme daty v cyklu `For`. Na závěr funkce vytvoříme datový zdroj jako datový pohled do tabulky.

Příklad – Vytvoření datového zdroje

```
Function CreateDataSource() As ICollection

   Dim dt As DataTable
   Dim dr As DataRow
   Dim i As Integer
```

```
    'Vytvoříme tabulku
    dt = New DataTable
    dt.Columns.Add(New DataColumn("IntegerValue", GetType(Integer)))
    dt.Columns.Add(New DataColumn("StringValue", GetType(String)))
    dt.Columns.Add(New DataColumn("DateTimeValue", GetType(DateTime)))

   'Naplňme ji daty
    For i = 1 To 9
       dr = dt.NewRow()
       dr(0) = i
       dr(1) = "Položka " & i.ToString()
       dr(2) = DateTime.Now.ToShortTimeString
       dt.Rows.Add(dr)
    Next

   'Vytvořme datový pohled
   CreateDataSource = New DataView(dt)
   End Function
```

Nakonec musíme samozřejmě zajistit vázání dat a obsloužit proceduru pro výběr položky:

Příklad – Procedury vázání dat a výběr položky

```
Sub Page_Load(sender As Object, e As EventArgs)
        If Not IsPostBack Then
                VazaniDat()
        End If
End Sub

Sub VazaniDat()
    DL1.DataSource = CreateDataSource()
    DL1.DataBind()
End Sub

Sub DataList_ItemCommand(sender As Object, e As
DataListCommandEventArgs)
    Dim cmd As String = e.CommandSource.CommandName
    If cmd = "select" Then
       DL1.SelectedIndex = e.Item.ItemIndex
    End If
    VazaniDat()
End Sub
```

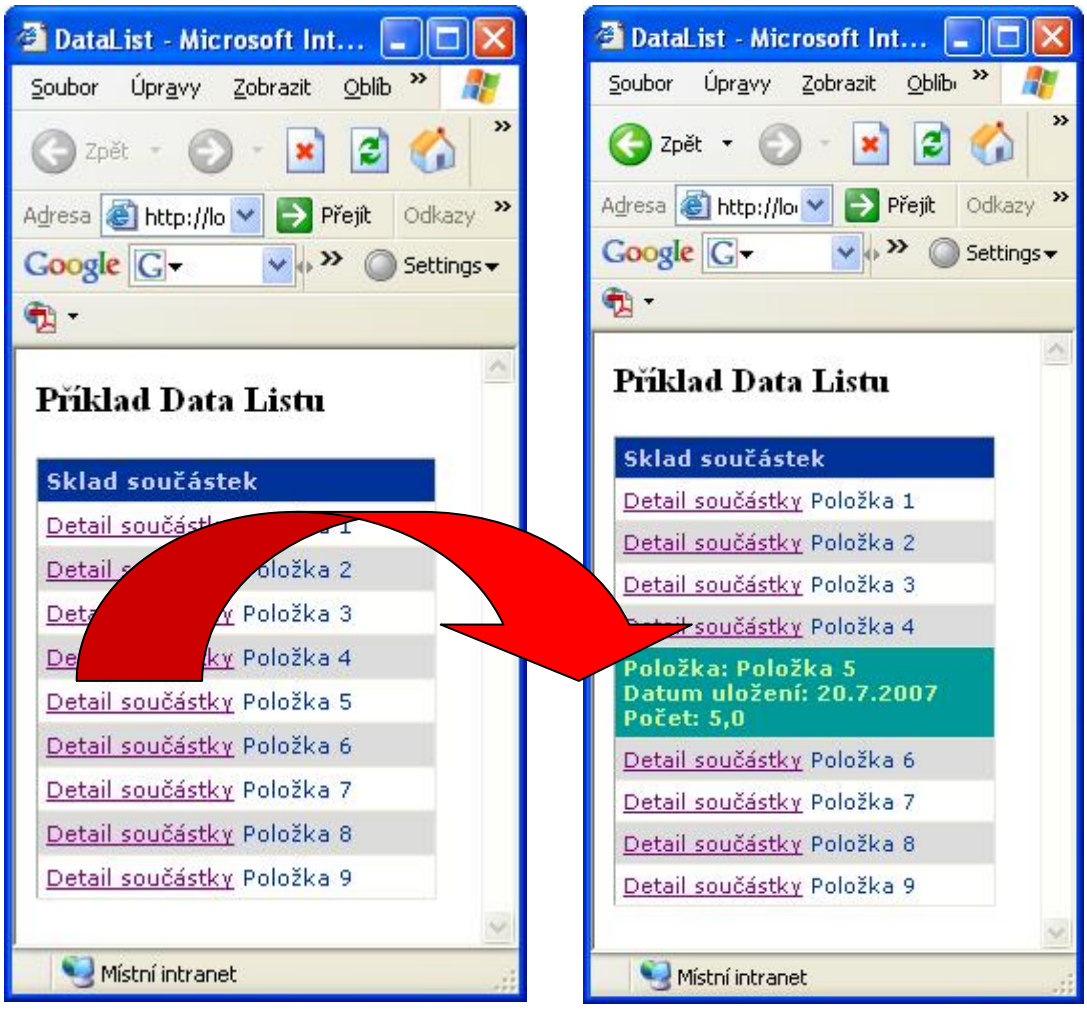
Výslednou aplikaci si můžeme prohlédnout na obrázku 9.4. Výhodou *Data Listu* a v podstatě důvodem, proč je stále využíván, je možnost zobrazovat záznamy ve více sloupcích.

Pojem k zapamatování: Vícesloupcové zobrazení záznamů v Data Listu

Tuto vlastnost nemá ovládací prvek Grid View. Proto budeme-li potřebovat zobrazit více záznamů vedle sebe či pod sebe, použijeme v Data Listu vlastnosti:

```
RepeatColumns="PočetSloupců" RepeatDirection="Horizontal|Vertical"
```

Uvedenou skutečnost si procvičíme na druhém příkladu s *Data Listem*. Naši databázovou tabulku knih si zobrazíme ve třech sloupcích, přičemž jednotlivé položky záznamu budeme chtít zobrazit pod sebe. Jako zdroj dat použijeme *SqlDataSource*, vyhneme se tak psaní zdrojového kódu.

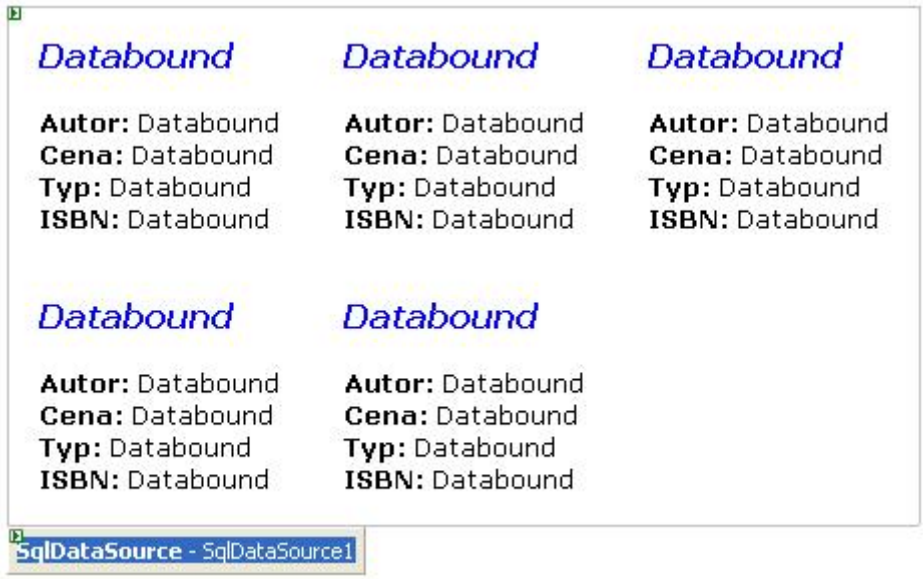


Obr. 9.4 Příklad ovládacího prvku Data List se šablonou SelectedItemTemplate

Příklad – Data List zobrazující záznamy ve více sloupcích

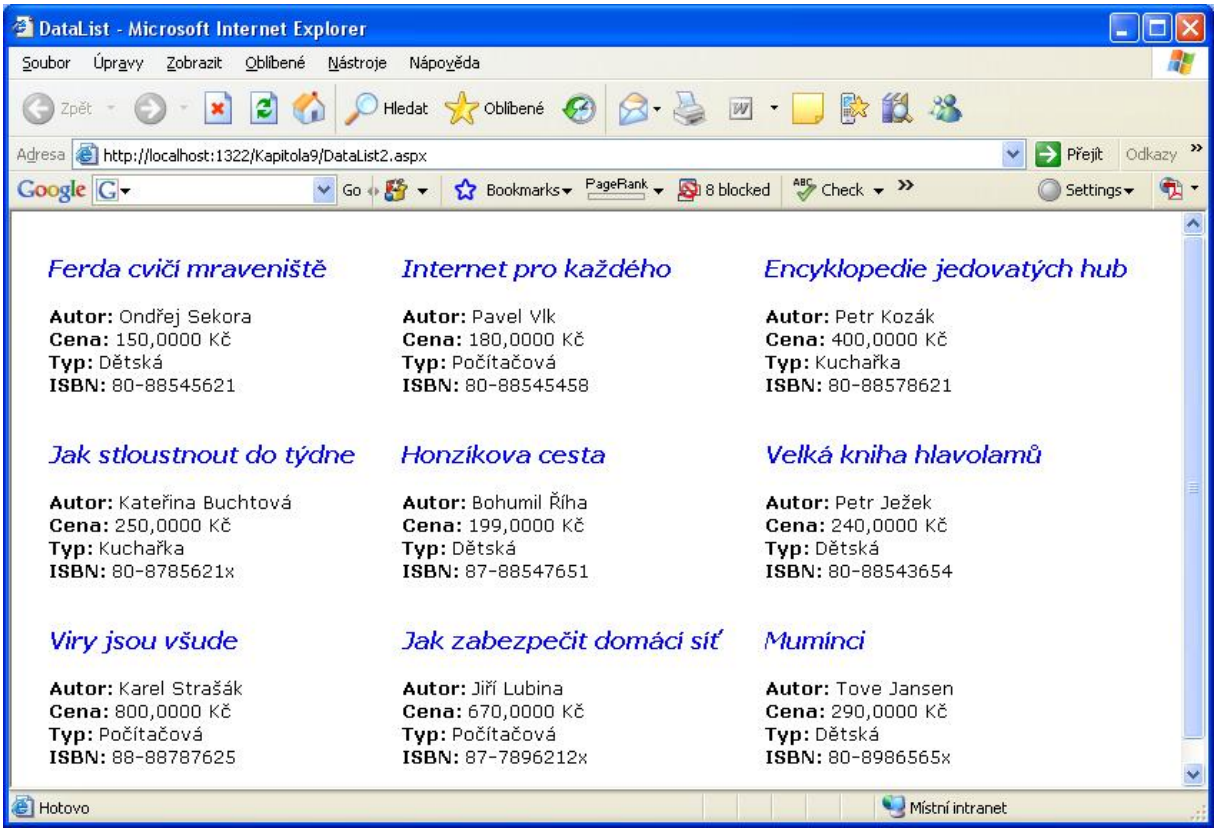
```
<asp:DataList id="MyDataList" DataSourceID="SqlDataSource1"
RepeatColumns="3" RepeatDirection="Horizontal" runat="server">

<ItemTemplate>
<div style="padding:15,15,15,15;font-size:10pt;font-family:Verdana">
<div style="font:12pt verdana;color:Blue">
  <i><b><%# DataBinder.Eval(Container.DataItem, "Nazev") %></i></b>
</div>
<br>
<b>Autor: </b><%#DataBinder.Eval(Container.DataItem, "Autor")%><br/>
<b>Cena: </b><%#DataBinder.Eval(Container.DataItem, "Cena", "{0}
Kč")%><br/>
<b>Typ: </b><%#DataBinder.Eval(Container.DataItem, "Typ")%><br/>
<b>ISBN: </b><%# DataBinder.Eval(Container.DataItem, "ISBN") %><br/>
</div>
</ItemTemplate>
<ItemStyle BorderColor="#E0E0E0" />
</asp:DataList>
```

Příklad – Data List v design módu



Příklad – Datový zdroj pro zobrazení tabulky knih v Data Listu

```
<asp:SqlDataSource ConnectionString="<%$
ConnectionStrings:DatabaseConnectionString1 %>" ID="SqlDataSource1"
    runat="server" SelectCommand="SELECT * FROM [Knihy]"/>
```



Obr. 9.5 Příklad ovládacího prvku Data List s vícesloupcovou strukturou

9.6 Hierarchické vázání – Tree View

Naše příklady na vázání dat ukončíme trochu odlišnými příklady, a to hierarchickým vázáním pomocí ovládacího prvku *Tree View*. Ten dokáže zobrazit úplnou strukturu XML dokumentu. Nejprve si připravme XML soubor s naplněnými daty. Zůstaneme u našeho oblíbeného tématu, jímž jsou knihy. Nadefinujeme si kořenový element Knižní obchod a podelementy s žánry a jednotlivými knihami.

Příklad – Úryvek z XML souboru

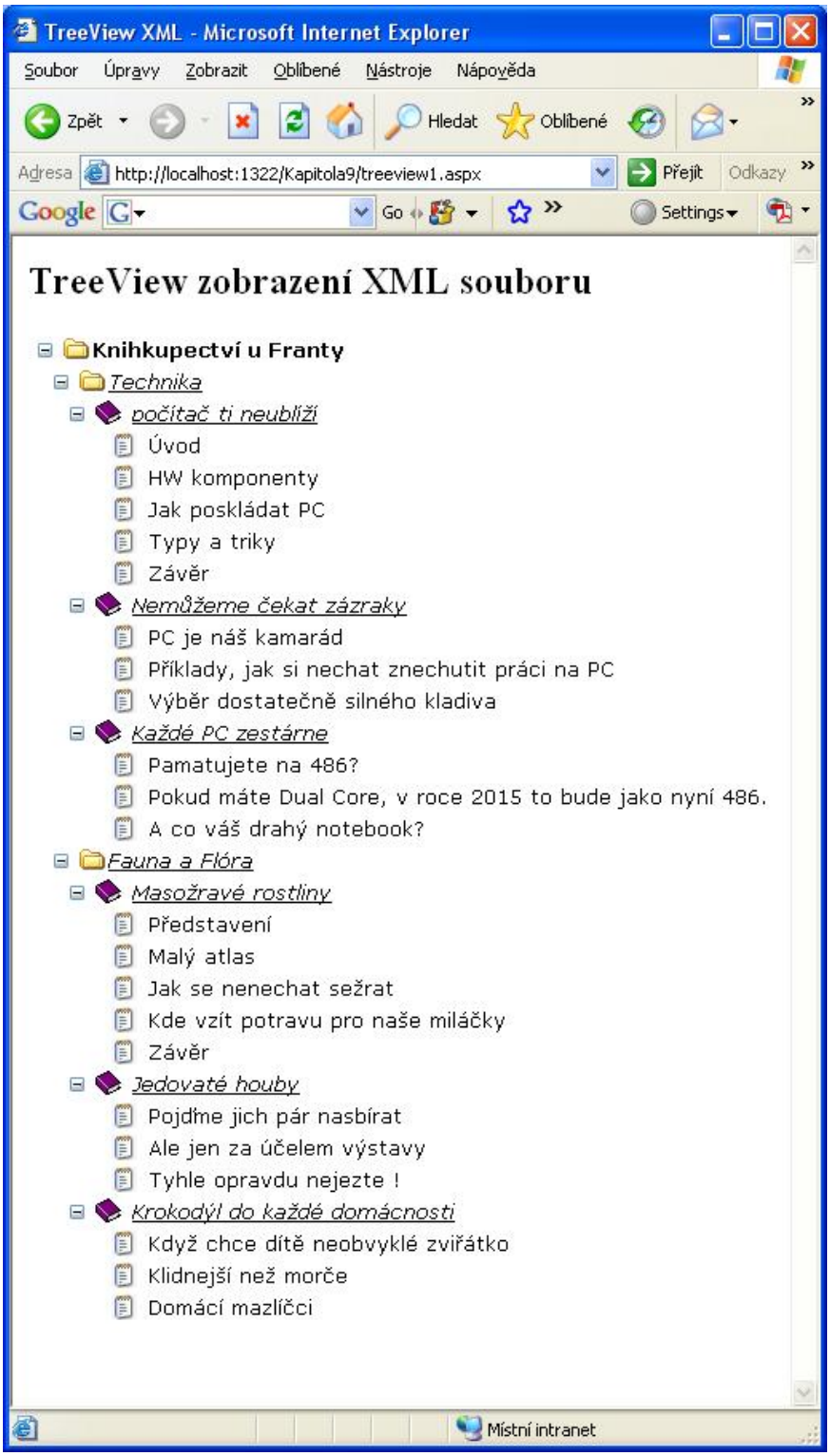
```
<Bookstore>
  <genre name="Technika">
    <book ISBN="881032" Title="počítač ti neublíží" Price="19.99">
      <chapter num="1" name="Úvod">
        Abstract...
      </chapter>
      <chapter num="2" name="HW komponenty">
        Abstract...
      </chapter>
      <chapter num="3" name="Jak poskládat PC">
        Abstract...
      </chapter>
      <chapter num="4" name="Typy a triky">
        Abstract...
      </chapter>
      <chapter num="5" name="Závěr">
        Abstract...
      </chapter>
    </book>
...XML soubor pokračuje
```

Vytvoříme datový zdroj `XmlDataSource` s cestou k našemu XML souboru, který máme uložen v projektu. Dále vytvoříme ovládací prvek `TreeView` s odkazem na ID datového zdroje.

Příklad – XML Data source a TreeView

```
<asp:XmlDataSource ID="ZdrojKnih" DataFile="~/App_Data/knihy.xml"
runat="server"/>

        <asp:TreeView ID="TreeView1"
          SkinId="Bookstore"
          DataSourceId="ZdrojKnih"
          ExpandDepth="3"
          MaxDataBindDepth="3"
          runat="server">
          <Databindings>
 <asp:TreeNodeBinding DataMember="Bookstore" Text="Knihkupectví u
     Franty" ImageUrl="~/images/folder.gif" />
 <asp:TreeNodeBinding DataMember="genre" TextField="name"
     ImageUrl="~/images/folder.gif" />
  <asp:TreeNodeBinding DataMember="book" TextField="Title"
     ImageUrl="~/images/closedbook.gif" />
 <asp:TreeNodeBinding DataMember="chapter" TextField="name"
     ImageUrl="~/images/notepad.gif" />
          </Databindings>

 </asp:TreeView>
```

Vázání dat se provádí v sekci `<Databindings>` prostřednictvím `<asp:TreeNodeBinding>`, který určuje vázaný uzel. Na následujícím obrázku vidíme okno s aplikací.



Obr. 9.6 Zobrazení TreeView

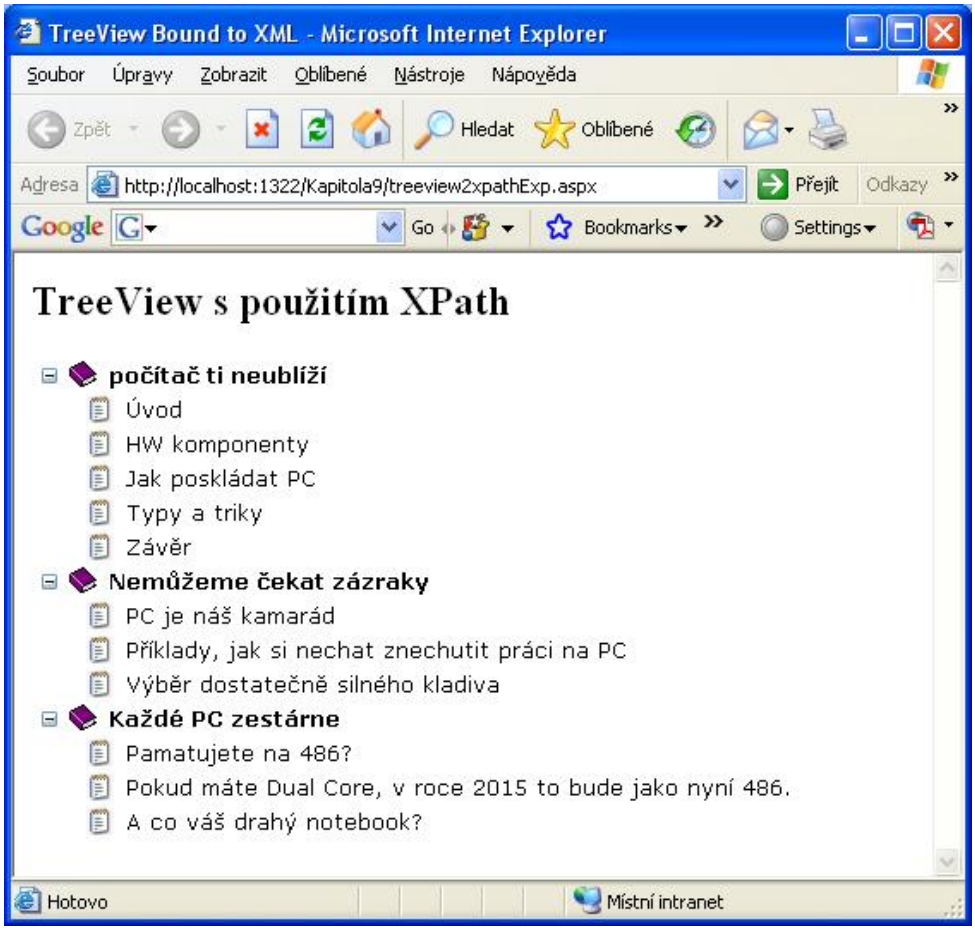
Nyní udělejme v příkladu malou změnu. Ovládací prvek má opět své průvodce v pravém horním rohu - *TreeView Tasks*. Klikneme na konfiguraci datového zdroje a zvolíme XPath výraz:

```
XPath="Bookstore/genre[@name='Technika']/book"
```

Nyní tedy máme v hierarchickém zobrazení knih pouze knihy z žánru Technika. Vygenerovaný

formulář v návrhovém módu i výslednou aplikaci vidíme zde:

Příklad – Formulář s Tree View



Obr. 9.7 Zobrazení TreeView s použitím XPath

Na úplný závěr kapitoly se naladíme na kapitolu následující a ukážeme si příklad, kdy budeme chtít vybrat z databázové tabulky jednu konkrétní hodnotu.
Jako zadání příkladu vezměme v úvahu vyhledání zaměstnance s nejdelší praxí. Formulář realizujme klidně přetažením tabulky zaměstnanců do návrhového módu stránky.

Příklad – Ukázka výběrového dotazu

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString="<%$ ConnectionStrings:DatabaseConnectionString1
%>"
                        ProviderName="<%$
ConnectionStrings:DatabaseConnectionString1.ProviderName %>"
             SelectCommand="SELECT MIN(ZamestnanecNastup) AS Nejdrive
From Zamestnanci"
        </asp:SqlDataSource>
```

Jako výběrový dotaz musíme zvolit minimální hodnotu ze všech záznamů ve sloupci určující nástup do zaměstnání. Protože konfigurujeme *SqlDataSource*, žádný další zdrojový kód psát nemusíme.

Příklad – Formulář k výběru nejmenšího prvku

Co ale musíme ještě zařídit, je zobrazení správného sloupce v *Grid View*. Protože ve výběrovém dotazu jsme provedli ALIAS, to znamená, že výsledek minimální hodnoty je reprezentován termínem *Nejdrive* viz:

```
SelectCommand="SELECT MIN(ZamestnanecNastup) AS Nejdrive From Zamestnanci"
```

Musíme v sekci *Columns* určit toto zobrazované pole v atributu *DataField*.

Příklad – Sloupec v Grid View podle SQL dotazu

```
<Columns>
      <asp:BoundField ItemStyle-HorizontalAlign="Center"
         DataField="Nejdrive" HeaderText="Zaměstanec s nejdelší
         praxí u nás nastoupil v roce" />
</Columns>
```

Na obrázku 9.8 vidíme výsledek zavolaného výběrového dotazu.

Ke kapitole 9 je přiřazena demonstrační animace č. 8

Animace č. 8 obsahuje práci s vázáním dat, demonstruje ovládací prvky Data List, Repeater, TreeView a Grid View.

Obr. 9.8 Příklad výběrového dotazu na minimum ze sloupce

Shrnutí kapitoly

Vázání dat je mechanismus implementovaný v ASP.NET, který dokáže automaticky zobrazovat data. Máme několik možností jak vázat data z datových zdrojů. Jednoduché vázání dat můžeme zajistit i s klasickými ovládacími prvky jako je textbox, label, linkButton a další. Jednoduché vázání dat je realizováno oddělovači <%# a %>.

Jednoduché vázání dat můžeme provést i se zmíněným ovládacím prvkem textbox
```
<asp:textbox ID="tb1" runat="server" Text='<%# retezec2 %>'/>
```
Aby tyto vázané bloky byly funkční, musíme zavolat metodu *DataBind()*. Bez této metody by ovládací prvky neměly co navázat.

S pokročilými ovládacími prvky pro práci s daty úzce souvisí ovládací prvky pro zdroje dat. Použijeme-li ovládací prvek *SqlDataSource* či *AccessDataSource*, nebudeme muset psát kód pro přístup k datům. Pokud je zdrojem dat XML soubor, použijeme *XmlDataSource*. *ObjectDataSource* slouží pro přístup k datům vlastní třídy a *SiteMapDataSource* umožňuje přístup k navigačnímu zdroji dat umístěnému v souboru *Web.SiteMap*.

Přetažením tabulky z průzkumníka serveru do návrhu stránky zobrazíme databázovou tabulku se všemi sloupci tak, jak je nadefinována. To zajistí vlastnost *AutoGenerateColumns* nastavená na *True*. Pokud chceme zobrazit pouze některé sloupce nebo je zobrazit v jiném pořadí, nastavíme tuto vlastnost na *False*. Poté nadefinujeme v sekci <columns> vlastní pořadí vázaných sloupců.

Ovládací prvek *Grid View* je založený na stylech. Tyto styly určují celkový formát tabulky a to pro každý řádek, i pro každý druhý řádek, hlavičku, zápatí editaci, stránkování atd. Tyto styly definují na příslušných řádcích barvy, velikosti, zarovnání, rámování a další známé vlastnosti stylů.

Pokud chceme přizpůsobit obsah zobrazovaných buněk svým potřebám, přidávat do nich různý HTML kód či ovládací prvky, nevystačíme s vázanými sloupci a musíme definovat svou vlastní šablonu ve sloupci *TemplateField*.

Repeater je kontejner určený k cyklickému procházení dat. Je definován bez defaultního vzhledu, vše zajišťujeme sami pomocí šablon. Tyto šablony určí výslednou podobu zobrazených dat. Šablon *Repeatru* je pět:

- *ItemTemplate* – Povinná šablona *Repeatru* vytvářející pro datový záznam jeden řádek výstupu. Vlastní data jsou vázána podle popsaného mechanismu výše.
- *AlternatingItemTemplate* – Šablona pro odlišení lichých a sudých řádků.
- *SeparatorTemplate* – Zobrazuje oddělovač mezi jednotlivými řádky dat.
- *HeaderTemplate* a *FooterTemplate* – Zajišťuje zobrazení hlavičky a zápatí tabulky.

Ovládací prvek *Data List* je mezistupněm mezi *Repeatrem* a *Grid View*. Do jisté míry se podobá *Repeatru*, je také založen na šablonách, obsahuje navíc možnosti komunikace s uživatelem při modifikaci záznamů. Základní šablony jsou shodné s *Repeatrem*, obsahuje však navíc dvě šablony:

- *SelectedItemTemplate* – Obsahuje další elementy pro výběr položky.
- *EditItemTemplate* – Definuje rozložení *Data Listu* v editačním módu.

Vícesloupcové zobrazení záznamů je možné pouze v *Data Listu*, tuto vlastnost nemá ovládací prvek Grid *View*. Proto budeme-li potřebovat zobrazit více záznamů vedle sebe či pod sebe, použijeme v Data Listu vlastnosti:

```
RepeatColumns="PočetSloupců" RepeatDirection="Horizontal|Vertical"
```

Hierarchické vázání docílíme pomocí ovládacího prvku *Tree View*. Ten dokáže zobrazit úplnou strukturu XML dokumentu.Vázání dat se provádí v sekci `<Databindings>` prostřednictvím `<asp:TreeNodeBinding>`, který určuje vázaný uzel.

Úkol k řešení 9.1 – Sloupce v Grid View

Vytvořte databázovou tabulku, ve které máte časové a datumové údaje a pomocí *Grid View* ji zobrazte do prohlížeče. Vyzkoušejte zobrazený formát krátkého a dlouhého data. Zajistěte zobrazení vlastního pořadí sloupců.

Úkol k řešení 9.2 –Vlastní šablona

Vytvořte databázovou tabulku, zobrazte ji pomocí *Grid View* do prohlížeče. Navrhněte vlastní vzhled šablon v sekci *TemplateField*.

Úkol k řešení 9.3 – Repeater

Stejnou databázovou tabulku cyklicky procházejte pomocí *Repeatru*. Navrhněte vlastní šablonu.

Úkol k řešení 9.4 – Data List

Zobrazte databázovou tabulku pomocí *Data Listu*. Vyzkoušejte vícesloupcovou strukturu, vyzkoušejte horizontální i vertikální zobrazení.

Kontrolní otázka 9.1

Jaké jsou možnosti definování vlastní šablony v *Grid View*?

Kontrolní otázka 9.2

Čím je charakteristický ovládací prvek *Repeater*?

Kontrolní otázka 9.3

Čím se liší *Data List* od *Repeatru*?

Kontrolní otázka 9.4

V čem je *Data List* jedinečný?

Kontrolní otázka 9.5

Jakým způsobem realizujeme hierarchické vázání dat?

Kontrolní otázka 9.6

Co je to XPath?

10. POKROČILÉ TECHNIKY PRO PRÁCI S DATY

Čas ke studiu: 2 hodiny

Cíl Po prostudování této kapitoly budete umět

- definovat parametrizované dotazy
- vysvětlit pojem útok SQL injektáží
- vytvářet aplikace s parametrizovaným dotazem
- získávat z tabulek filtrované dotazy
- vytvářet uložené procedury
- programovat aplikace využívající uložené procedury

Výklad

Náš jednosemestrální kurz věnující se technologii ASP.NET se chýlí ke konci, a tak tato třetí kapitola patřící k práci s databázemi je zároveň poslední z této oblasti. Věnuje se poměrně důležitému tématu, kterým jsou parametrizované dotazy a uložené procedury. Bez tohoto aparátu není doporučováno vůbec databázovou aplikaci vypustit do ostrého provozu, především z hlediska bezpečnosti. Součástí kapitoly je opět několik řešených příkladů, které si procvičíme a ukážeme v přiložené animaci.

10.1 Dotazy s parametry

V aplikacích, kdy po uživatelích chceme zadávat data do textboxu pro práci s databázovými tabulkami si musíme uvědomit fakt, že pracujeme s určitým citlivým místem aplikace. Databázová tabulka je naplněna ostrými daty a samozřejmě ne všechny jsou přístupné pro libovolného uživatele. Očekáváme, že uživatel zadá do *textboxu* patřičná data a aplikace mu poskytne odpovídající výsledky. Jenže jazyk SQL je velice mocný nástroj a uživatel nemusí vždy do *textboxu* zadat údaj, který bychom očekávali. Může dokonce nastat situace, kdy uživatel nemá dobré úmysly a záměrně zadává do *textboxu* neočekávané hodnoty a ještě hůře kus SQL kódu, při kterém by mohl získat citlivá data. Tento způsob napadení aplikace je nazýván útok SQL injektáží. Představíme-li si aplikaci, která vytváří SQL dotaz za chodu pomocí apostrofů a uvozovek a dat získaných z *textboxů*, úryvek zdrojového kódu může vypadat například takto:

```
"WHERE Objednavka.ZakaznikID = ' " + tbID.text + " ' " +
"GROUP BY Objednavka.KosikID";
```

Úryvek kódu je konec SQL řetězce, který žádá přihlašovací údaj zákazníka, který je získán z *textboxu* a doplněn do vytvořeného SQL dotazu. Jenže uživatel nezadá své přihlašovací jméno např. *Bond007* ale ***Bond007 ' OR '1' = '1***

V tomto případě bude SQL dotaz vytvořen s tímto fragmentem kódu:

```
WHERE Objednavka.ZakaznikID = 'Bond007' OR '1' = '1'
```

Uživatel tak v podstatě šikovně doplnil sekvenci znaků tak, aby ve výsledku dávala smysluplný význam v jazyce SQL a protože uvedený výraz přeložený takto: (kde zákazník je Bond 007 nebo 1=1),

je platný pro všechny řádky tabulky, uživatel obdrží na výstupu citlivá data, která se k němu nikdy neměla dostat.

Pojem k zapamatování: Špatně napsaná aplikace dovolí útok injektáží SQL

Útok injektáží je postaven na principu, že se do aplikace dostanou data od uživatele, které vývojář neočekával. Proto se dynamicky vytvářející SQL dotaz změní na útočný kód, ze kterého může útočník zjistit citlivá data, či je dokonce zničit.

To byla pouze triviální ukázka injektáže. Jazyk SQL je tak mocný, že jeho konstrukcemi může útočník zajistit i rafinovanější útok jako např. přidání dalšího příkazu do zdrojového kódu apod.

Jaká je tedy obrana? Můžete omezit maximální délku vloženého řetězce ve vlastnostech *textboxu*, můžete omezit zadávání speciálních znaků, avšak nejlepším způsobem obrany je použití parametrizovaných dotazů a uložených procedur.

Pojem k zapamatování: Parametrizovaný dotaz

Parametrizovaný dotaz je příkaz, který používá v SQL řetězci zástupné symboly. Tyto zástupné symboly jsou definovány v sekci <Parameters> a jsou zasílány přímo příkazům SQL.

Jako příklad můžeme uvést následující konstrukci:

SELECT * FROM Objednavky WHERE ZakaznikID = @ZkzID

@ZkzID je parametr, který bude reprezentovat hodnotu v *textboxu* a bude zaslán do SQL příkazu.

Jako příklad parametrizovaných dotazů si vypišme z tabulky zaměstnanců všechny zaměstnance, kteří mají křestní jméno, které zadáme z *textboxu*. Formulář bude obsahovat *label*, *textbox*, tlačítko a *Grid View*.

Příklad – Formulář pro výběr dat z tabulky

Vyber zaměstance s křestnim jménem: Karel OK

Jméno	Příjmení	Adresa	Město	PSČ	Nástup do zaměstnání	Pozice
abc	abc	abc	abc	abc	0	abc
abc	abc	abc	abc	abc	1	abc
abc	abc	abc	abc	abc	2	abc
abc	abc	abc	abc	abc	3	abc
abc	abc	abc	abc	abc	4	abc

Nejprve tedy známým způsobem vytvoříme *Grid View* přetažením tabulky zaměstnanců do návrhového módu stránky. Poté v *GridViewTasks* nakonfigurujeme datový zdroj. Ten opět budeme tvořit pomocí vestavěného průvodce. Visual Studio poskytuje intuitivní průvodce i pro tvorbu SQL dotazů za použití parametrů. Postupujeme tedy známým způsobem až do doby, kdy máme potvrdit výsledný SQL příkaz.

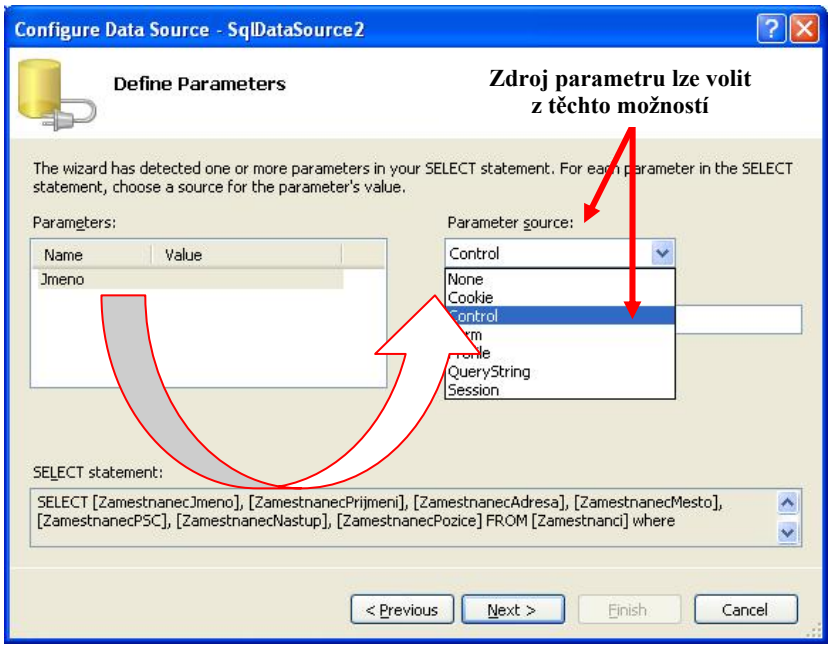
Příklad – Grid View tabulky zaměstnanců

```
<asp:GridView ID="GridView2" runat="server"
AutoGenerateColumns="False"
DataSourceID="SqlDataSource2"
EmptyDataText=" Nejsou k dispozici žádná data.">
        <FooterStyle BackColor="#990000" Font-Bold="True" />
        <RowStyle BackColor="#FFFBD6" ForeColor="#333333" />
        <SelectedRowStyle BackColor="#FFCC66" Font-Bold="True" />
        <PagerStyle BackColor="#FFCC66" HorizontalAlign="Center" />
        <HeaderStyle BackColor="#990000" Font-Bold="True" />
        <AlternatingRowStyle BackColor="White" />
<Columns>
<asp:BoundField DataField="ZamestnanecJmeno" HeaderText="Jméno" />
<asp:BoundField DataField="ZamestnanecPrijmeni" HeaderText="Příjmení"/>
<asp:BoundField DataField="ZamestnanecAdresa" HeaderText="Adresa" />
<asp:BoundField DataField="ZamestnanecMesto" HeaderText="Město" />
<asp:BoundField DataField="ZamestnanecPSC" HeaderText="PSČ" />
<asp:BoundField DataField="ZamestnanecNastup" HeaderText="Nástup do
                          zaměstnání" />
<asp:BoundField DataField="ZamestnanecPozice" HeaderText="Pozice" />
</Columns>
</asp:GridView>
```

Pro náš SQL dotaz doplníme nakonec:

WHERE ZamestnanecJmeno = @Jmeno.

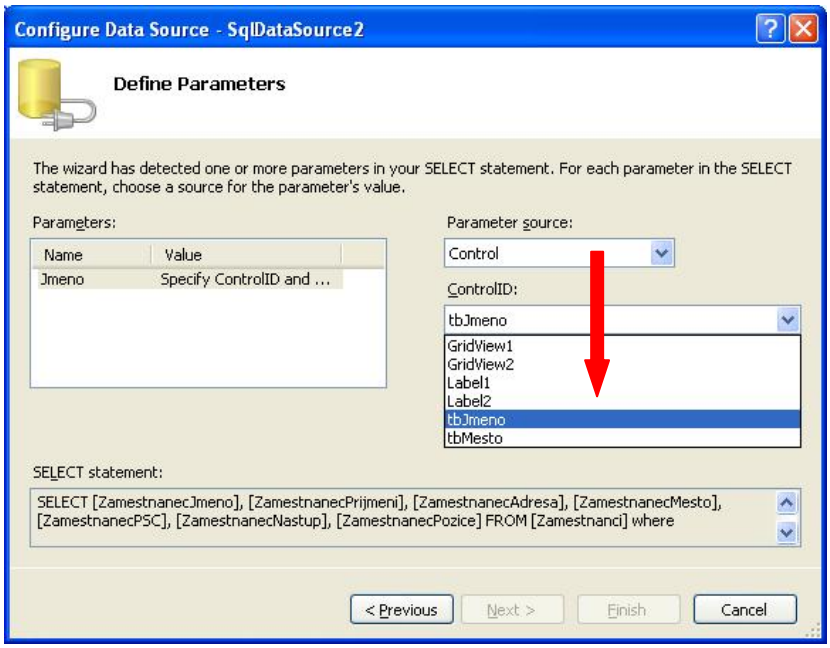
Vytvořili jsme tak parametr *@Jmeno*. Při dalším kroku průvodce ihned rozpozná, že se pokoušíme vytvořit parametrizovaný dotaz a nabídne nám v rozbalovacím seznamu zdroj, odkud budeme chtít parametr získat (viz. obr. 10.1).



Obr. 10.1 Definice parametru pro SQL dotaz – volba zdroje

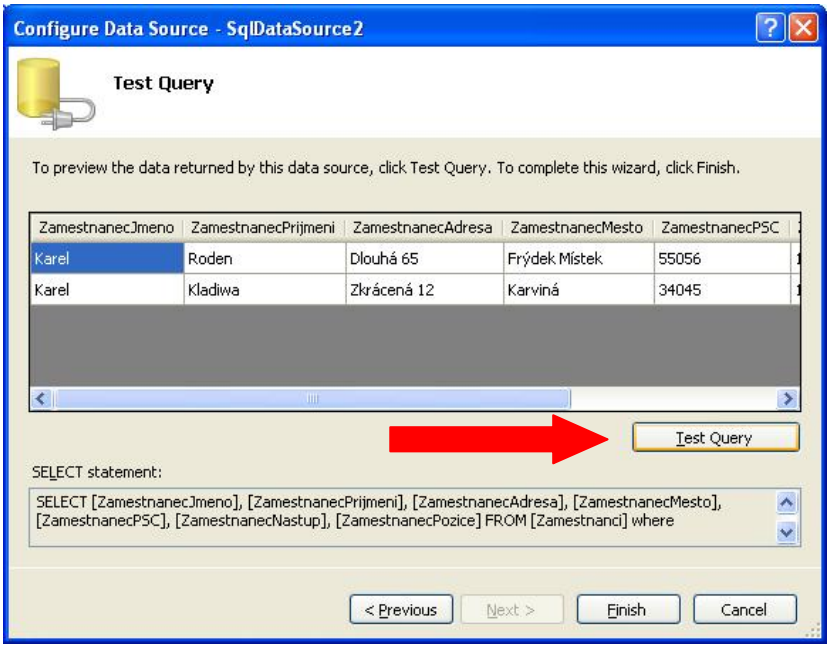
Máme na výběr několik možností, protože parametry nemusí zadávat přímo uživatel, můžeme je také získat např. z proměnné relace či cookies. My ale parametrizovanou hodnotu získáme z ovládacího prvku *textbox*, proto zvolíme volbu *Control*.

V dalším kroku průvodce vyhledá všechny dosud naimplementované ovládací prvky v naší stránce a zobrazí jejich seznam.



Obr. 10.2 Definice parametru pro SQL dotaz – volba ovládacího prvku

Zvolíme *textbox* připravený k získání křestního jména zaměstnance. Potvrdíme volbu a průvodce dokončí tvorbu dotazu. Výsledný dotaz můžeme otestovat ještě dříve, než provedeme poslední krok k dokončení celé akce.



Obr. 10.3 Otestování výsledného dotazu

Pohledem na datový zdroj zjistíme několik skutečností. Vlastnost *SelectCommand* je ve tvaru parametrizovaného dotazu a vygenerovaná sekce `<SelectParameters>` obsahuje parametr, který jsme uvedli v průvodci. Až nám tyto situace budou připadat jako samozřejmé, nemusíme průvodce používat a uvedené atributy můžeme generovat přímo ve zdrojovém kódu stránky.

Ještě než si ukážeme vygenerovanou aplikaci, ověřme si naše znalosti na tvorbě druhého formuláře. Pokusíme se parametrizovaným dotazem zjistit počet studentů z konkrétního města. Město zadáme opět z textového pole.

Příklad – Vytvořený datový zdroj s parametrizovaným dotazem – Př. 1

```
<asp:SqlDataSource ID="SqlDataSource2" runat="server"
ConnectionString="<%$ ConnectionStrings:DbaseConnStr1 %>"
ProviderName="<%$ ConnectionStrings:DbaseConnStr1.ProviderName %>"
SelectCommand="SELECT [ZamestnanecJmeno], [ZamestnanecPrijmeni],
[ZamestnanecAdresa], [ZamestnanecMesto], [ZamestnanecPSC],
[ZamestnanecNastup], [ZamestnanecPozice] FROM [Zamestnanci] where
ZamestnanecJmeno = @Jmeno">
       <SelectParameters>
       <asp:ControlParameter ControlID="tbJmeno"
       DefaultValue="Karel" Name="Jmeno" PropertyName="Text" />
       </SelectParameters>
</asp:SqlDataSource>
```

Datová tabulka bude obsahovat pouze jeden vázaný sloupec informující o počtu studentů. Formulář je jednoduchý, opět si vystačíme s popiskem, *textboxem* a tlačítkem.

Příklad – Grid View pro zobrazení počtu studentů

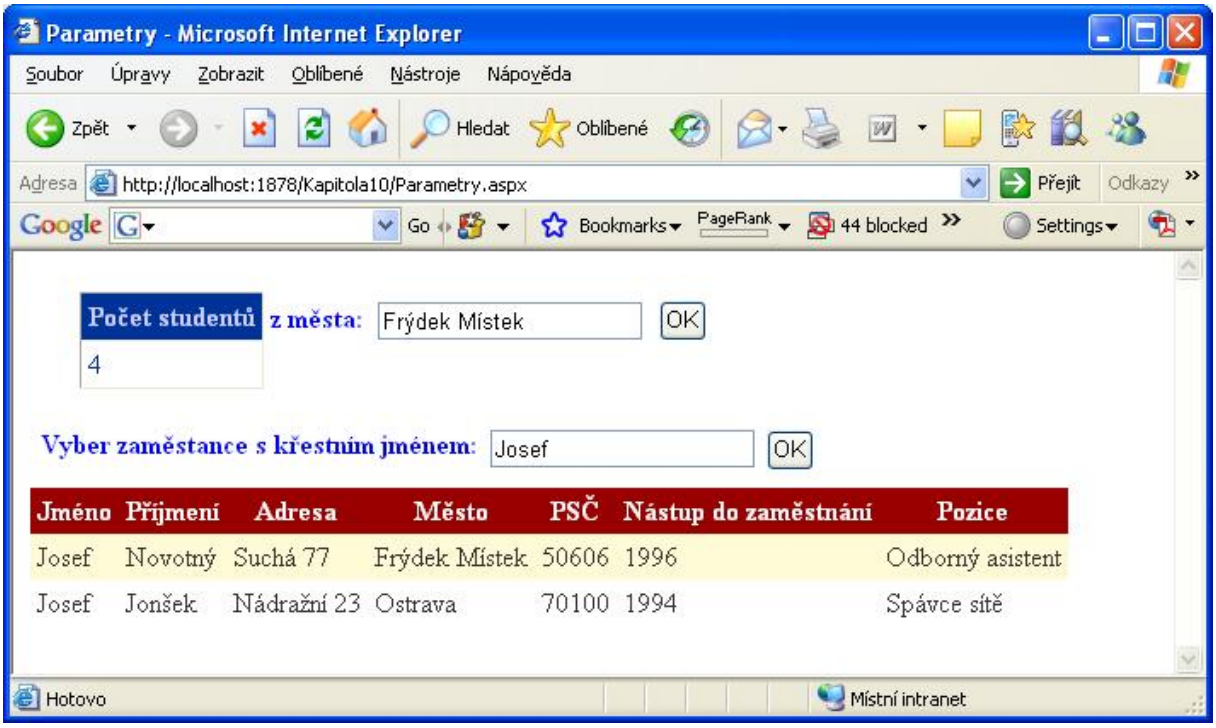
```
<asp:GridView ID="GridView1" runat="server"
AutoGenerateColumns="False"
DataSourceID="SqlDataSource1"
EmptyDataText="Nejsou k dispozici žádná data." >

   <FooterStyle BackColor="#99CCCC" ForeColor="#003399" />
   <RowStyle BackColor="White" ForeColor="#003399" />
   <SelectedRowStyle BackColor="#009999" Font-Bold="True" />
   <PagerStyle BackColor="#99CCCC" HorizontalAlign="Left" />
   <HeaderStyle BackColor="#003399" Font-Bold="True" />

   <Columns >
   <asp:BoundField DataField="Pocet" HeaderText="Počet studentů" />
   </Columns>
</asp:GridView>
```

V *GridViewTasks* opět nakonfigurujeme zdroj dat. Obdobným způsobem jako z předchozího příkladu nadefinujeme parametr ***@StudMesto***. Tento dotaz je však odlišný, hledáme počet studentů, a tak si pečlivě prohlédněte *SelectCommand* datového zdroje.

Příklad – Vytvořený datový zdroj s parametrizovaným dotazem – Př. 2

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString="<%$ ConnectionStrings:DbaseConnStr1 %>"
ProviderName="<%$ ConnectionStrings:DbaseConnStr1.ProviderName %>"
SelectCommand="SELECT COUNT(studJmeno) AS Pocet FROM Studenti WHERE
StudMesto = @StudMesto">
    <SelectParameters>
    <asp:ControlParameter ControlID="tbMesto" DefaultValue="Ostrava"
    Name="StudMesto" PropertyName="Text" />
     </SelectParameters>
</asp:SqlDataSource>
```

Po otestování vygenerovaného dotazu a potvrzení posledního kroku průvodce můžeme spustit hotovou aplikaci. Ta obsahuje oba příklady. Užitečnými vlastnostmi jsou `EmptyDataText` ovládacího prvku *Grid View* pro případ, kdy žádná data nevyhovují podmínce, popřípadě `DefaultValue` v parametru dotazu, kdy docílíme implicitní hodnoty výběrového dotazu například při prvním spuštění.



Obr. 10.4 Výsledná aplikace s parametrizovanými dotazy

10.2 Uložené procedury

Uložená procedura je dávka obsahující jeden či celou sadu SQL dotazů. Je uložena přímo v databázi. Můžeme je chápat jako zapouzdřené funkce, které prostřednictvím vstupních parametrů přebírají data a poskytují ucelenou sadu výsledků. Jejich použití je velmi doporučováno. Největšími přednostmi uložených procedur jsou:

- **Přehledná a snadná údržba**

  Protože uložené procedury nejsou součástí zdrojového kódu aplikace, nemusíme při jejich editaci aplikaci znovu kompilovat. Výhodou je rovněž přehledná administrace na jednom místě v Server Exploreru.

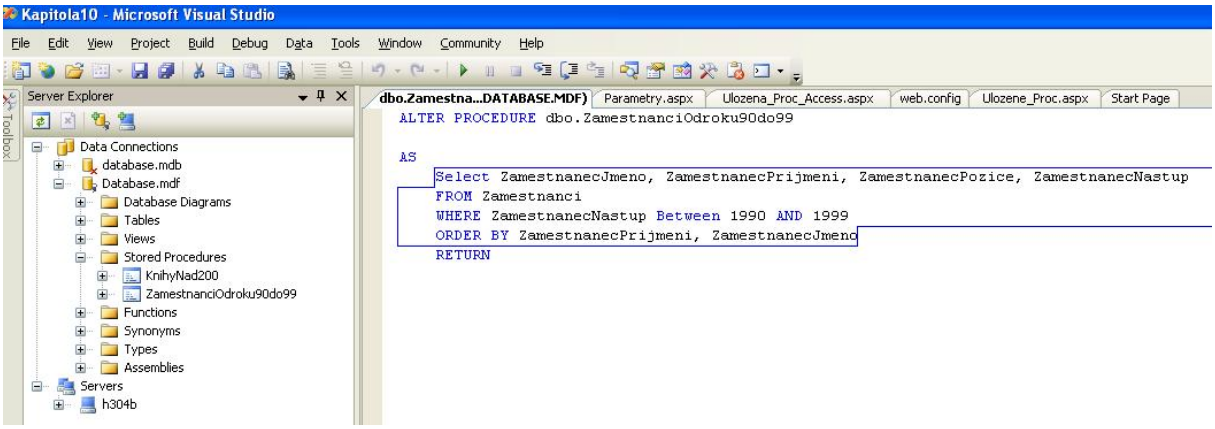
- **Bezpečnost aplikace využívající databázi**

  V úvodu kapitoly již byly zmíněny některé nepříliš bezpečné situace při generování SQL dotazů přímo ve zdrojovém kódu aplikace.

- **Zvýšení výkonu**

  Uložené procedury jsou dávkovým proudem příkazů, který může při jednom připojení na databázový server zastat mnoho práce. To oceníme zejména v situacích, kdy databázový server běží na jiném stroji.

Vraťme se k našemu projektu. Uložené procedury jsou součástí databázového *mdf* souboru, takže je najdeme v Server Exploreru. Podobně jako tabulky jsou součástí rozbalovacího menu a pravým tlačítkem na složce *Stored Procedures* můžeme provádět nabízené akce, jako např. vytvoření nové uložené procedury.



Obr. 10.5 Formát uložené procedury ve formě SQL dotazu

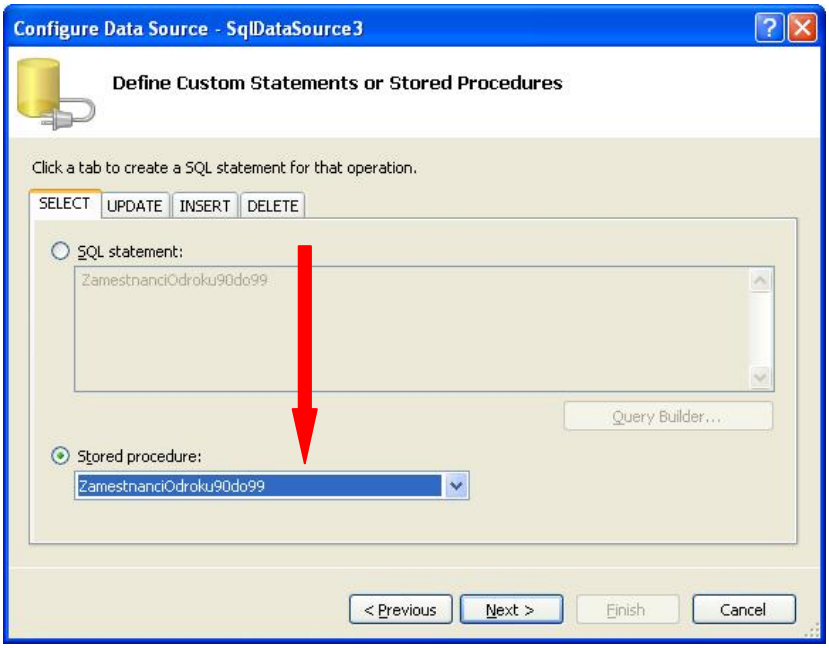
Při konfiguraci datového zdroje, který je nám již dobře známou posloupností akcí, nám ještě zbývá právě možnost výběru uložené procedury.

Našim úkolem bude vytvořit zobrazení všech zaměstnanců z databáze, kteří nastoupili do zaměstnání mezi rokem 1990 a 1999. Nejprve vytvořme v Server Exploreru uloženou proceduru, jejíž výpis vidíme v následujícím příkladu.

Příklad – Uložená procedura – Př.1

```
CREATE PROCEDURE dbo.ZamestnanciOdroku90do99
      AS
      Select ZamestnanecJmeno, ZamestnanecPrijmeni,
ZamestnanecPozice, ZamestnanecNastup
      FROM Zamestnanci
      WHERE ZamestnanecNastup Between 1990 AND 1999
      ORDER BY ZamestnanecPrijmeni, ZamestnanecJmeno
```

Necháme tedy zobrazit pouze jméno a příjmení zaměstnance, jeho pracovní zařazení a rok, kdy nastoupil do zaměstnání. To vše seřadíme podle abecedy.



Obr. 10.6 Dotaz provedeme z uložené procedury

Při konfiguraci datového zdroje pomocí známého průvodce můžeme místo SQL příkazu nastavit možnost dotazovacího řetězce z uložené procedury. Pokud tuto volbu zvolíme, průvodce vyhledá všechny uložené procedury náležící datovému spojení a nabídne je do *listboxu*. Zvolíme tedy naši uloženou proceduru pro zobrazení zaměstnanců (viz. obr. 10.6)

Poté již můžeme pokračovat průvodcem standardním postupem. Po jeho ukončení můžeme vidět nakonfigurovaný datový zdroj s uloženou procedurou. Ten je společně s ovládacím prvkem *Grid View* zobrazen v následujících příkladech.

Příklad – Grid View pro zobrazení zaměstnanců

```
<asp:GridView ID="GridView2" runat="server"
AutoGenerateColumns="False"
DataSourceID="SqlDataSource2"
EmptyDataText="There are no data records to display." >
        <FooterStyle BackColor="#99CCCC" ForeColor="#003399" />
        <RowStyle BackColor="White" ForeColor="#003399" />
         <SelectedRowStyle BackColor="#009999" Font-Bold="True" />
         <PagerStyle BackColor="#99CCCC" HorizontalAlign="Left" />
         <HeaderStyle BackColor="#003399" Font-Bold="True" />
  <Columns>
<asp:BoundField DataField="ZamestnanecJmeno" HeaderText="Jméno" />
<asp:BoundField DataField="ZamestnanecPrijmeni"
    HeaderText="Příjmení" />
<asp:BoundField DataField="ZamestnanecPozice" HeaderText="Pozice" />
<asp:BoundField DataField="ZamestnanecNastup" HeaderText="Nástup v
    roce" />
</Columns>
</asp:GridView>
```

Příklad – Datový zdroj s uloženou procedurou výběru zaměstnanců

```
<asp:SqlDataSource ID="SqlDataSource2" runat="server"
ConnectionString="<%$ ConnectionStrings:DbaseConnStr1 %>"
ProviderName="<%$ ConnectionStrings:DbaseConnStr1.ProviderName %>"
SelectCommand="ZamestnanciOdroku90do99"
SelectCommandType="StoredProcedure">
</asp:SqlDataSource>
```

Je naším zvykem procvičit si danou problematiku alespoň na dvou příkladech. Než tedy spustíme výslednou aplikaci, procvičíme si nejen procedury ale i vlastní tvorbu výběrových dotazů na druhém příkladu. Protože v naší databázi máme nadefinováno více tabulek, zobrazme například z tabulky knih všechny knihy, které stojí více než 200 Kč. Výsledný formulář celé aplikace se vlastně skládá pouze z *labelů*, datových zdrojů a *Grid View*.

Příklad – Uložená procedura – Př. 2

```
ALTER PROCEDURE dbo.KnihyNad200
AS
      SELECT Nazev, Cena, Typ
FROM Knihy
WHERE Cena >= 200
ORDER BY Typ
```

Hledáme tedy všechny knihy, které ve sloupci cena mají hodnotu větší nebo rovno 200. Uložená procedura se stále nachází ve stejném databázovém souboru *mdf*.

Příklad – Formulář k aplikaci s uloženými procedurami

Knihy nad 200 Kč

Název	Cena	Kategorie
abc	0	Databound
abc	0,1	Databound
abc	0,2	Databound
1 2		

Zaměstnanci, kteří nastoupili do zaměstnání od roku 1990 do roku 1999

Jméno	Příjmení	Pozice	Nástup v roce
abc	abc	abc	0
abc	abc	abc	1
abc	abc	abc	2
abc	abc	abc	3
abc	abc	abc	4

Stejným postupem jako v předchozím příkladě uložíme proceduru v Server Exploreru a v návrhovém zobrazení konfigurujeme *Grid View* a datový zdroj.

Příklad – Grid View pro zobrazení knih

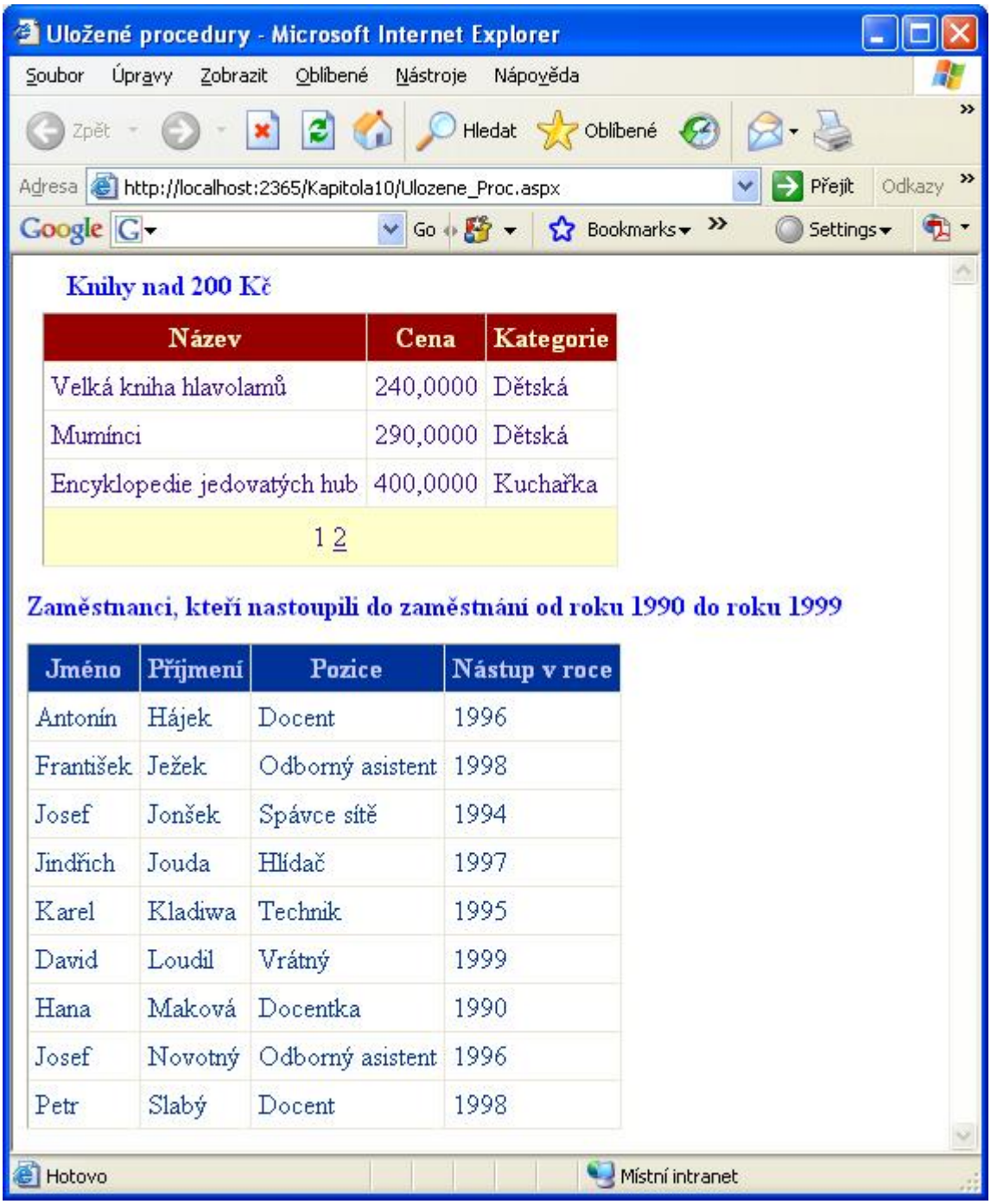
```
<asp:GridView ID="GridView1" runat="server"
AutoGenerateColumns="False"
DataSourceID="SqlDataSource1"
EmptyDataText="There are no data records to display. "
AllowPaging="True"
PageSize="3">
         <FooterStyle BackColor="#FFFFCC" ForeColor="#330099" />
         <SelectedRowStyle BackColor="#FFCC66" Font-Bold="True" />
         <PagerStyle BackColor="#FFFFCC" HorizontalAlign="Center" />
         <HeaderStyle BackColor="#990000" Font-Bold="True" />
    <Columns >
     <asp:BoundField DataField="Nazev" HeaderText="Název" />
     <asp:BoundField DataField="Cena" HeaderText="Cena" />
     <asp:BoundField DataField="Typ" HeaderText="Kategorie" />
    </Columns>
    <RowStyle BackColor="White" ForeColor="#330099" />
</asp:GridView>
```

Pomocí průvodce opět zvolíme volbu *Stored Procedures* a v nabízeném seznamu zvolíme proceduru KnihyNad200. Výsledný datový zdroj vidíme zde:

Příklad – Datový zdroj s uloženou procedurou výběru knih

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString="<%$ ConnectionStrings:DbaseConnStr1 %>"
ProviderName="<%$ ConnectionStrings:DbaseConnStr1.ProviderName %>"
SelectCommand="KnihyNad200"
SelectCommandType="StoredProcedure" >
</asp:SqlDataSource>
```

Oba příklady jsou hotové, nyní můžeme spustit aplikaci v prohlížeči. Na obrázku 10.7 vidíme výsledek, který bychom měli po provedení obou uložených procedur obdržet.

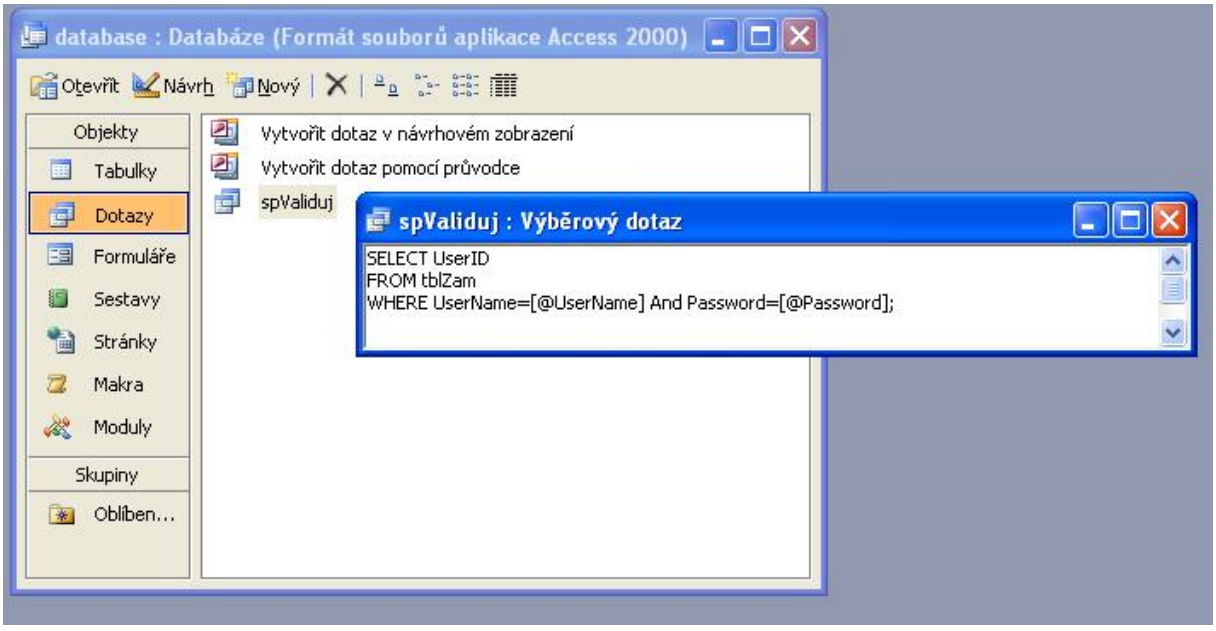


Obr. 10.7 Výsledná aplikace příkladu se dvěma uloženými procedurami

Závěrečný příklad ukáže poněkud odlišný přístup k parametrizovanému dotazu a uložené proceduře. Uvedené skutečnosti obsloužíme ručně psaným kódem a jako datový zdroj zvolíme tabulku MS Accessu. Definujme si nejprve zadání. V tabulce MS Access, se kterou pracujeme v našich projektech, máme pole uživatelského jména a hesla. Využijme parametrizovaného dotazu a uložené procedury k jednoduché aplikaci validace uživatelů. Nejprve vytvoříme formulář, skládající se z popisků, dvou *textboxů* pro jméno a heslo a potvrzovacího tlačítka.

Příklad – Formulářk příkladu uložené procedury v MS Access

Uložená procedura bude v záložce *Dotazy* naší databázové tabulky pod názvem *spValiduj* a bude mít následující tvar (viz. obr. 10.8).

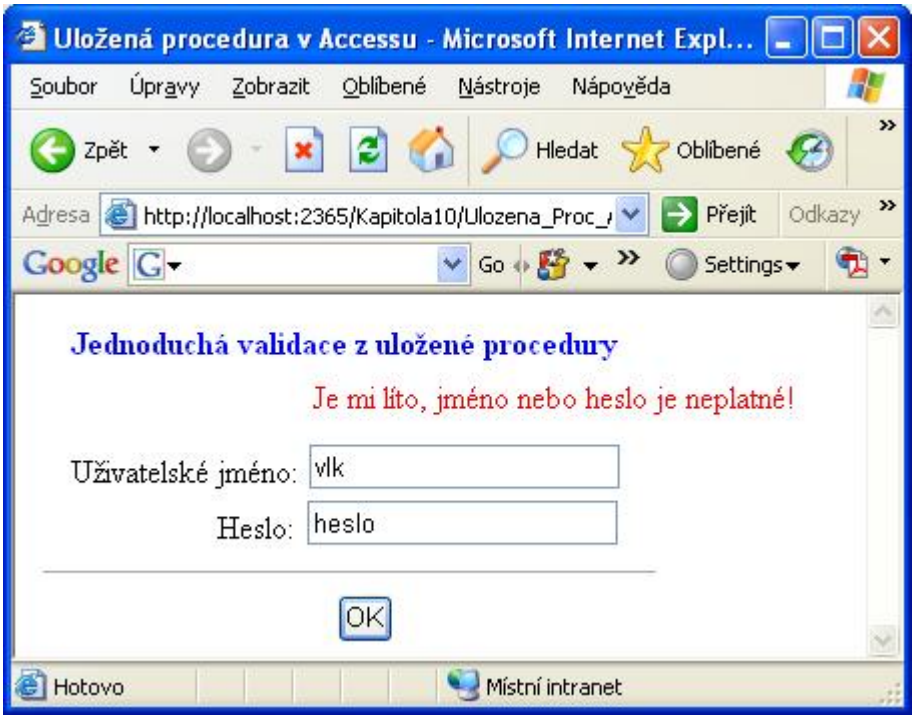


Obr. 10.8 Uložená procedura v MS Access

Do uložené procedury vstupují dva parametry z *textboxu,* jméno a heslo. Po kliknutí na tlačítko OK máme za úkol spustit uloženou proceduru. Všimneme si definice objektu *OleDbCommand*, který místo SQL dotazu má na místě prvního parametru uloženou proceduru *spValiduj*. V dalších definicích musíme určit typ příkazu na uloženou proceduru a nadefinovat parametry `@UserName` a `@Password`. Oba jsou typu *OleDbType.BSTR*, což znamená řetězcový typ.

Příklad – Procedura login využívající uložené procedury spValiduj

```
Sub klikLogin(ByVal obj As Object, ByVal e As EventArgs)
   Dim intID As Integer = 0
    Dim Conn As New OleDbConnection("Provider =
    Microsoft.Jet.OLEDB.4.0;
    Data Source=D:\UT_Příklady\Kapitola10\App_Data\database.mdb")
   Dim objCmd As OleDbCommand = New OleDbCommand("spValiduj", Conn)
   objCmd.CommandType = CommandType.StoredProcedure
   Dim objParam As OleDbParameter
   objParam = objCmd.Parameters.Add("@UserName", OleDbType.BSTR)
   objParam.Value = tbLogin.Text
   objParam = objCmd.Parameters.Add("@Password", OleDbType.BSTR)
   objParam.Value = tbHeslo.Text
    Try
       objCmd.Connection.Open()
       intID = CType(objCmd.ExecuteScalar, Integer)
       objCmd.Connection.Close()
    Catch ex As OleDbException
       lblMessage.Text = ex.Message
    End Try
       If intID <> 0 Then
          Response.Cookies("Account").Value = intID
          lblMessage.Text = "Heslo je správné"
    Else
          lblMessage.Text = "<font color=red> Je mi líto, jméno nebo
          heslo je neplatné! </font><p>"
    End If
End Sub
```



Obr. 10.9 Výsledná aplikace validace dat z uložené procedury v MS ACCESS

Ke kapitole 10 je přiřazena demonstrační animace č. 9

Animace č. 9 obsahuje příklady této kapitoly zahrnující práci s parametrizovanými dotazy a uloženými procedurami.

Shrnutí kapitoly

Útok injektáží je postaven na principu, že se do aplikace dostanou data od uživatele, která vývojář neočekával. Proto se dynamicky vytvářející SQL dotaz změní na útočný kód, ze kterého může útočník zjistit citlivá data, či je dokonce zničit.

Jaká je proti němu obrana? Můžete omezit maximální délku vloženého řetězce ve vlastnostech textboxu, můžete omezit zadávání speciálních znaků, avšak nejlepším způsobem obrany je použití parametrizovaných dotazů a uložených procedur.

Parametrizovaný dotaz je příkaz, který používá v SQL řetězci zástupné symboly. Tyto zástupné symboly jsou definovány v sekci <Parameters> a jsou zasílány přímo příkazům SQL.

Visual Studio poskytuje intuitivní průvodce i pro tvorbu SQL dotazů za použití parametrů. Pohledem na datový zdroj zjistíme několik skutečností. Vlastnost *SelectCommand* je ve tvaru parametrizovaného dotazu a vygenerovaná sekce `<SelectParameters>` obsahuje parametr, který jsme uvedli v průvodci.

Uložená procedura je dávka obsahující jeden či celou sadu SQL dotazů. Je uložena přímo v databázi. Můžeme je chápat jako zapouzdřené funkce, které prostřednictvím vstupních parametrů přebírají data a poskytují ucelenou sadu výsledků. Jejich použití je velmi doporučováno. Největšími přednostmi uložených procedur jsou:

- **Přehledná a snadná údržba**

  Protože uložené procedury nejsou součástí zdrojového kódu aplikace, nemusíme při jejich editaci aplikaci znovu kompilovat. Výhodou je rovněž přehledná administrace na jednom místě v Server Exploreru.

- **Bezpečnost aplikace využívající databázi**

  V úvodu kapitoly již byly zmíněny některé nepříliš bezpečné situace při generování SQL dotazů přímo ve zdrojovém kódu aplikace.

- **Zvýšení výkonu**

  Uložené procedury jsou dávkovým proudem příkazů, který může při jednom připojení na databázový server zastat mnoho práce.

Úkol k řešení 10.1 – Parametrizované dotazy

Zobrazte pomocí parametrizovaných dotazů tabulku knih s cenou, která odpovídá rozsahu zadaného uživatelem, např. 200 – 250 Kč.

Úkol k řešení 10.2 – Parametrizované dotazy

Zobrazte jména studentů a zaměstnanců z města, které zvolí uživatel (např. z listboxu) pomocí parametrizovaného dotazu.

Úkol k řešení 10.3 – Uložené procedury

Vytvořte aplikaci, která zobrazí do tabulky knihy podle typu (dětská, počítačová, kuchařka), který zvolí uživatel. Využijte tvorbu uložených procedur.

Úkol k řešení 10.4 – Uložené procedury

Doplňte do tabulky studentů sloupec datum narození a zobrazte nejstaršího a nejmladšího studenta pomocí uložených procedur.

Kontrolní otázka 10.1

Co je to útok injektáží SQL?

Kontrolní otázka 10.2

Co je to parametrizovaný dotaz?

Kontrolní otázka 10.3

Jakým způsobem vytváříme parametrizované dotazy?

Kontrolní otázka 10.4

Jaký význam mají uložené procedury?

Kontrolní otázka 10.5

Jakým způsobem vytváříme aplikace s uloženými procedurami?

11. VÝVOJ WEBOVÉ SLUŽBY

Čas ke studiu: 2 hodiny

Cíl Po prostudování této kapitoly budete umět

- vysvětlit pojem *webová služba* a její význam
- napsat vlastní webovou službu
- definovat standardy webových služeb
- otestovat webovou službu přímo ve Visual Studiu
- vytvořit vlastní testovací prostředí
- definovat pojem *přístupová proxy třída*
- naprogramovat vlastní klientskou aplikaci využívající webovou službu

Výklad

Semestrální kurz věnovaný technologii ASP.NET se blíží ke konci. Ale byli byste určitě ochuzeni o vědomosti, pokud bychom si nepředstavili webové služby. Tento moderní nástroj se neustále dostává do popředí zájmu vývojářů a softwarových firem. Věnujme mu tedy alespoň dvě praktické kapitoly.

11.1 Vývoj webových služeb

Uveďme rychlý úvod do problému. Představte si situaci, kdy administrujete nějaký webový portál burzovního zpravodajství či třeba jen cestovní kanceláře. Máte na výběr zaměstnat člověka, který bude denně aktualizovat hodnoty kurzů akcií nebo počasí v různých rekreačních letoviscích. Tento způsob je jistě možný a hojně využívaný, nicméně bude určitě časově náročný a nezanedbatelná bude i finanční stránka. Druhou možností je konzumovat nějakou webovou službu, která tyto data automaticky poskytuje sama bez nutnosti ruční administrace. Samozřejmě tato webová služba může být placená, ale je otázkou jestli nebude výhodnější než další zaměstnanec.

V praxi by to vypadalo tak, že do webové stránky vaší cestovní kanceláře putuje informace o počasí ze vzdálené webové služby, přesto o tom koncoví zákazníci nemají ani potuchy, proč taky, aplikace funguje a jak je naimplementována, je vaše věc. To je jen malý příklad, webové služby mají obrovský potenciál vyměňovat data různého typu mezi aplikacemi.

Webové služby umožňují sdílení komponent a jejich funkcí. Je to v podstatě nejlépe demonstrovaná možnost znovupoužití kódu. Nespornou výhodou je údržba vaší aplikace. Nemusíte již instalovat řady programů a aplikačních objektů. Jednoduše je vzdáleně využíváte.

Pojem k zapamatování: Webová služba

Webová služba je novým mechanismem pro výměnu dat mezi aplikacemi. Pomocí XML a dalších technologií umožňuje komunikaci aplikace s jinou aplikací, která ji konzumuje vzdáleně bez ohledu na to, na jakém hardwaru a softwaru je implementována.

Nyní vytvoříme jednoduchou web službu ve Visual Studiu. Nejprve tedy založíme projekt a jako typ

zvolíme nikoliv *new ASP.NET web Site*, nýbrž *ASP.NET web Service*. Pokud již projekt vytvořený máme, webovou službu můžeme vytvořit položkou *Add New Item*. Všimneme si, že se neotevřela stránka *aspx* ale stránka *asmx*, což je přípona souboru s web službou. Také úvodní direktiva již není *@Page* ale *@WebService*. Vytvořená webová služba bude mít zdrojový kód v pozadí, to znamená v souboru s příponou vb.

Příklad – Vytvoření web služby jednoduchého kalkulátoru

```
<%@ WebService Language="VB" CodeBehind="~/App_Code/Kalkulator.vb"
Class="Kalkulator" %>
```

V souboru se zdrojovým kódem musíme naimportovat nejprve příslušné jmenné prostory. Pokud vložíme do projektu novou web službu, defaultně se nám vytvoří služba, která vrací textový řetězec *hello world*.

Příklad – Triviální implicitní web service helloworld

```
Imports System.Web
Imports System.Web.Services
Imports System.Web.Services.Protocols

<WebService(Namespace:="http://tempuri.org/")> _
<WebServiceBinding(ConformsTo:=WsiProfiles.BasicProfile1_1)> _
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()>
_
Public Class WebService
     Inherits System.Web.Services.WebService

    <WebMethod()> _
    Public Function HelloWorld() As String
        Return "Hello World"
    End Function

End Class
```

Ze zdrojového kódu vidíme, že při tvorbě vlastní web služby musíme vytvořit vlastní třídu, která bude obsahovat veřejné funkce definované jako *WebMethod()*. Vytvořme tedy naši vlastní webovou službu, implicitní kód smažeme, ponecháme import jmenných prostorů a vytvoříme třídu *Kalkulator*.

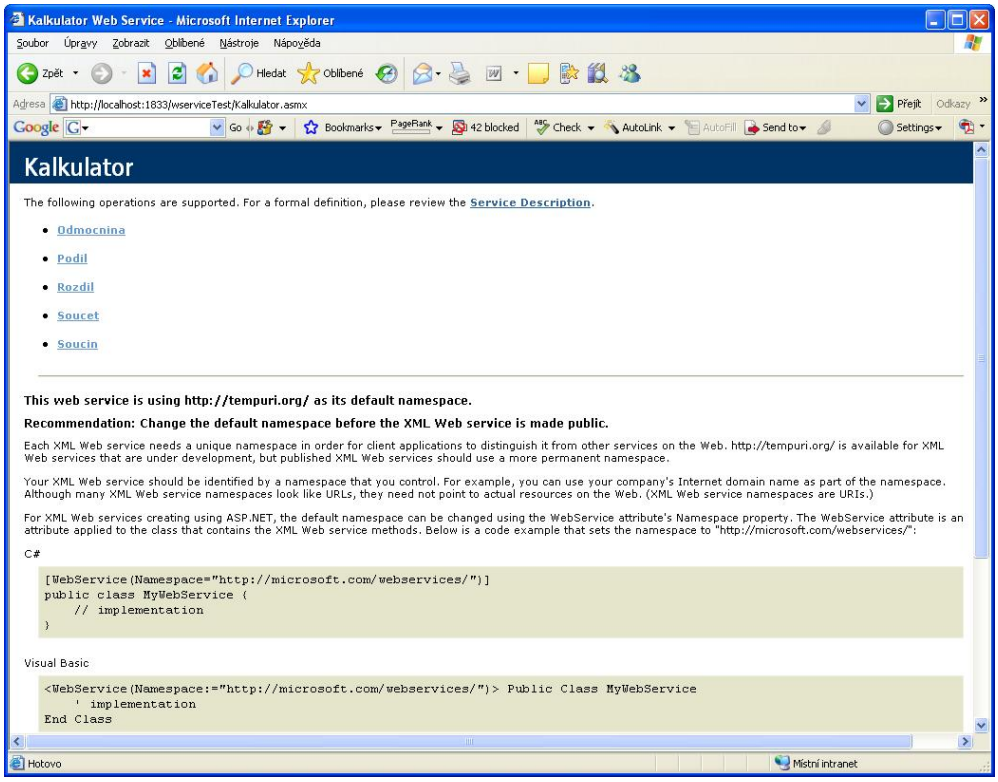
Naše webová služba kalkulátoru bude poskytovat čtyři základní aritmetické operace součtu, rozdílu, součinu, podílu a jedné další operace, kterou bude odmocnina. Zdrojový kód vidíme v následujícím příkladu.

Příklad – Metody webové služby kalkulátoru

```
Imports System.Web
Imports System.Web.Services
Imports System.Web.Services.Protocols


Public Class Kalkulator : Inherits WebService
    <WebMethod()> Public Function Soucet(ByVal promA As Single,
ByVal promB As Single) As Single
        Return (promA + promB)
    End Function
    <WebMethod()> Public Function Rozdil(ByVal promA As Single,
```

```
ByVal promB As Single) As Single
        Return (promA - promB)
    End Function
    <WebMethod()> Public Function Soucin(ByVal promA As Single,
ByVal promB As Single) As Single
        Return (promA * promB)
    End Function
    <WebMethod()> Public Function Podil(ByVal promA As Single, ByVal
promB As Single) As Single
        Return (promA / promB)
    End Function
    <WebMethod()> Public Function Odmocnina(ByVal promA As Single)
As Single
        Return (Math.Sqrt(promA))
    End Function
End Class
```

Webovou službu máme možnost ve Visual Studiu ihned otestovat bez tvorby klientské aplikace. Klikneme pravým tlačítkem na soubor se službou s příponou *asmx* a nastavíme ji jako startovací stránku, popřípadě ihned zvolíme možnost *View in Browser*. Webová služba se spustí v prohlížeči s připraveným prostředím, ve kterém vidíme jednotlivé metody, které můžeme otestovat. Kalkulátor se seznamem implementovaných metod vidíme na obrázku 11.1.



Obr. 11.1 Defaultní prostředí při spuštění web služby asmx ve Visual Studiu

Odkud se ale vzalo prostředí, které vidíme při testování služby?

Testovací stránka je realizována za chodu vygenerováním stránky DefaultWsdlHelpGenerator.aspx, kterou najdete v adresáři, kde máte nainstalovaný operační systém:

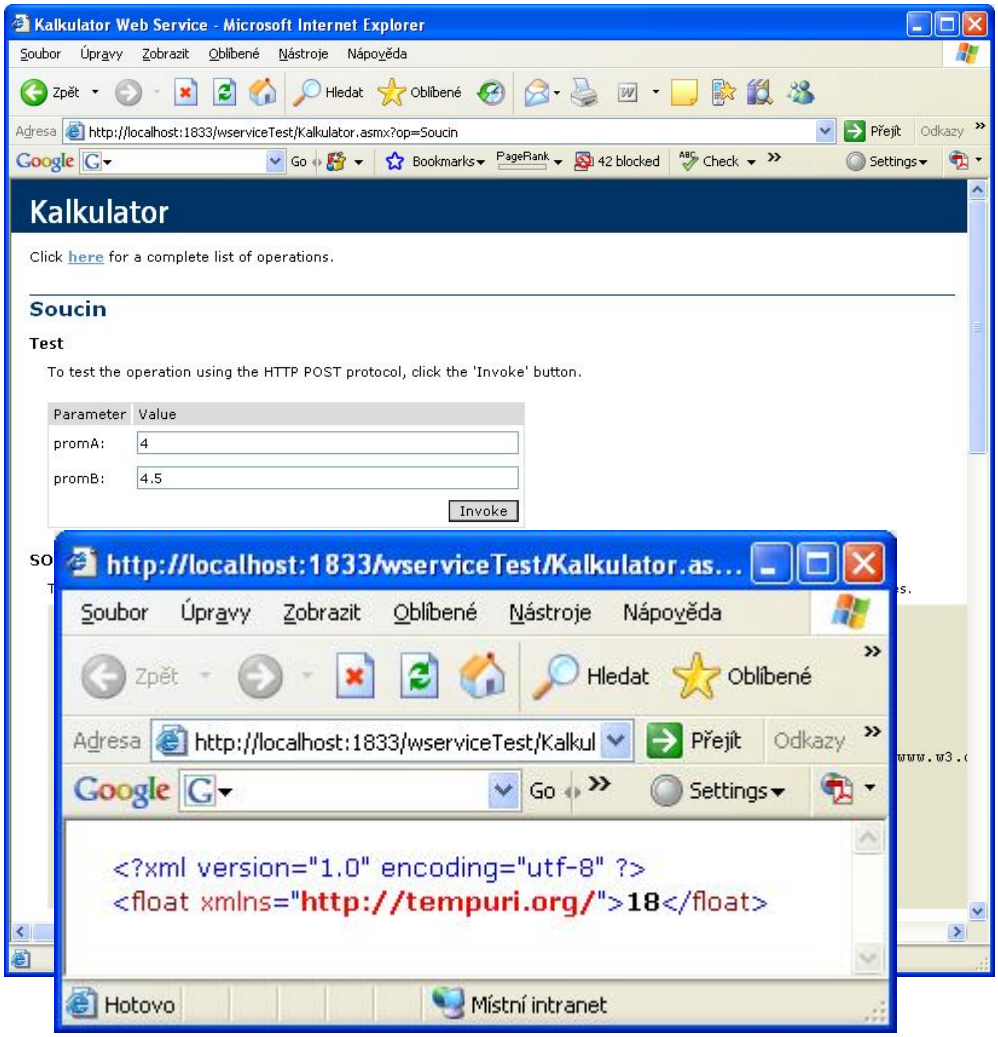
Microsoft.NET\Framework\[verze frameworku]\Config

Pokud budeme chtít z nějakého důvodu vzhled této testovací stránky změnit, zkopírujeme ji do adresáře našeho projektu. Tam ji můžete upravit dle svého uvážení a musíme k ní nastavit cestu v konfiguračním souboru web.config. To provedeme následujícím způsobem:

Příklad – Editace souboru web.config za účelem změny vzhledu testovací stránky

```
<configuration>
    <system.web>
      <webServices>
        <wsdlHelpGenerator href="MujWsdlHelpGenerator.aspx"/>
      </webServices>
     <system.web>
<configuration>
```

Kliknutím na konkrétní metodu, kterou chceme otestovat, otevřeme okno, ve kterém máme názvy parametrů web metody a příslušné *textboxy* pro naplnění hodnoty. Po odeslání tlačítkem *Invoke* získáme výsledek ve formátu XML. Celý proces je znázorněn na obrázku 11.2.



Obr. 11.2 Výsledek je vrácen ve formátu XML

11.2 Standardy webových služeb

Webové služby jsou úzce svázány s několika standardy, které používají pro svou činnost. Jedná se o standardy jazyků a protokolů, pomocí nichž jsou služby objevovány, popisovány a pomocí nichž komunikují. Ne všechny standardy uslyší člověk, který se doposud nesetkal s webovými službami

poprvé. Například XML jazyk a HTTP přenos známe téměř všichni. Mimo těchto známých pojmů definujeme další čtyři standardy:

- WSDL
  Web Service Description Language je jazyk založený na formátu XML. Popisuje doslova všechno, co je třeba, aby se klient o nabízené webové službě dověděl. Jakým způsobem se k webové službě přistupuje, jaké má metody, jaké parametry a datové typy. To vše je nezbytně nutné, abychom mohli vzdálené webové službě porozumět a efektivně využít její funkcionalitu.
- SOAP
  Protokol *Simple Object Access Protocol* je poměrně novým standardem na poli informačních technologií. Umožňuje klientským aplikacím odesílat data na server a přijímat data ze serveru. Využívá formátu XML, který má před http formátem žádosti POST a GET mnohem širší použití. Mohou se tak na místo parametrů a jejich hodnot odesílat složitější struktury dat, objekty a další. SOAP lze také používat i s jinými protokoly než http.
- DISCO
  Je to standard odvozený ze slova *Discovery* (objevování) a vytváří pouze jeden soubor s touto příponou. V tomto souboru jsou umístěné odkazy na přístupné web služby, které daná společnost uveřejňuje na svém serveru.
- UDDI
  Universal Description, Discovery and Integration je centralizovaný adresář nabízených webových služeb. Zde vývojářské firmy nabízejí své služby. Do UDDI adresáře je nutno se zaregistrovat.

Standard DISCO je podporován pouze Microsoftem a je pravděpodobné, že bude v budoucnu nahrazen ucelenějším standardem.

Standard WSDL zajišťuje samopopisnost webových služeb, přesto máme možnost webovou službu detailněji popsat. To provedeme prostřednictvím atributu *Description*, který náleží jak webové službě, tak webové metodě. Dalším atributem webové služby je jmenný prostor. ASP.NET implicitně nastavuje jmenný prostor jako `"http://tempuri.org/"`. Ten je však určen pro testování, doporučuje se nastavit vlastní jmenný prostor. Jmenné prostory jsou ve tvaru URL adresy, která je plně pod kontrolou vlastníka web služby, nemusí být však funkční.

Korespondenční úkol – Soubory v projektu s webovou službou

Prohlédněte si formát souboru s příponou disco ve vašem projektu. Prozkoumejte i další vytvořené soubory.

11.3 Proxy třída webové služby

Abychom mohli využívat webových služeb ve svých aplikacích, musíme vygenerovat přístupovou proxy třídu. Jinak bychom museli psát mnoho zdrojového kódu na nižší úrovni aplikace.

Pojem k zapamatování: Proxy třída

Proxy třída je komponenta, která umožňuje volání webové služby. Díky této přístupové třídě máme možnost zavolat metodu webové služby, jako by šlo o metodu v lokální komponentě. Třída proxy obsluhuje formát SOAP zpráv a řídí jejich přenos pomocí HTTP.

Proxy třídu můžeme vytvořit dvěma způsoby:

- Pomocí nástroje příkazového řádku a programu *wsdl.exe.*
- Pomocí webové reference ve Visual Studiu.

Oba dva způsoby umožňují prakticky stejný výsledek, rozdíly mezi nimi však jsou evidentní. Na jedné straně pomocí webové reference vše vyklikáme za několik okamžiků myší a nemusíme používat pro někoho nepopulární příkazový řádek, na straně druhé pomocí wsdl.exe vygenerujeme proxy třídu, jejíž zdrojový kód je nám k dispozici a lze ho v určitých případech měnit a i když ho měnit nebudeme, můžeme ho alespoň vidět, což v opačném případě není možné.

Než si předvedeme oba způsoby tvorby proxy třídy, uveďme v tabulce alespoň několik parametrů nástroje wsdl.exe, neboť některé z nich budeme nutně potřebovat.

Parametr	Charakteristika
URL nebo cesta	URL cesta k WSDL, XSD schématu nebo souboru .discomap.
/language:	Jazyk, ve kterém se vygeneruje proxy třída. Implicitní nastavení je C#.
/namespace:	Jmenný prostor vygenerované proxy třídy.
/out:	Výstupní cesta vygenerovaného souboru.
/protocol:	Nastavení verze protokolu.
/username: /password: /domain:	Údaje nutné pro server, jenž vyžaduje autentizaci.
/proxy:	URL proxy serveru pro HTTP požadavky. Výchozím nastavením je systémová proxy.

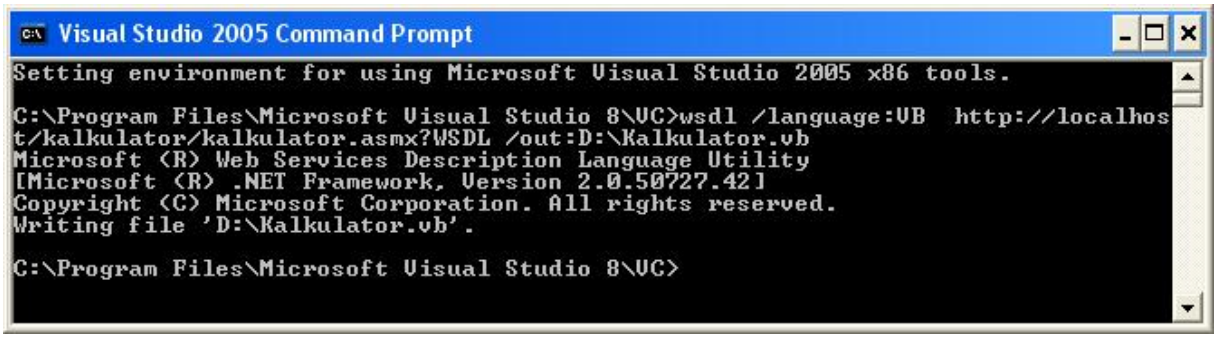
Tab. 11.1 Parametry příkazu wsdl

Nejpve tedy vytvořme proxy třídu pomocí nástroje wsdl.

Příkaz bude mít následující syntaxi:

wsdl /language:VB http://localhost/nazevprojektu/nazev.asmx?WSDL

Příkaz spustíme z *CommandPromptu* Visual Studia. Pro naši aplikaci bude název projektu *Kalkulator* a název web služby *kalkulator.asmx*. Protože v prostředí počítačových učeben nemáme práva zápisu na disk C, zvolíme ještě parametr z tabulky 11.1 pro výstupní cestu. Výsledný tvar příkazu i s reakcí *.NET Frameworku* vidíme na následujícím obrázku. Proxy třídu máme vygenerovanou na disku D v souboru *Kalkulator.vb*



Obr. 11.3 Formát příkazu wsdl.exe spuštěný z Command Promptu

Uvedený postup bude shodný i pro vzdálené web služby. Jen místo URL adresy začínající klíčovým slovem localhost, budeme mít URL vzdálené webové služby.

Nahlédnutím do zdrojového kódu proxy třídy zjistíme, že nástroj vygeneroval desítky (v obsáhlejších službách to mohou být i stovky) řádků kódu. To vše bychom museli obsloužit sami. Nyní ještě přístupovou proxy třídu zkompilujeme do komponenty, tzn. do *dll* knihovny. Tuto *dll* knihovnu umístíme do adresáře *Bin* naší klientské aplikace a můžeme využít webovou službu.

Pro uvedenou kompilaci použijeme kompilátor jazyka VB.NET opět z příkazového řádku:

vbc /t:library /out:D:\kalkulator.dll /r:system.dll /r:system.xml.dll /r:system.web.services.dll D:\Kalkulator.vb

Druhou možností je přidání web reference do projektu. V Solution Exploreru klikneme pravým tlačítkem na název projektu a zvolíme *Add Web Reference*. Otevře se okno průvodce s možností přímého zadání URL. Máme také možnost procházet všechny web služby na lokálním počítači či prohledat UDDI registr. Zvolíme naší službu Kalkulator.asmx a potvrdíme. Máme možnost vidět vytvoření složky *App_WebReferences* v projektu. Nyní můžeme pokračovat v tvorbě klientské aplikace konzumující web metody.

11.4 Vytvoření klientské aplikace ASP.NET

Předchozí odstavec nám položil stavební kámen pro vytvoření klientské aplikace. Pro rychlou tvorbu aplikace využijeme možnosti vytvoření přístupové proxy třídy vložením reference do projektu. Přidejme do projektu tedy web referenci na webovou službu *Kalkulator* nebo vytvořme přístupovou proxy třídu pomocí příkazového řádku Visual Studia. Vytvořme novou aspx stránku, která bude obsahovat formulář jednoduchého kalkulátoru využívající metod webové služby, kterou jsme si vytvořili.

Příklad – Formulář návrhu aplikace konzumující web službu

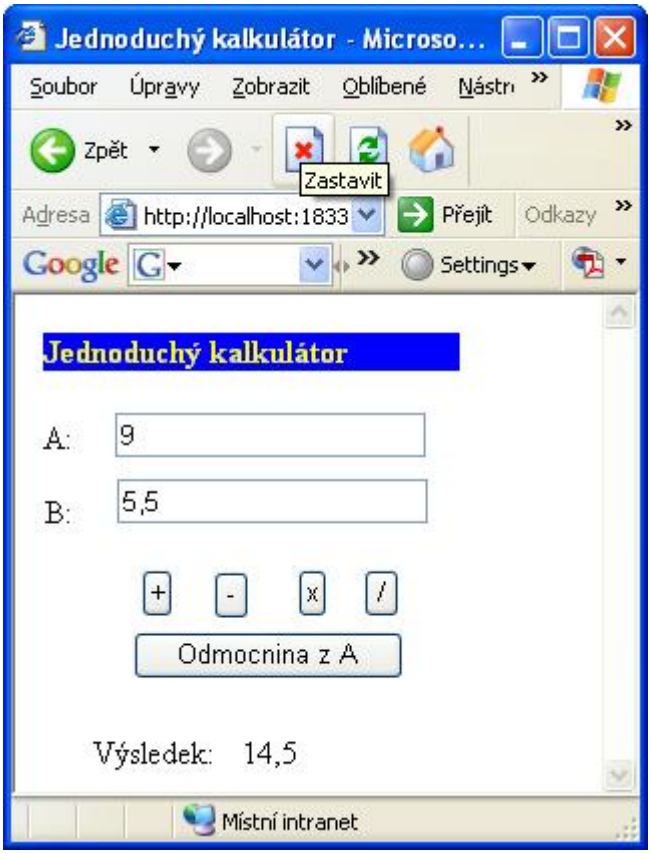
Nyní vytvořme celý skript aplikace. Bude tvořen pouze definicí objektu kalkulátoru a obsluhy událostí kliknutí pěti tlačítek. Vytvoříme tedy objekt *objCalc*, který bude zpřístupňovat metody webové služby:

Příklad – Jednoduchý skript aplikace

```
<script runat="server">
    Dim objCalc As New Kalkulator

    Sub btnNasob(ByVal obj As Object, ByVal e As EventArgs)
        lblVysledek.Text = objCalc.Soucin(tbA.Text, tbB.Text)
    End Sub

    Sub btnVydel(ByVal obj As Object, ByVal e As EventArgs)
        lblVysledek.Text = objCalc.Podil(tbA.Text, tbB.Text)
    End Sub
    Sub btnSecti(ByVal obj As Object, ByVal e As EventArgs)
        lblVysledek.Text = objCalc.Soucet(tbA.Text, tbB.Text)
```

```
        End Sub
        Sub btnOdecti(ByVal obj As Object, ByVal e As EventArgs)
            lblVysledek.Text = objCalc.Rozdil(tbA.Text, tbB.Text)
        End Sub
        Sub btnOdmocni(ByVal obj As Object, ByVal e As EventArgs)
            lblVysledek.Text = objCalc.Odmocnina(tbA.Text)
        End Sub
    </script>
```

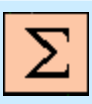
Jednoduché že? Aplikace je samozřejmě triviální, ale svůj účel splnila. Vytvořili jsme klientskou aplikaci, která konzumuje metody webové služby, aniž by o tom měl uživatel ponětí. Webová služba je na stejném počítači (localhost) jako aplikace. V další kapitole si již ukážeme konzumaci vzdálených webových služeb.



Obr. 11.4 klientská aspx stránka využívající webovou službu

Ke kapitole 11 je přiřazena demonstrační animace č. 10

Animace č. 10 obsahuje vývoj vlastní webové služby od počátku a tvorbu klientské aplikace, která webovou službu využívá. Animace bude dále obsahovat příklady z následující kapitoly.



Shrnutí kapitoly

Webové služby umožňují sdílení komponent a jejich funkcí. Je to v podstatě nejlépe demonstrovaná možnost znovupoužití kódu. Nespornou výhodou je údržba vaší aplikace. Nemusíte již instalovat řady programů a aplikačních objektů. Jednoduše je vzdáleně

využíváte.

Webová služba je novým mechanismem pro výměnu dat mezi aplikacemi. Pomocí XML a dalších technologií umožňuje komunikaci aplikace s jinou aplikací, která ji konzumuje vzdáleně bez ohledu na tom, na jakém hardwaru a softwaru je implementována.

Webovou službu máme možnost ve Visual Studiu ihned otestovat bez tvorby klientské aplikace. Klikneme pravým tlačítkem na soubor se službou s příponou *asmx* a nastavíme ji jako startovací stránku, popřípadě ihned zvolíme možnost *View in Browser*. Webová služba se spustí v prohlížeči s připraveným prostředím, ve kterém vidíme jednotlivé metody, které můžeme otestovat. Kliknutím na konkrétní metodu, kterou chceme otestovat otevřeme okno, ve kterém máme názvy parametrů web metody a příslušné textboxy pro naplnění hodnoty. Po odeslání tlačítkem Invoke získáme výsledek ve formátu XML.

Webové služby jsou úzce svázány s několika standardy, které používají pro svou činnost. Jedná se o standardy jazyků a protokolů pomocí nichž jsou služby objevovány, popisovány a pomocí nichž komunikují:

- WSDL
  Web Service Description Language je jazyk založený na formátu XML. Popisuje doslova všechno, co je třeba, aby se klient o nabízené webové službě dověděl. Jakým způsobem se k webové službě přistupuje, jaké má metody, jaké parametry a datové typy. To vše je nezbytně nutné, abychom mohli vzdálené webové službě porozumět a efektivně využít její funkcionalitu.
- SOAP
  Protokol *Simple Object Access Protocol* je poměrně novým standardem na poli informačních technologií. Umožňuje klientským aplikacím odesílat data na server a přijímat data ze serveru. Využívá formátu XML, který má před http formátem žádosti POST a GET mnohem širší použití. Mohou se tak na místo parametrů a jejich hodnot odesílat složitější struktury dat, objekty a další. SOAP lze také používat i s jinými protokoly než http.
- DISCO
  Je to standard odvozený ze slova *Discovery* (objevování) a vytváří pouze jeden soubor s touto příponou. V tomto souboru jsou umístěné odkazy na přístupné web služby, které daná společnost uveřejňuje na svém serveru.
- UDDI
  Universal Description, Discovery and Integration je centralizovaný adresář nabízených webových služeb.

Standard WSDL zajišťuje samopopisnost webových služeb, přesto máme možnost webovou službu detailněji popsat. To provedeme prostřednictvím atributu Description, který náleží jak webové službě, tak webové metodě. Dalším atributem webové služby je jmenný prostor. ASP.NET implicitně nastavuje jmenný prostor jako `"http://tempuri.org/"`. Ten je však určen pro testování, doporučuje se nastavit vlastní jmenný prostor.

Abychom mohli využívat webových služeb ve svých aplikacích, musíme vygenerovat přístupovou proxy třídu. Jinak bychom museli psát mnoho zdrojového kódu na nižší úrovni aplikace. Proxy třída je komponenta, která umožňuje volání webové služby. Díky této přístupové třídě máme možnost zavolat metodu webové služby, jako by šlo o metodu v lokální komponentě. Třída proxy obsluhuje formát SOAP zpráv a řídí jejich přenos pomocí HTTP. Proxy třídu můžeme vytvořit dvěma způsoby:

- Pomocí nástroje příkazového řádku a programu *wsdl.exe.*
- Pomocí webové reference ve Visual Studiu.

Jakmile máme vytvořenou proxy třídu, můžeme jednoduchým a známým způsobem postupovat při tvorbě klientské aspx stránky.

Úkol k řešení 11.1 – Vytvoření web služby

Vytvořte webovou službu poskytující kořeny kvadratické rovnice.

Úkol k řešení 11.2 – Vytvoření web služby

Vytvořte webovou službu poskytující převody mezi stupni Celsia a Fahrenheita.

Úkol k řešení 11.3 – Vytvoření proxy třídy

Vytvořte přístupové proxy třídy pro obě web služby z předchozích příkladů.

Úkol k řešení 11.4 – Vytvoření klientské aplikace

Vytvořte klientskou aplikaci, která konzumuje webové služby z předchozích příkladů.

Kontrolní otázka 11.1

Co je to webová služba a jaký je její význam?

Kontrolní otázka 11.2

Jaké standardy využívají webové služby?

Kontrolní otázka 11.3

Co je to přístupová proxy třída a jak ji vytváříme?

Kontrolní otázka 11.4

Jakým způsobem se vyvíjí klientská aplikace ASP.NET konzumující webovou službu?

Kontrolní otázka 11.5

Jakým způsobem se webová služba dokumentuje?

12. PŘÍKLADY VYUŽITÍ WEBOVÉ SLUŽBY

Čas ke studiu: 2 hodiny

Cíl Po prostudování této kapitoly budete umět

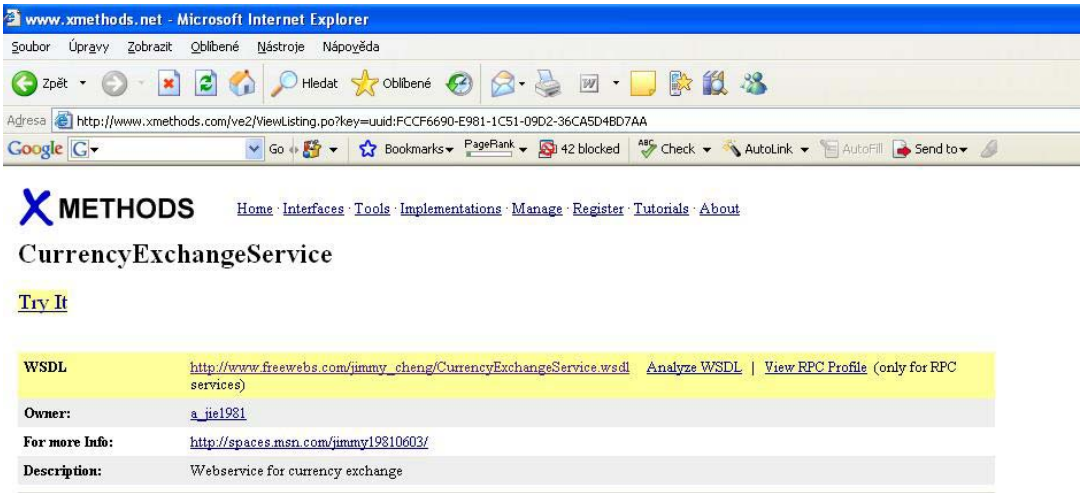
- hledat přístupné webové služby
- analyzovat wsdl přístupných web služeb
- vkládat referenci na web služby do projektu Visual Studia
- používat IntelliSense k tvorbě instancí objektů webové služby
- zjišťovat, které metody web služby a s jakými parametry můžete použít
- konzumovat webové služby ve svých aplikacích

Výklad

Navážeme na předchozí kapitolu, kde jsme k naší vyvinuté webové službě vytvořili klientskou aplikaci. Nyní využijeme služeb, které naprogramovali jiní autoři a dali je k dispozici pro testování široké veřejnosti. Řada těchto služeb je velice užitečných. V našich příkladech tak budeme konzumovat webové služby do svých aplikací, které budou zobrazovat např. převody měn jednotlivých zemí, popřípadě počasí v různých světových lokalitách.

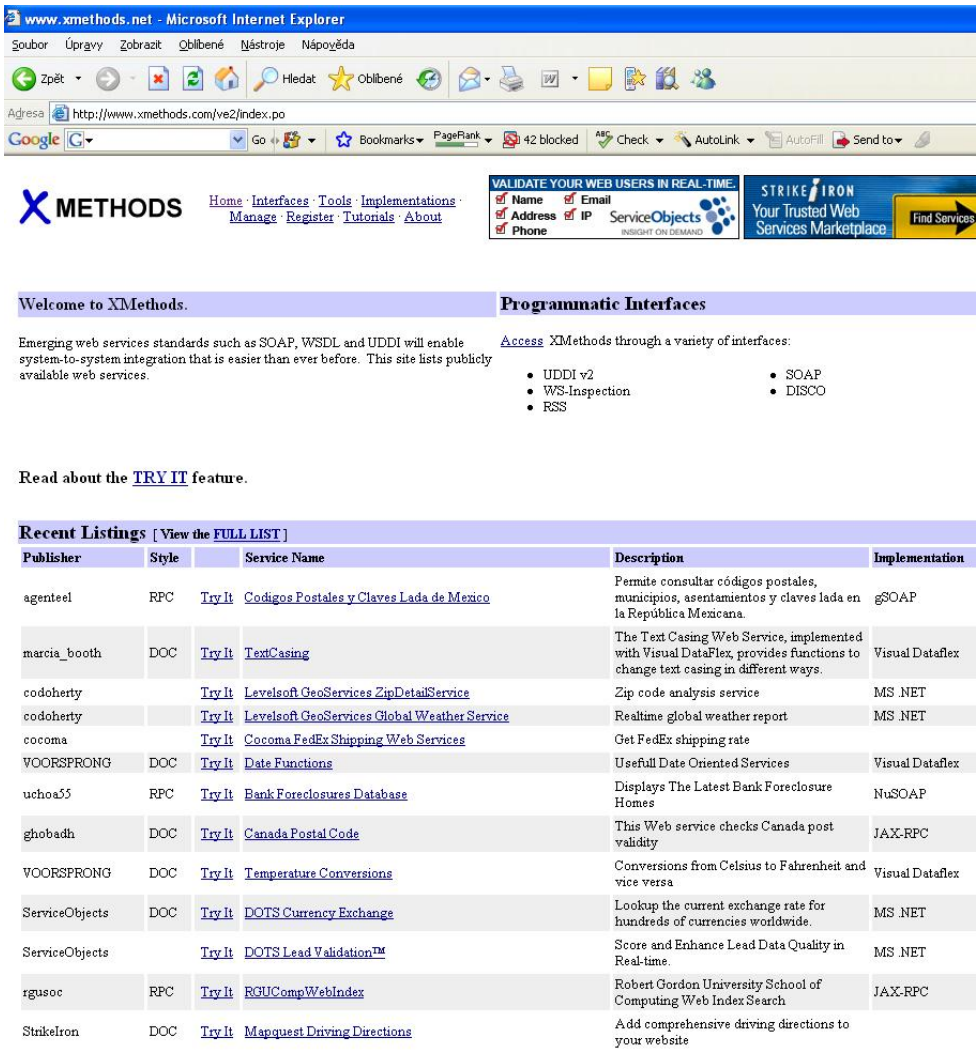
12.1 Webové služby dostupné na internetu

Na internetu lze nalézt mnoho webových služeb různých autorů, které jsou víceméně použitelné v různých aplikacích. Víceméně proto, že převážná většina serverů, kde se shlukují webové služby je amerických a podle toho odpovídá i jejich nabídka. A tak určitě nevyužijeme službu pro generování poštovních směrovacích čísel v určitých amerických státech, ale lze najít velice užitečné a použitelné služby jako například převod měn, kontrola existence e-mailové adresy, počasí a další.



Obr. 12.1 Web služba pro převod měn

Jedním z těchto serverů je www.xmethods.com. Zde najdeme rozsáhlý seznam veřejně dostupných služeb. Kliknete-li na tačítko *Full List*, zobrazí se vám jejich kompletní výpis. Samozřejmě si web služby prezentované na tomto serveru můžete vyzkoušet, použijete-li link *Try It*.



Obr. 12.2 Stránka Xmethods s volnými web službami, které budeme využívat

Právě tuto akci jsme provedli, když jsme našli službu pro převod měn *CurrencyExchangeService*. Na obrázku 12.1 vidíme stránku, kde je zobrazen její wsdl, který lze analyzovat a vypsat tak dostupné metody této služby. Službu tedy otestujeme.

Korespondenční úkol - Dostupné webové služby

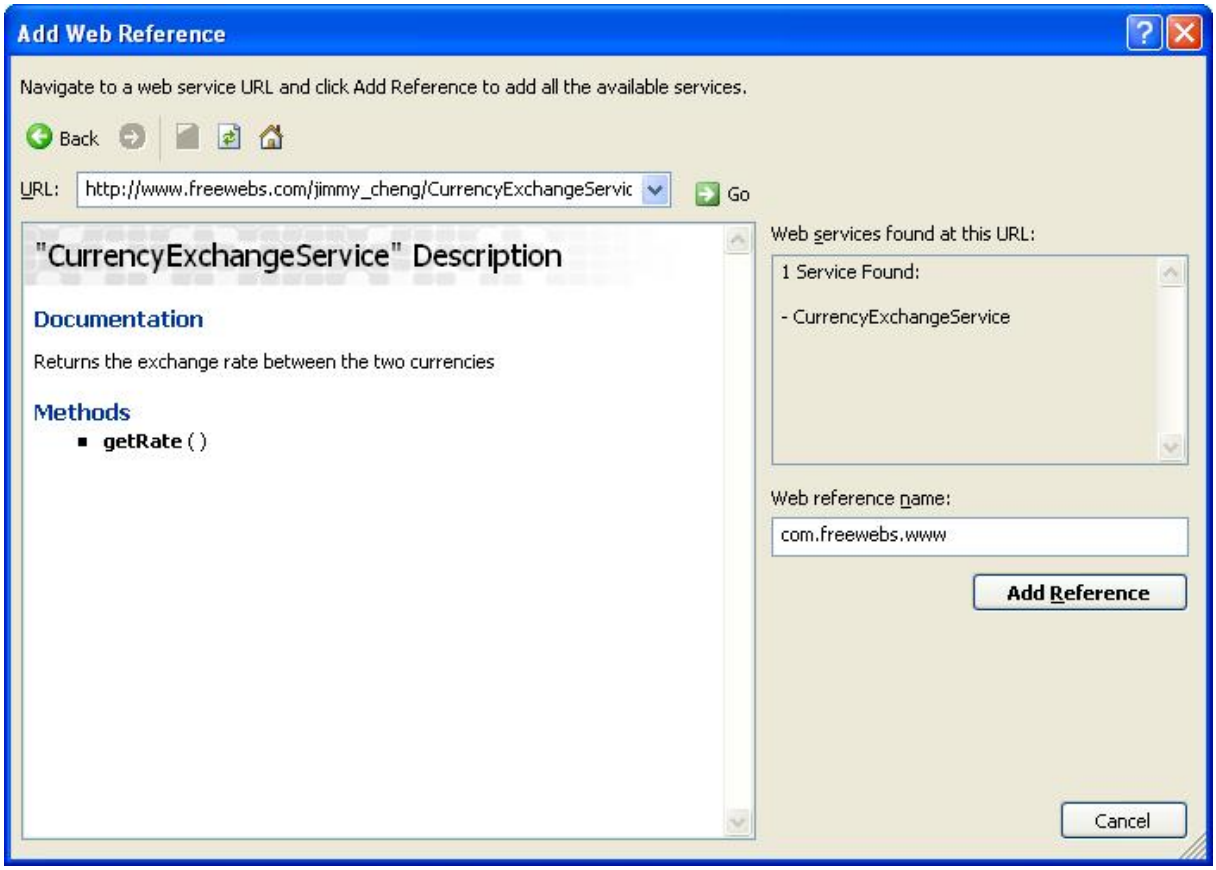
Vyhledejte na tomto serveru ale i jiných serverech různé webové služby a vyzkoušejte je.

12.2 Tvorba aplikací s dostupnými webovými službami

V tomto odstavci si ukážeme, jakým způsobem můžeme vyhledanou webovou službu konzumovat v naší klientské aplikaci. Provedeme to pomocí dostupných nástrojů Visual Studia a ukážeme si jak je to pohodlné.

Našim úkolem je tedy vytvořit aplikaci pro převod měn mezi dvěma zeměmi a využívat přitom

metodu webové služby. Pokud máme na serveru stále otevřenou web službu *CurrencyExchangeService*, zkopírujeme si do schránky její wsdl URL. V Solution Exploreru klikneme pravým tlačítkem na název otevřeného projektu a zvolíme položku *Add Web Reference*.



Obr. 12.3 URL web služby přidáme do projektu v Solution Exploreru jako web referenci

Otevře se průvodce, který umožňuje přidat referenci na webovou službu do projektu. Do pole URL adresy vložíme URL wsdl webové služby, které máme ve schránce. Potvrdíme tlačítkem *Go*.

V textové oblasti na pravé straně vidíme výsledek této akce. Na dané URL adrese byla nalezena jedna webová služba s názvem *CurrencyExchangeService*. V textboxu níže si můžeme všimnout webové reference v obrácené notaci (www je až na konci). V levém okně průvodce vidíme popis nalezené služby a zobrazenou metodu *getRate()*, kterou bychom na webu *xmethods* viděli, pokud bychom klikli na položku *Analyze WSDL*.

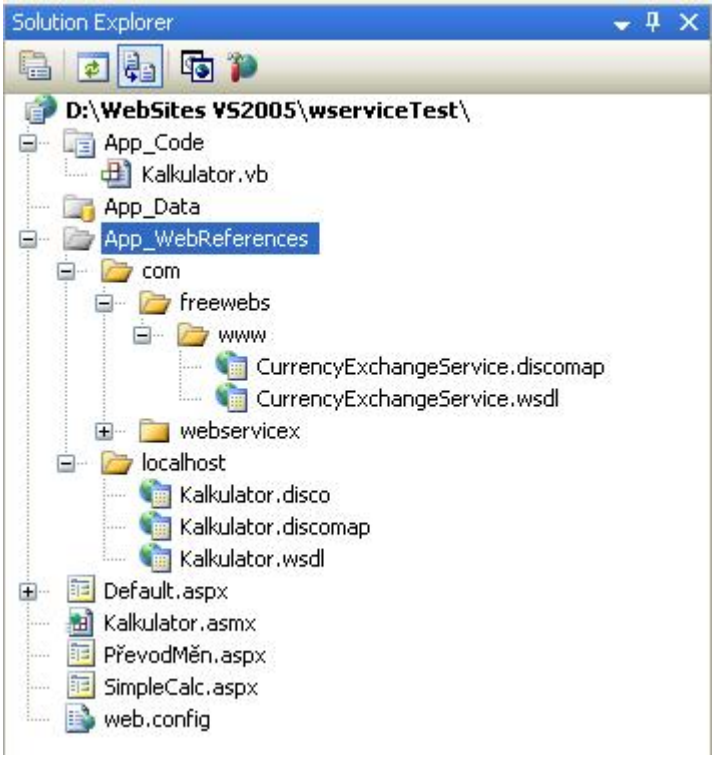
Posledním krokem tohoto průvodce bude kliknout na tlačítko *Add Reference* a tímto se přidá odkaz do projektu a průvodce se uzavře. Celé okno průvodce máme zobrazeno na obrázku 12.3.

Podívejme se zpět do *Solution Exploreru*. Ve složce *App_WebReferences* již nemáme pouze složku *localhost*, která obsahuje referenci na službu kalkulátoru z minulé kapitoly, ale také hierarchickou strukturu *com – freewebs – www* s odkazem na službu *CurrencyExchangeService*.

Současná podoba projektu v Solution Exploreru je na obrázku 12.4. Můžeme si všimnout, že kromě tohoto odkazu ve složce nalezneme ještě složku *webservicex*, která obsahuje web referenci na službu, kterou již máme připravenou pro druhý příklad této kapitoly.

Pokračujeme tedy dále a vytvoříme klientskou aplikaci neboli aspx stránku s formulářem pro převod měn mezi dvěma zeměmi. Protože jsme již na konci našeho kurzu, nebudeme si popisovat zdrojový kód formuláře, postačí když jej ukážeme v návrhovém módu stránky.

Aplikace bude obsahovat dvě tlačítka, první pro převod měn mezi jednotlivými zeměmi a druhým tlačítkem bude častá operace převodu měny některé země na Euro. *Textboxy* nám postačí dva a ještě definujme *label* pro zobrazení výsledku.

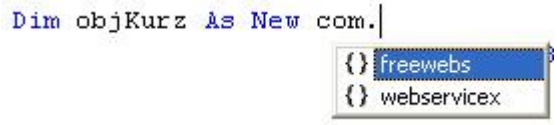


Obr. 12.4 Hierarchická struktura cesty k web referenci

Příklad – Formulář klientské aplikace konzumující web službu

Nyní napišme zdrojový kód obsluhy dvou tlačítek. Přesto všechno, jakou funkcionalitu nám aplikace nabízí, její zdrojový kód nám již zabere pouze pár řádků a to ještě za pomocí *IntelliSense*.

Nadefinujeme si nový objekt jako instanci objektu *CurrencyExchangeService*. Pojmenujme jej třeba *objKurz*. Podle hierarchické struktury web reference vidíme v Solution Eploreru, že nejprve přichází řada na *com*. Protože tečková konvence zůstává zachována vložíme tečku a vidíme, že IntelliSense ihned zareagovalo.

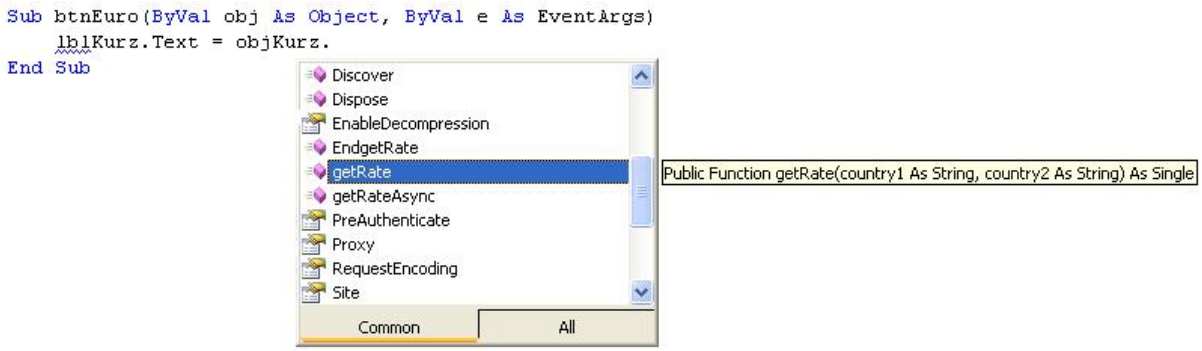


Obr. 12.5 Vytvoření nového objektu za pomocí IntelliSense

Byly nám nabídnuty dvě možnosti (jak bylo uvedeno výše v projektu jsou již dvě web reference). Zvolíme tu správnou a IntelliSense pokračuje dále až k nabídce webové služby *CurrencyExchange Service*.

Obr. 12.6 Pomocí Intelisense se tečkovou konvencí dostaneme až k názvu web služby

Objekt *objKurz* máme nadefinován, můžeme přejít na obsluhu stisku tlačítek. Pokud tedy stiskneme tlačítko převodu měny na Euro, chceme do výsledného *labelu* zobrazit výsledek metody *getRate()* objektu *objKurz*. Při použití *objKurz* zapracuje IntelliSense a metodu nám nabídne. V nápovědném žlutém rámečku se zobrazí parametry metody se svými datovými typy.



Obr. 12.7 Intellisense nám dokonce nabídne i metodu web služby

Vidíme že parametry jsou řetězce, tudíž pro převod měn mezi oběma zeměmi volíme vstupní data obou textboxů a pro převod země na Euro bude prvním parametrem právě „Euro“ v uvozovkách.

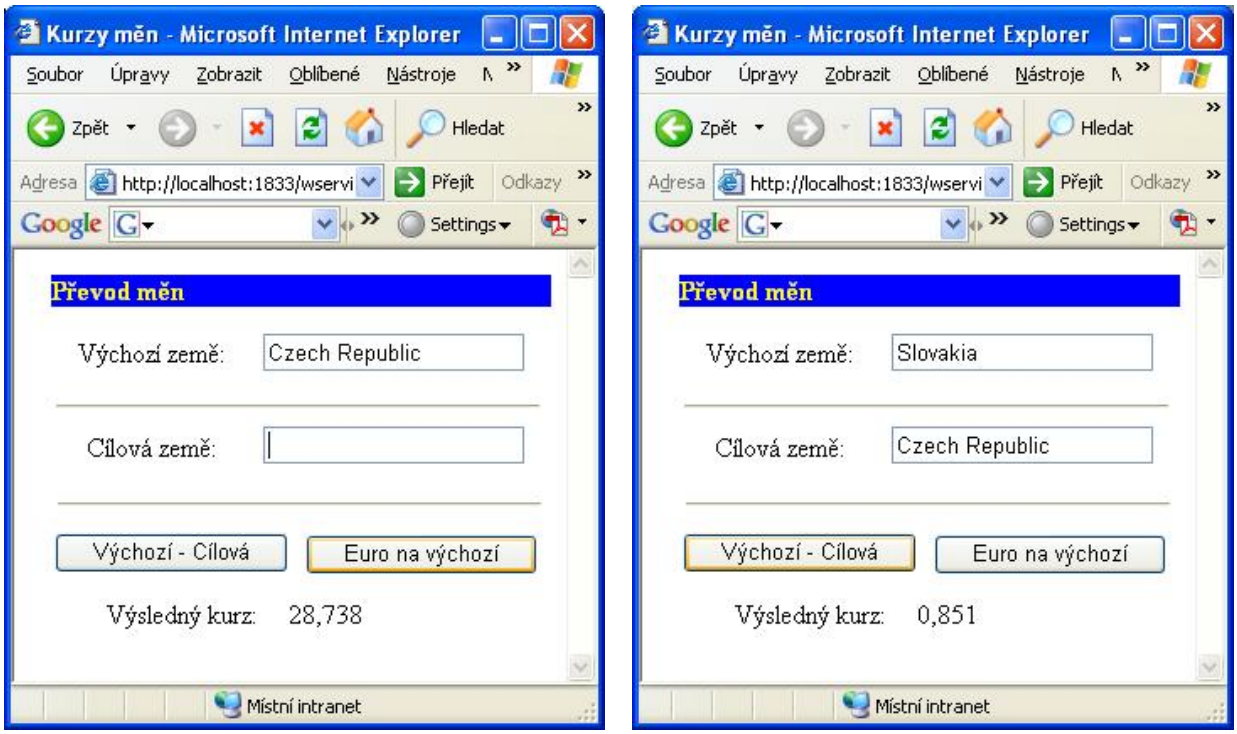
Výsledný skript má opravdu pouze pár řádků:

Příklad – Skript aplikace převodu měn

```
<script runat="server">
    Dim objKurz As New com.freewebs.www.CurrencyExchangeService
    Sub btnKurz(ByVal obj As Object, ByVal e As EventArgs)
        lblKurz.Text = objKurz.getRate(tbVychozi.Text,
tbCilova.Text)
    End Sub

    Sub btnEuro(ByVal obj As Object, ByVal e As EventArgs)
        lblKurz.Text = objKurz.getRate("Euro", tbVychozi.Text)
    End Sub
</script>
```

A jsme hotovi. Spustíme aplikaci a vyzkoušíme převod měn mezi různými zeměmi a také převod měn mezi zeměmi a Eurem. Na obrázku 12.8 je zobrazena aplikace, která využila kurzovního lístku v červnu 2007.

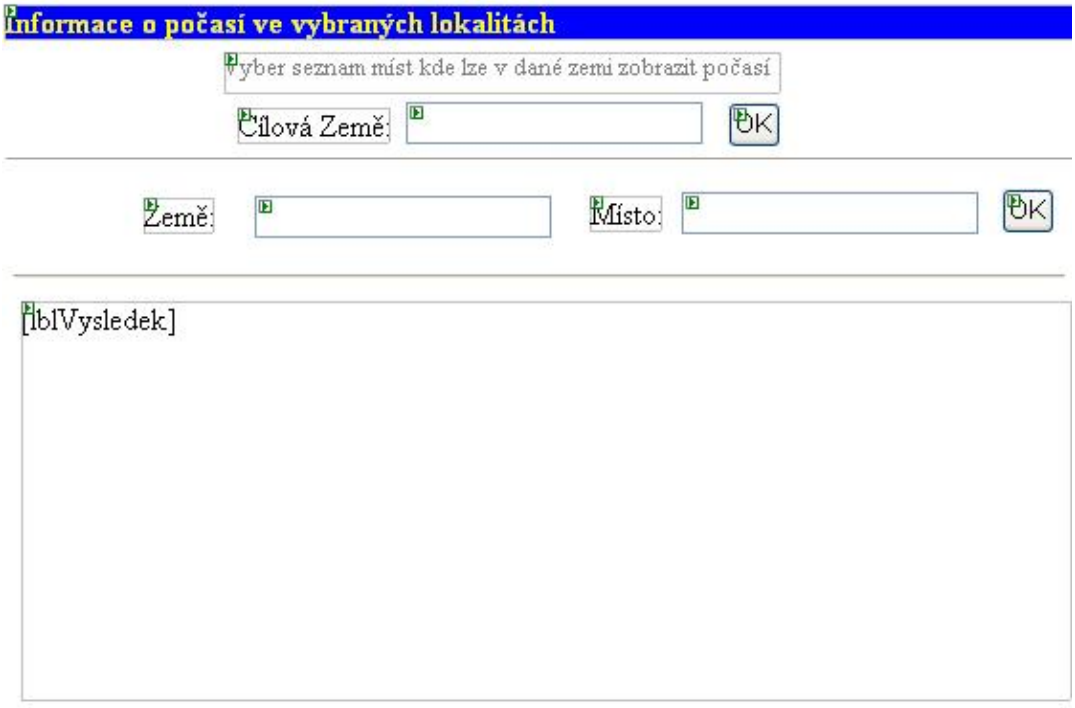
Druhý příklad konzumace webové služby bude obdobný. Stejným postupem vložíme do projektu web referenci. Jedná se o službu *GlobalWeather*, stále na sevru *xmethods*.



Obr. 12.8 Výsledná aplikace převodu měn

Webová služba poskytuje informace o počasí ve vybraných světových lokalitách. Abychom zjistili, jaké počasí je v konkrétním světovém městě, musíme zjistit, zda-li je v něm dostupná meteo stanice. Proto webová služba poskytuje dvě metody *GetWeather()* pro zjištění počasí v konkrétním městě a *GetCitiesByCountry()* pro zobrazení měst v daném státě, kde lze získat informace o počasí. Formulář tedy přizpůsobíme uvedeným skutečnostem.

Příklad – Formulář k aplikaci využívající web službu počasí

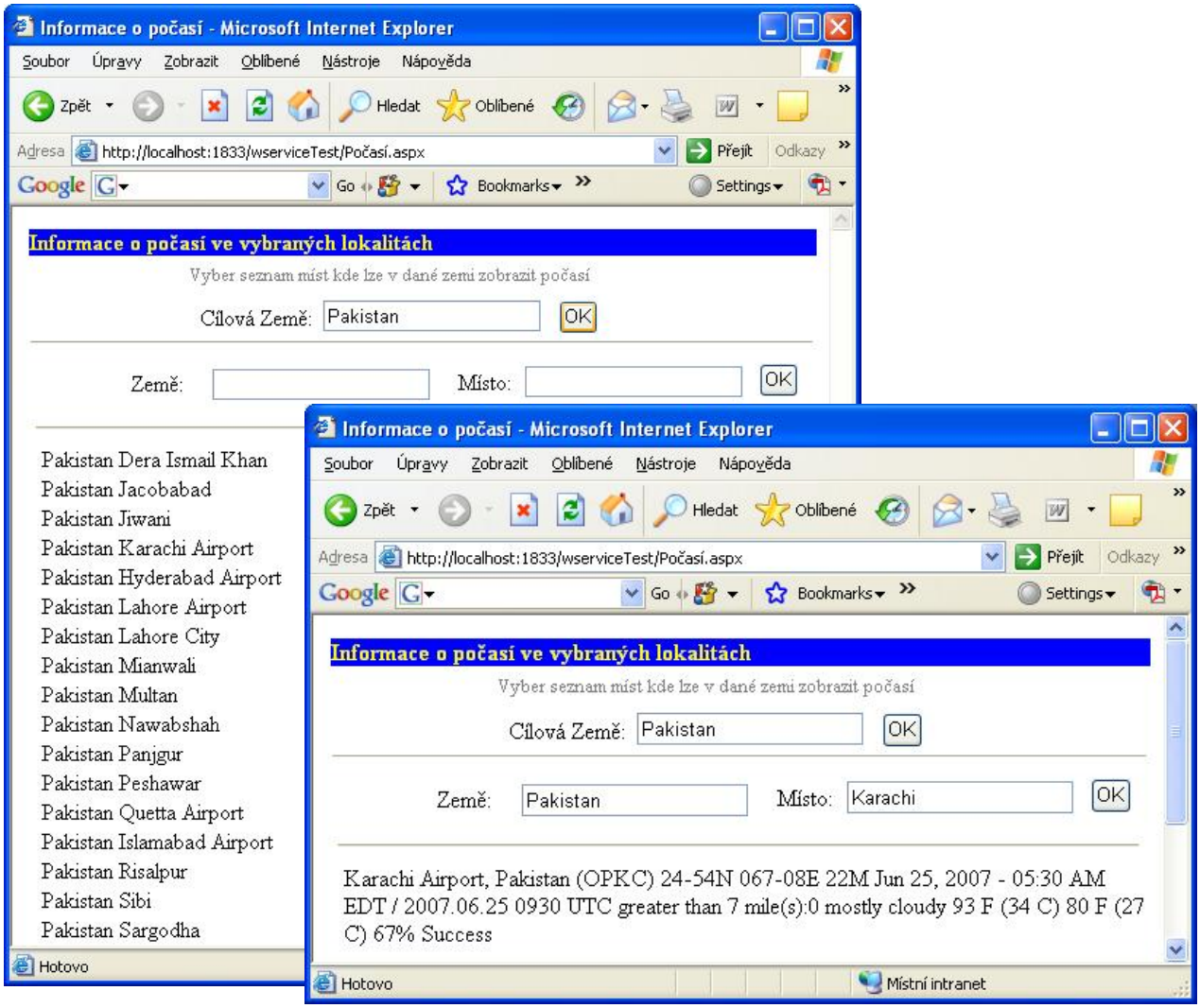


Webovou referenci jsme vložili stejným způsobem, obdobně jako v předchozím příkladu budeme pokračovat i dále. Vytvoříme objekt *objPocasi* jako instanci *GlobalWeather*. Poté opět pomocí IntelliSense zajistíme, abychom na vložená tlačítka volali správné metody.

Příklad – Skript aplikace využívající web službu počasí

```
<script runat="server">
    Dim objPocasi As New com.webservicex.www.GlobalWeather
    Sub btnVratMesta(ByVal obj As Object, ByVal e As EventArgs)
        lblVysledek.Text = objPocasi.GetCitiesByCountry(tbZeme.Text)
    End Sub

    Sub btnVratPocasi(ByVal obj As Object, ByVal e As EventArgs)
        lblVysledek.Text = objPocasi.GetWeather(tbMesto.Text,
tbStat.Text)
    End Sub
```

Nejprve tedy zjistíme, která města v cílové zemi poskytují informaci o počasí, tato metoda má pouze jeden parametr a poté již můžeme zavolat druhou metodu, se dvěma parametry: země, město. Výsledné chování aplikace pro Pákistán a následnou volbu města je zobrazeno na obrázku 12.9.



Obr. 12.9 Dvě okna výsledné aplikace pro zjištění počasí ve vybrané lokalitě

Ke kapitole 12 je přiřazena demonstrační animace č. 10

Animace č. 10 obsahuje kromě příkladů z minulé kapitoly také tvorbu dvou aplikací z kapitoly 12.

Shrnutí kapitoly

Na internetu lze nalézt mnoho webových služeb různých autorů, které jsou víceméně použitelné v různých aplikacích. Uvedené web služby můžeme poměrně snadno konzumovat do svých aplikací.

Nejprve v projektu přidáme referenci. V Solution Exploreru klikneme pravým tlačítkem na název otevřeného projektu a zvolíme položku *Add Web Reference*.

Otevře se průvodce, který umožňuje přidat referenci na webovou službu do projektu. Do pole URL adresy vložíme URL wsdl webové služby, kterou chceme použít. Potvrdíme tlačítkem *Go*.

V textové oblasti na pravé straně vidíme výsledek této akce. Na dané URL adrese byla nalezena webová služba a zobrazen její název. V textboxu níže si můžeme všimnout webové reference v obrácené notaci (www je až na konci). V levém okně průvodce vidíme popis nalezené služby a zobrazené metody, které služba poskytuje.

Posledním krokem tohoto průvodce bude kliknout na tlačítko Add Reference a tímto se přidá odkaz do projektu a průvodce se uzavře. Nyní máme v *Solution Exploreru* ve složce App_WebReferences složku s hierarchickou strukturou cesty např. *com – freewebs – www* s odkazem na webovou službu popsanou souborem wsdl.

Pokračujeme dále a vytvoříme klientskou aplikaci neboli aspx stránku s formulářem pro konkrétní úlohu. Poté vytvoříme obslužný skript, kde nadefinujeme instanci objektu webové služby a vhodným způsobem konzumujeme potřebné metody.

Úkol k řešení 12.1 – Vytvoření aplikace k webové službě

Vyhledejte web službu k získání informace, zda-li zadaná e-mailová adresa existuje a vytvořte k ní klientskou aplikaci.

Úkol k řešení 12.2 – Vytvoření aplikace k webové službě

Vyhledejte nějakou užitečnou webovou službu obsahující více metod a vytvořte k ní klientskou aplikaci.

Úkol k řešení 12.3 – Vytvoření proxy třídy

Vyzkoušejte možnost vytvořit přístupovou proxy třídu pro vzdálené webové služby prostřednictvím nástroje *wsdl.exe*.

Úkol k řešení 12.4 – Úprava aplikace

Vytvořenou aplikaci v této kapitole upravte tak, aby město, ve kterém chce uživatel zjistit počasí, vybral z nějakého ovládacího prvku pro seznamy.

Kontrolní otázka 12.1

Jakým způsobem vložíme odkaz na webovou službu do projektu?

Kontrolní otázka 12.2

Jakým způsobem využijeme IntelliSense při tvorbě klientské aplikace konzumující web službu?

Kontrolní otázka 12.3

Jaké URL zadáváme v původci vložení web reference?

Kontrolní otázka 12.4

Uveďte typický postup při konzumaci web služby v klientské aplikaci.

Kontrolní otázka 12.5

Co znamenají zkratky WSDL, Disco, UDDI, SOAP?

13. CO SE DO UČEBNÍHO TEXTU NEVEŠLO

Čas ke studiu: libovolný čas ke zdokonalování samostudiem

Cíl Po prostudování této kapitoly budete umět

- definovat všechna důležitá témata pro práci v ASP.NET
- ohodnotit své znalosti technologie ASP.NET

Výklad

Tato kapitola se původně měla jmenovat závěr. Ale co bychom vlastně uzavírali? Pouze učební text jednosemestrálního kurzu věnovaného technologii ASP.NET. Všichni si uvědomujeme, že 14-ti týdení semestr se dvěma hodinami cvičení týdně z nás v žádném případě vývojáře neudělá. Může nám ale ukázat cestu, motivovat nás v další práci a touze po zdokonalení a zisku dalších vědomostí.

13.1 Vždy se máme co učit

Na závěr učebního textu máte uveden seznam literatury, který vám může pomoci k zisku dalších teoretických a praktických zkušeností. Jako autor tohoto učebního textu vám je mohu doporučit, neboť jsem je četl všechny. Ano opravdu, pokud chcete porozumět určité technologii, a neplatí to jen v programování, musíte být připraveni studovat odbornou literaturu. Tato literatura se nečte jako román od začátku do konce, jednoduše se zajímáte o aktuální témata, víte kde vás bota tlačí a tak se věnujete konkrétním kapitolám, až zjistíte, že některé kapitoly čtete denně a do jiných jste jen párkrát nakoukli.

Tento předmět je limitován svým rozsahem 14 cvičení a tomu také odpovídá délka učebního textu. Ale věžte, že i přes tento menší rozsah jsme probrali více než polovinu témat, kterým se věnují tisícistránkové knihy! Není účelem vše detailně popsat, ale ukázat vám cestu a procvičit dané téma na vhodných příkladech. Zbytek už je na vás, záleží jak moc chcete danému tématu porozumět. Svou roli hraje volba vašeho předmětu. Zdali jsem fanouškem vývoje web stránek a rád si nějaké vytvořím pro demonstraci např. svých koníčků na internetu, či zdali chci programovat profesionálně nebo vést nějaký tým, který se danou technologií zabývá, takže by bylo vhodné abych ji také uměl, a nebo jsem si tento předmět zvolil z čisté zvědavosti či jsem neměl jinou možnost kvůli počtu kreditů. Tato volba je jen na vás a mým úkolem bylo zpřístupnit vám danou problematiku a udělat ten první krok.

Jak jsem již naznačil, k úplnému popisu technologie ASP.NET bych potřeboval ještě jeden semestr, a tak jsem se snažil vybrat vhodná témata, která považuji za důležitá. Kdyby se mi naskytla šance zvětšit rozsah předmětu, témata, která by vás jistě zajímala, jsou následující. Říkejme tomu pracovně Programování aplikací pro Internet III:

1. Tvorba aplikace s použitím techniky Master Page.
2. Práce se soubory v ASP.NET – upload souborů na server.
3. Bezpečnostní prvky aplikace – Autorizace, autentizace.
4. Ovládací prvky typu Login pro bezpečnost aplikace.
5. Členství.

6. Mapa Webu – Web Site Map.
7. Tvorba webových portálů – Web Parts a Web Zones.
8. Tvorba vlastních serverových prvků a jejich umístění do Toolboxu.
9. Práce s XML.
10. Cacheování.
11. Konfigurace ASP.NET aplikací.
12. Umístění web aplikace na server.

13.2 Zadání zápočtového projektu

Pro zisk zápočtu z předmětu Programování aplikací pro Internet II je zapotřebí následující:

1. Aktivní účast na cvičeních.
2. Naprogramování vlastní web aplikace využívající práci s databázemi a webovou službu.
3. Nastudování některého tématu uvedeného v kapitole 13.1, který bude demonstrován na vhodném příkladu.

Jako garant předmětu Programování aplikací pro Internet II vám přeji hodně úspěchů ve studiu, ať vás zvolený předmět baví a předá vám užitečné znalosti, které vám obohatí váš profesní či osobní život.

Marek Babiuch

Další zdroje

ASP.NET 2.0

Programujeme profesionálně

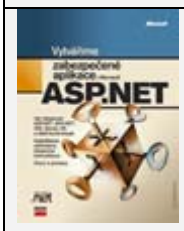
Bill Evjen, Scott Hanselman, Farhan Muhammad, Srinivasa Sivakumar, Devin Rader

1224 stran, rok vydání 2007
ISBN: 978-80-251-1473-5

ASP.NETa ADO.NET 2.0 - Hotová řešení

Luboslav Lacko

388 stran, rok vydání 2006
ISBN: 80-251-1028-1



Vytváříme zabezpečené aplikace v Microsoft ASP.NET

544 stran, rok vydání 2004,
ISBN: 80-251-0466-4

Programujeme mobilní aplikace - ve Visual Studiu .NET

Luboslav Lacko

480 stran, rok vydání 2004,
ISBN: 80-251-0176-2

Microsoft ASP.NET Konfigurace a nastavení

Kapesní rádce programátora a administrátora

James Avery

216 stran , rok vydání 2004,
ISBN: 80-251-0121-5

Programování v Microsoft .NET

Webové aplikace v C#, ASP.NET a .NET Framework

Jeff Prosise

736 stran, rok vydání 2003,
ISBN: 80-7226-879-1

Naučte se ADO.NET za 21 dní

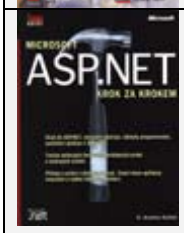
Dan Fox

528 stran, rok vydání 2002,
ISBN: 80-7226-772-8

Naučte se ASP.NET za 21 dní

Chris Payne

786 stran, rok vydání 2003,
ISBN: 80-7226-605-5



MS ASP.NET

Krok za krokem

G. Andrew Duthie

511 stran, rok vydání 2002,
ISBN: 8086593339

Developing More-Secure Microsoft ASP.NET 2.0 Applications

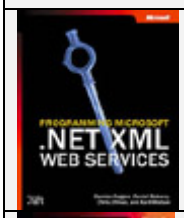
Dominick Baier

rok vydání 2006,
ISBN: 0-7356-2331-7

Programming Microsoft ASP.NET 2.0 Applications: Advanced Topics

Dino Esposito (Solid Quality Learning)

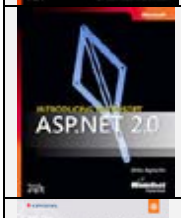
rok vydání 2006,
ISBN: 0-7356-2177-2



Programming Microsoft .NET XML Web Services

Damien Foggon, Daniel Maharry, Chris Ullman, and Karli Watson

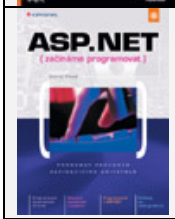
720 stran, rok vydání 2006,
ISBN: 0-7356-1912-3



Introducing Microsoft ASP.NET 2.0

Dino Esposito (Wintellect)

448 stran, rok vydání 2006,
ISBN: 0-7356-2024-5

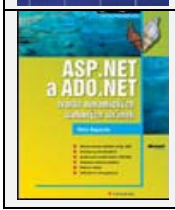


ASP.NET

začínáme programovat

Slavoj Písek

228 stran, rok vydání 2003,
ISBN: 80-247-0526-5



ASP.NET a ADO.NET

tvorba dynamických webových stránek

Dino Esposito

352 stran, rok vydání 2003,
ISBN: 80-247-0474-9

Klíč k řešení – kapitola 1

Odpověď na kontrolní otázku 1.1

Microsoft .NET Framework je platforma pro vytváření a provozování aplikací, zahrnující řadu jazyků (C#, C++, Visual Basic – VB.NET, Pearl a další).
Systém *.NET Framework* je souhrn objektů a šablon pro vytváření aplikací .NET, jak desktopových tak webových. Neustále se rozšiřuje a ve svých nových verzích přidává celou řadu nových prvků, podle požadavků současného rozvoje informačních technologií.
Microsoft .NET Framework se skládá z několika komponent a mezi ty nejzákladnější patří společný jazykový běhový modul (Common Language Runtime – CLR) a knihovna tříd rámce (Framework Class Library – FLC). Microsoft .NET Framework podporuje mnoho programovacích modelů. Kromě vytváření webových služeb XML můžete psát konzolové aplikace, aplikace s grafickým uživatelským rozhraním GUI (v .NET nazvané „formuláře Windows" – WinForms), webové aplikace („webové formuláře" – WebForms), a nebo dokonce i služby Windows, častěji označované jako služby NT.

Odpověď na kontrolní otázku 1.2

Jakmile je otevřen vykonatelný soubor např. *.aspx, překladač jazyka .NET vygeneruje řízený (někdy také nazvaný jako spravovaný – managed) kód, který se skládá z instrukcí zapsaných v kódu MSIL. Tento kód je někdy také označován jako společný zprostředkovací jazyk CIL (Common Language Interface). Instrukce MSIL se na požádání zkompilují metodou JIT (Just In Time) do nativního strojového kódu za běhu. Kompilace JIT funguje tak, že kód, který se nikdy nezavolá, se ani nezkompiluje. Ve většině případů se určitá metoda zkompiluje technikou JIT jen jednou — při svém prvním volání — a následně se uloží do paměti, aby ji příště bylo možné vykonat bez zpoždění.

Odpověď na kontrolní otázku 1.3

Vývoj ASP.NET stránek se od ostatních technologií odlišuje. V mnoha odlišnostech poskytuje právě výhody tvorby web aplikací pomocí ASP.NET:

- ASP.NET je svázáno s .NET frameworkem, stránky se neinterpretují ale kompilují.
- ASP.NET stránky jsou objektově orientované.
- ASP.NET můžeme psát ve svém oblíbeném jazyce.

ASP.NET poskytuje automatickou správu paměti, typovou bezpečnost, zpracování vyjímek, ladění web aplikací, pokročilé prvky zabezpečení a další.

Odpověď na kontrolní otázku 1.4

ADO.NET je model Microsoftu pro práci s daty v *.NET Frameworku*. ADO.NET představuje databázové rozhraní pro aplikace poskytované jako sada tříd *.NET Frameworku*. Jedná se o novou generaci technologie přístupu k datům ActiveX Data Object (ADO), obsahující mnoho vylepšení. Na rozdíl od ADO a OLE DB, kteří jsou jejími bezprostředními předchůdci, byla platforma ADO.NET vyvíjena především pro práci na webu. Výborně také podporuje XML a umožňuje tak používat jak relačními data, tak i data v podobě XML.

Klíč k řešení – kapitola 2

Odpověď na kontrolní otázku 2.1

Pokud chceme stránky prohlížet v jiném prohlížeči, který máme nainstalovaný na svém PC, nepoužijeme defaultní volbu spouštění stránky ve Visual Studiu, ale musíme v *Solution Exploreru* kliknout pravým tlačítkem na požadovanou stránku a zvolit volbu *Browse With…*

Poté můžeme ze seznamu prohlížečů vybrat ten, ve kterém chceme stránku zobrazit. V tomto okně můžeme také přidat další prohlížeče do seznamu a nastavit zvolený prohlížeč jako implicitní. Poté již můžeme spouštět stránky ve Visual Studiu klasickým způsobem či v *Solution Exploreru* volbou *View in Browser*.

Odpověď na kontrolní otázku 2.2

Mezi výhody tvorby web aplikací ve Visual Studiu patří: WYSIVYG model, používání šablon, nástroj IntelliSense, minimalizace psaní kódu, rozbalovací části kódu, vestavěný web server, možnosti ladění a sledování proměnných, panely nástrojů, drag & drop modul ovládacích prvků, pokročilé průvodce a mnoho dalších.

Odpověď na kontrolní otázku 2.3

IntelliSense je výborná pomůcka pro zjednodušení psaní, která mimo jiné také ve svém důsledku minimalizuje počet syntaktických chyb, které uživatel může vytvořit. *IntelliSense* je nástroj který na základě vámi psaného zdrojového textu nabízí možné použití výběru dalších zdrojových slov. Můžete tedy například vybrat ze seznamu dlouhý název ovládacího prvku, který byste jinak psali ručně. *IntelliSense* také nabízí všechny dostupné metody a vlastnosti při použití konkrétního objektu, třídy či proměnné. Tím je zajištěna zmíněná redukce chyb, neboť při použití *IntelliSense* se vývojáři nemůže stát, že vybere metodu či vlastnost, která pro daný objekt není definována.

Odpověď na kontrolní otázku 2.4

Mezi typy souborů, se kterými se nejčastěji setkáme, patří: *.aspx*, *.ascx*, *.asmx*, *.vb*, *.cs*, *web.config*, a *global.asax*.

Aspx je přípona ASP web stránek v prostředí .NET. Tyto stránky se po kompilaci zobrazují koncovým uživatelům v internetovém prohlížeči. Příponou *ascx* jsou reprezentovány uživatelské ovládací prvky. Tyto prvky jsou podobné aspx stránkám, obsahují však pouze části uživatelského rozhraní, které lze v různých aplikacích opětovně využívat. Tvorbě uživatelských prvků se budeme věnovat v samostatné kapitole. Příponou *asmx* jsou označeny webové služby. Tyto služby se chovají jinak než *aspx* stránky, využívají však stejné zdroje aplikace a různá konfigurační nastavení. Dalším typem dokumentů uložených v projektu aplikace jsou zdrojové kódy oddělené od ASPX stránek. Jsou psány nejčastěji v programovacích jazycích Visual Basic a C# a tomu odpovídají přípony *vb* a *cs*.

Odpověď na kontrolní otázku 2.5

Pokud chceme aplikaci ladit, zvolíme z nástrojové lišty tlačítko *Start Debugging*. Obdobně jako u desktopových aplikací využíváme zarážky – tzv. *breakpointy*. Pokud máme umístěnou zarážku kdekoliv v programu a spustíme ladění, program se zastaví na daném řádku kódu. Zastavený program je na řádku označený červeným puntíkem, jenž symbolizuje vložený *breakpoint*. V menu *debug* máme mimo jiné nabídku možností *Step Into*, *Step Over*, *Step Out* a *Continue*. Tyto volby mají také přiřazeny horké klávesy a nástrojové ikonky. Step Into znamená krokování dovnitř a znamená fakt, že pokud na daném řádku bude volání nějaké funkce či metody, bude krokování pokračovat právě uvnitř

této funkce. Pokud zvolíme Step Over – krokovat přes, toto volání funkce provedeme celé, neboť je reprezentováno právě jedním řádkem kódu. Odkrokovat ven z volané procedury můžeme příkazem Step Out. Pokud nepotřebujeme program krokovat, zvolíme *Continue* – pokračovat a program poběží dále, dokud nenarazí na další *breakpoint*.

Odpověď na kontrolní otázku 2.6

Soubor web.config je konfigurační soubor každé web aplikace v ASP.NET. Je založen na formátu XML a jeho obsahem jsou široké možnosti konfigurací od nastavení ladění, přes konfiguraci databázového připojovacího řetězce až po bezpečnost celé web aplikace.
Jeho důležitými XML elementy jsou sekce <appSettings /> a <system.web /> do kterých se umísťují další emementy XML, které konfigurují celou aplikaci.

Klíč k řešení – kapitola 3

Odpověď na kontrolní otázku 3.1

Každý jmenný prostor je kolekcí šablon objektů pro příslušnou funkcionalitu aplikace. Pokud budeme chtít například pracovat s OLE databází a využívat objektů pro vázání dat, budeme muset příslušné jmenné prostory naimportovat.
Jmenné prostory připomínají adresářovou strukturu a umožňují seskupit spolu související funkce a typy na jednom pojmenovaném logickém místě, čímž je vývojářům usnadněna orientace. Dále řeší konflikty v pojmenování, neboť jméno funkce či typu musí být jedinečné pouze v daném jmenném prostoru. Pro zápis se používá tečková notace, kdy jména jednotlivých do sebe zanořených prostorů a jméno typu umístěného ve jmenném prostoru jsou oddělena tečkami.
Výchozí jmenný prostor, který obsahuje kromě většiny dalších systémových prostorů deklarace všech základních tříd a datových typů, událostí, atributů a výjimek, se jmenuje ***System***. Základní podobu každé stránky ASP.NET aplikace tvoří šablona objektu ***System.Web.UI.Page***. Každá zkompilovaná web stránka bude postavena na uvedené šabloně tohoto objektu a bude mít určen způsob své funkčnosti.

Odpověď na kontrolní otázku 3.2

Použijeme metodu IsPostBack.

```
If not Page.IsPostBack then
     'Blok kódu pro první načtení stránky
 else
     'Blok kódu pro každé další načtení stránky
end if
```

Odpověď na kontrolní otázku 3.3

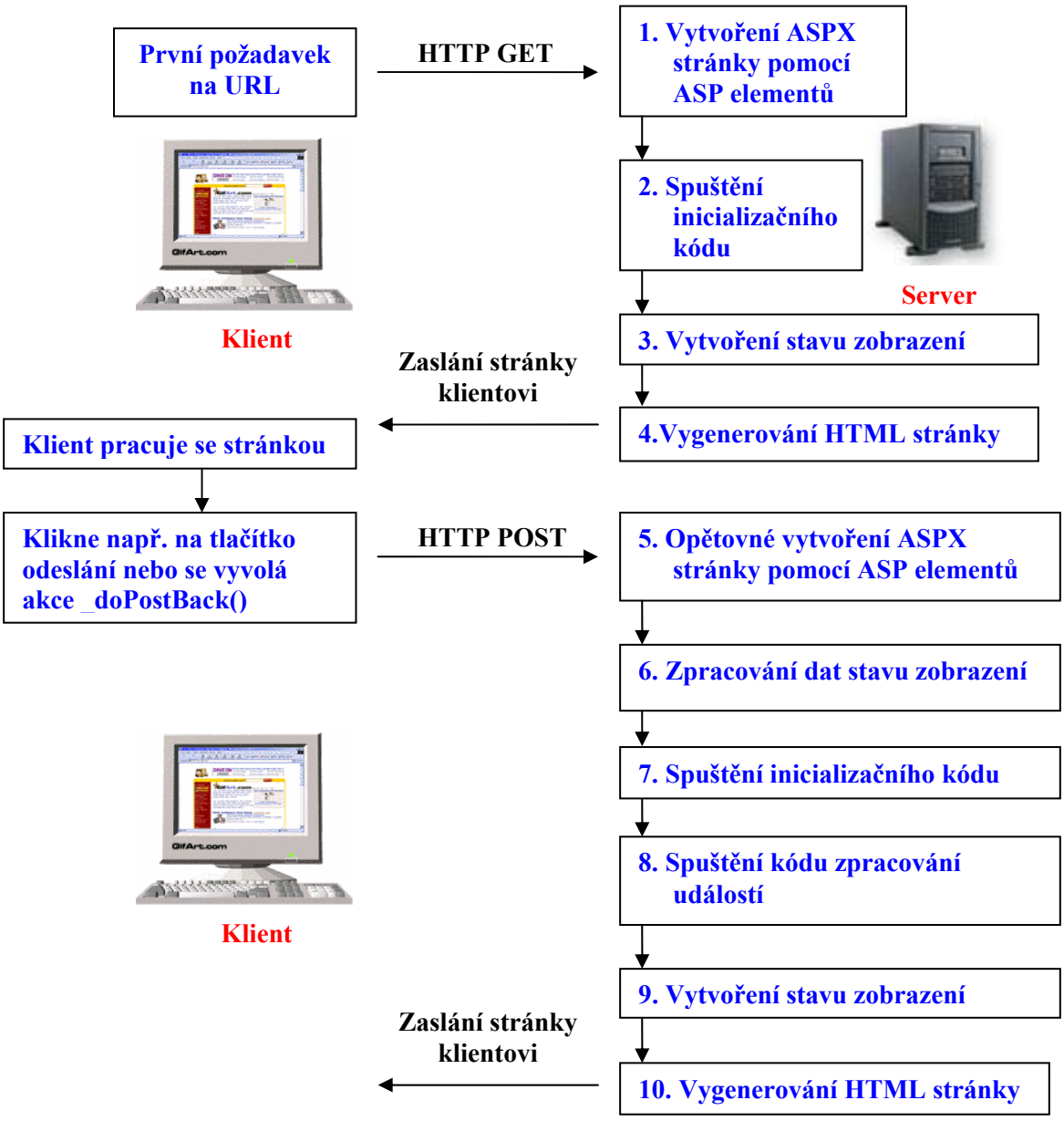
Jakmile je relace přerušena, dojde ke ztrátě těchto informací. To je možné několika způsoby: zavřením prohlížeče, přistoupením aplikace z jiného okna prohlížeče, automatickým vypršením z důvodu nečinnosti a nebo programovým kódem na ukončení relace Session.Abandon().

Odpověď na kontrolní otázku 3.4

Session neboli relace je důležitým pojmem v oblasti správy stavu webových aplikací a často využívanou skutečností. Každý klient, který má v prohlížeči spuštěnou web aplikaci, udržuje se serverem tzv. relaci neboli Session.

V této relaci udržuje svou kolekci informací, která je jedinečná právě pro tohoto klienta. Naproti tomu Objekt httpApplication pracuje s aplikací globálně, to znamená že pro proměnnou budou mít všichni klienti přiřazenou stejnou hodnotu.

Odpověď na kontrolní otázku 3.5



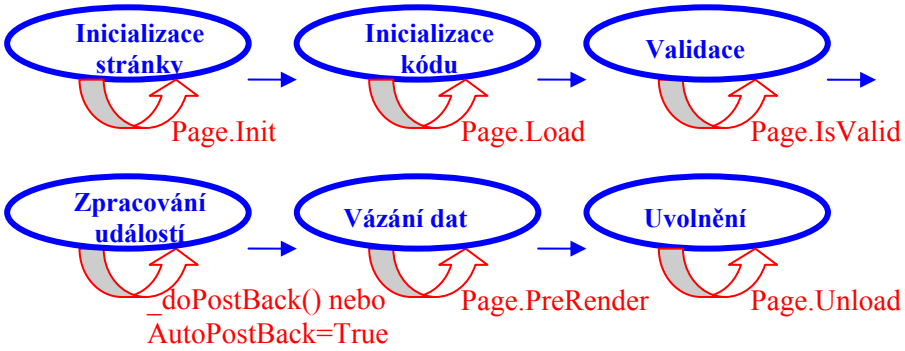
Odpověď na kontrolní otázku 3.6

Objekt Trace je mocný monitorovací nástroj, využívající nejen při ladění web aplikace. Umožňuje zapisovat události do konkrétního log souboru, ve kterém můžeme spatřit i časové údaje o proběhnutých událostech. Do protokolu o sledování můžeme zapisovat i vlastní zprávy pomocí metod *Trace.Write()* nebo *Trace.Warn()*.

Klíč k řešení – kapitola 4

Odpověď na kontrolní otázku 4.1

Životní cyklu stránky můžeme zobrazit podle pořadí jednotlivých událostí:



Odpověď na kontrolní otázku 4.2

AutoPostBack je schopnost automatického odeslání zpět na server (AutoPostBack). Je to vlastnost serverových ovládacích prvků.

Odpověď na kontrolní otázku 4.3

Ovládací prvky můžeme v ASP.NET rozdělit do několika kategorií. Jsou jimi HTML prvky, webové ovládací prvky, které jsou používány nejčastěji, a dále skupiny prvků pro validaci dat, pro práci s daty, WebParts, Login a Navigation.

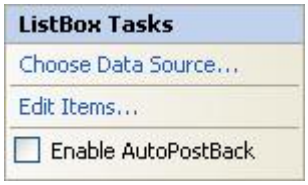
Webové ovládací prvky si rozdělme do třech kategorií:
1. Standardní webové ovládací prvky – serverové ASP prvky obdobné k HTML prvkům (např. tlačítko, popisek, textbox, atd.).
2. Webové ovládací prvky pro seznamy – ASP prvky pro různé druhy seznamů, tzv. listů.
3. Speciální webové ovládací prvky – speciální ASP prvky jako např. kalendář.
Všechny webové ovládací prvky se deklarují značkou <asp:prvek>. Webové ovládací prvky na rozdíl od prvků HTML mají schopnost automatického odeslání zpět na server (AutoPostBack).

Odpověď na kontrolní otázku 4.4

Ovládací prvky HTML jsou jistou alternativou, se kterou ovšem mnohdy nevystačíme. Proto je k dispozici vybudována celá základna webových ovládacích prvků, které nabízejí oproti HTML prvkům bohatší výbavu a funkcionalitu. Webové ovládací prvky na rozdíl od prvků HTML mají schopnost automatického odeslání zpět na server (AutoPostBack).
Všechny webové ovládací prvky jsou serverové, takže musí obsahovat povinný atribut `runat="server"`.

Odpověď na kontrolní otázku 4.5

Visual studio nabízí v design módu rychlé nastavení funkčnosti ovládacích prvků, tzv. T*asks*. Tato možnost je přístupná v pravém horním rohu při editaci prvku. Na obrázku vidíme *Tasks* ovládacího prvku ListBox.



Můžeme zvolit přímo datový zdroj pro ovládací prvek nebo můžeme přímo vytvářet položky v nabízeném editoru (viz. obrázek). Pokud dáváme přednost psaní zdrojového kódu, můžeme vše vyřešit i tímto způsobem za použití všudypřítomného nástroje *IntelliSense*.

Klíč k řešení – kapitola 5

Odpověď na kontrolní otázku 5.1

Některé standardní ovládací prvky:

ASP.NET značka	HTML ekvivalent	Klíčové členy
<asp:Buton>	<input type="buton">	Text, CausesValidation, PostBackUrl, ValidationGroup, Click
<asp:CheckBox>	<input type="checkbox">	AutoPostBack, Checked, Text, TextAlign, CheckedChanged
<asp:FileUpload>	<input type="file">	FileBytes, FileContent, FileName, HasFile, PostedFile, SaveAs()
<asp:HiddenField>	<input type="hidden">	Value
<asp:HyperLink>	<a>	ImageUrl, NavigateUrl, Target, Text
<asp:Image>	<img>	AlternateText, ImageAlign, ImageUrl
<asp:ImageButton>	<input type="image">	CausesValidation, ValidationGroup, Click
<asp:ImageMap>	<map>	HotSpotMode, HotSpots, AlternateText, ImageAlign, ImageUrl
<asp:Label>		Text, AssociatedControlID
<asp:LinkButton>	<a><img>	Text, CausesValidation, ValidationGroup, Click
<asp:Panel>		BackImageUrl, DefaultButton, GroupingText, HorizontalAlign, Scrollbars, Wrap
<asp:RadioButton>	<input type="radio">	AutoPostBack, Checked, GroupName, Text, Textalign, CheckedChanged
<asp:Table>	<table>	BackImageURL, CellPadding, CellSpacing, GridLines, HorizontalAlign, Cells
<asp:TableCell> <asp:TableRow>	<td> <tr>	ColumnSpan, HorizontalAlign, RowSpan, Text, VerticalAlign, Wrap, Cells
<asp:TextBox>	<input type="text"> <textarea>	AutoPostBack, Columns, MaxLength, ReadOnly, Rows, Text, TextMode, Wrap, TextChanged

Odpověď na kontrolní otázku 5.2

Ovládací prvky pro seznamy jsou následující:

Ovládací prvek	Charakteristika
<asp:ListBox>	Otevřený seznam s kolekcí prvků <asp:listItem>
<asp:DropDownList>	Rozbalovací seznam s kolekcí prvků <asp:listItem>
<asp:RadioButtonList>	Seznam s přepínači
<asp:CheckBoxList>	Seznam se zaškrtávacími poli
<asp:BulletedList>	Odrážkový či číselný seznam s HTML ekvivalenty

Odpověď na kontrolní otázku 5.3

Metoda Focus() je novinkou v ASP.NET 2.0. U ovládacích prvků, kde je vyžadován vstup z klávesnice můžeme nastavit *focus*, což je možnost, že po načtení formuláře do prohlížeče se začne

pracovat jako první s tímto ovládacím prvkem. Metodu focus nelze uplatnit u HTML ovládacích prvků a z dřívějších dob si pamatujeme, že jsme tuto vymoženost museli programovat javascriptem.

Odpověď na kontrolní otázku 5.4

MultiView je ovládací prvek umožňující deklarovat více zobrazení a přitom v dané situaci zobrazit pouze jedno. Nemá ale žádné výchozí rozhraní, vše musíme obstarat sami pomocí HTML prvků a dalších serverových ovládacích prvků.

Odpověď na kontrolní otázku 5.5

Barvy zajišťuje objekt Color ze jmenného prostoru System.Drawing. Barvy definujeme známým způsobem ve formátu *#<Red><Green><Blue>*. Takže například zelenou barvu nastavíme hodnotou *#00FF00*. Druhou možností je použití předdefinovaného názvu barev .NET ze třídy *Color*. Obdobným způsobem můžeme zadat název HTML barvy, tu specifikujeme jako řetězec pomocí třídy *ColorTranslator*. Posledním způsobem je možnost vytvořit barvu pomocí RGB ještě s hodnotou alfa kanálu, tzv. ARGB.

Klíč k řešení – kapitola 6

Odpověď na kontrolní otázku 6.1

Validační prvek	Charakteristika
<asp:RequiredFieldValidator>	Zkontroluje, zda-li ovládací prvek, do kterého se vyplňují data není prázdný.
<asp:CompareValidator>	Zkontroluje porovnáním, zda-li požadovaná hodnota vyhovuje předepsané hodnotě.
<asp:RangeValidator>	Zkontroluje, zda-li vstupní hodnota vyhovuje určitému povolenému rozsahu hodnot.
<asp:RegularExpressionValidator>	Zkontroluje, zda-li vstupní hodnota vyhovuje předepsanému regulárnímu výrazu.
<asp:CustomValidator>	Zkontroluje, zda-li vstupní data odpovídají předepsané vlastní logice, kterou implementoval vývojář aplikace.
<asp:ValidationSummary>	Zobrazí souhrné chybové hlášení ze všech přítomných validátorů.

Odpověď na kontrolní otázku 6.2

Nejčastěji se ověřují vstupní data z textboxů, ale i další prvky můžeme validovat. Patří mezi ně ListBox, DropDownList a RadioButtonList, u kterých ověřujeme platnost vybrané položky. V seznamu možných prvků pro validaci jsou i HTML ovládací prvky InputText, TextArea a Select.

Nemáme však žádné mechanismy pro ověřování zaškrtnutí položky v RadioButtonu či CheckBoxu.

Odpověď na kontrolní otázku 6.3

Datový typ může nabývat hodnot Integer, Double, String, Currency a Date a nastavuje se u RangeValidátoru a CompareValidátoru.

Odpověď na kontrolní otázku 6.4

Regulární výrazy jsou tvořeny metaznaky a kvantifikátory. Metaznaky určují různé zápisy libovolných znaků, které se mohou v řetězci vyskytovat. Pomocí metaznaků definujeme přípustné posloupnosti znaků. Aby byl nástroj regulárních výrazů ještě mocnější, jsou zavedeny tzv. kvantifikátory, které určují počet výskytů jednotlivých skupin metaznaků.

Metaznaky	Charakteristika
.	Tečka reprezentuje jakýkoliv znak kromě \n
[abc]	Vyhovuje jakýkoliv jediný znak, který je uveden v množině
[^abc]	Vyhovuje jakýkoliv znak, který není uveden v množině
[1-9a-zA-Z]	Vyhovující jsou znaky z uvedeného rozsahu
\w	Vyhovuje jakýkoliv slovní znak, tj. písmena, číslice a podtržítko
\W	Vyhovuje jakýkoliv jiný znak než písmeno, číslice a podtržítko
\s	Vyhovuje jakýkoliv prázdný znak (mezera, tabulátor, nový řádek atd.)
\S	Vyhovuje jakýkoliv neprázdný znak
\d	Vyhovuje jakýkoliv znak, který je číslicí
\D	Vyhovuje libovolný znak, který není číslicí

Kvantifikátor	Charakteristika
*	Vyhovuje jakýkoliv počet výskytů (i nulový)
+	Vyhovuje jeden nebo více výskytů
?	Vyhovuje žádný nebo jeden výskyt
{N}	Vyhovuje N výskytů
{N,}	Vyhovuje N nebo více výskytů
{N,M}	Vyhovuje N až M výskytů

Odpověď na kontrolní otázku 6.5

Vlastnost *Operator* určuje, jakým způsobem se hodnoty porovnají. Máme na výběr tyto možnosti: *Equal*, *NotEqual*, *GreaterThan*, *GreaterThanEqual*, *LessThan* a *LessThanEqual*. Kromě těchto hodnot známých relačních operátorů máme ještě možnost zvolit hodnotu *DataTypeCheck*, která zkontroluje, zda-li hodnota odpovídá datovému typu ve vlastnosti *Type*.

Odpověď na kontrolní otázku 6.6
Vlastnosti *ControlToCompare* pro porovnání s hodnotou jiného ovládacího prvku a *ValueToCompare* pro porovnání s konstantní hodnotou patří *CompareValidátoru*. Máme možnost použít pouze jednu z těchto možností.
ErrorMessage je chybové hlášení, které nastavujeme u všech validátorů v případě nevyhovujících vstupních dat. *ValidationExpression* je vlastnost *RegularExpressionValidátoru* a vkládáme do ní tvar regulárního výrazu pro validaci vázaného ovládacího prvku. onServerValidate je vlastnost CustomValidátoru a definujeme v ní logiku vlastní validace na serveru.

Klíč k řešení – kapitola 7

Odpověď na kontrolní otázku 7.1

Uživatelský ovládací prvek umožňuje opětovné použití nějaké části web stránky ve více stránkách, může obsahovat HTML kód, webové ovládací prvky i vlastnosti, metody a události. Tento prvek je vhodné implementovat v těch případech, kdy nám nestačí použití vestavěných ovládacích prvků, když nějaký prvek postrádáme a nebo když potřebujeme určité bloky prvků sloučit v jeden formulář, který opětovně využíváme.

Odpověď na kontrolní otázku 7.2

Uživatelský ovládací prvek vytvoříme buďto konverzí některé připravené stránky aspx a nebo vytvoříme v Solution Exploreru nový *Web User Control*. Do něj pak vepíšeme potřebné HTML elementy, ovládací prvky, popřípadě další části. Zdrojový kód obsahuje na začátku direktivu *Control*. Je uložen s příponou ascx.

Odpověď na kontrolní otázku 7.3

Uživatelské prvky často tvoříme jako triviální serverové ovládací prvky jako např. *Label*, protože mnohdy jej využijeme vícekrát a nemusíme pokaždé nastavovat vlastnosti jako velikost, font, barvy apod. Můžeme ale také vytvořit složitější konstrukce uživatelských ovládacích prvků se skriptem, s využitím objektů ASP.NET, s vlastními metodami i událostmi.

Odpověď na kontrolní otázku 7.4

Pokud budeme chtít vytvořit ovládací prvek s již existující *aspx* stránky, musíme dodržet následující postup:

1. Odstranit elementy <*html*>, <*body*> a <*form*>, protože tyto značky mohou být ve zdrojovém kódu stránky pouze jednou, takže je nemůže obsahovat uživatelský ovládací prvek, který pak budeme registrovat do hlavní stránky.

2. Odstranit direktivu *Page*, popřípadě ji změnit na direktivu *Control*, se všemi přípustnými atributy.

3. Zaměnit příponu *aspx* na *ascx*.

Odpověď na kontrolní otázku 7.5

Základem stránky, kde budeme chtít využít ovládací prvek, který jsme vytvořili, je direktiva *Register*. Ovládací prvek se svým názvem musíme zaregistrovat a vytvořit pro něj vlastní prefix. Dalším atributem direktivy je název ovládacího prvku a odkaz na soubor ascx s ovládacím prvkem.

```
<%@ Page Language="VB"%>
<%@ Register TagPrefix="Můj_Definovaný_Prefix"
TagName="Muj_Definovaný_TagLabel"
Src="Soubor_S_Mým_Prvkem.ascx" %>
```

Máme-li zaregistrovaný vlastní prefix, pokud jej použijeme, Visual Studio jej rozpozná a pomocí IntelliSense nám nabídne všechny zaregistrované uživatelské ovládací prvky s tímto prefixem.

Příklad registrace a použití uživatelského ovládacího prvku:

```
<%@ Register TagPrefix="MyAsp" TagName="Dnes" Src="datum.ascx" %>

<MyAsp:Dnes ID="dnes1" runat="server" />
```

Klíč k řešení – kapitola 8

Odpověď na kontrolní otázku 8.1

Stránkování tabulky v Grid View zajistíme buďto ručně dopsáním atributů **`AllowPaging="True" PageSize="Velikost stránky"`** nebo zaškrtávacím tlačítkem *Enable Paging* v *GridViewTasks*. Velikost stránky potom nastavíme ve vlastnostech ovládacího prvku Grid View.

Odpověď na kontrolní otázku 8.2

Typ datového zdroje určíme v *GridView Tasks* položkou *Choose Data Source*. Máme v seznamu nabídku již vytvořených datových zdrojů, nebo můžeme vytvořit zdroj nový z možností objektu, databáze, XML zdroje či SiteMap.

Odpověď na kontrolní otázku 8.3

Datový pohled je reprezentován třídou Data View a je to pohled na reprezentaci dat z tabulky v takové formě, která nám umožní různá nastavení pro filtrování a řazení. Je to užitečný nástroj při technice vázání dat. Zobrazuje podmnožinu tabulky, se kterou chceme pracovat.

Odpověď na kontrolní otázku 8.4

Připojovací řetězec je definice cesty ke zdrojové databázi a najdeme ho v souboru web.config:

```
    <connectionStrings>
              <add name="DatabaseConnectionString1" connectionString="Data
                   Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\Dat
                   abase.mdf;Integrated Security=True;User Instance=True"
                  providerName="System.Data.SqlClient" />
    </connectionStrings>
```

Dále můžeme najít připojovací řetězec u ovládacích prvků pro zdroj dat jako např:

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$
ConnectionStrings:DatabaseConnectionString1 %>">
```

A samozřejmě připojovací řetězec najdeme u objektů, které jej vyžadují. Můžeme jej zapsat ručně, či použít připojovací řetězec z konfiguračního souboru web.config.

```
Dim MyConnection As New SqlConnection("Data
Source=.\SQLEXPRESS;AttachDbFilename=D:\UT_Příklady\Kapitola8\App_Data\Data
base.mdf;Integrated Security=True;Connect Timeout=30;User Instance=True").
```

Odpověď na kontrolní otázku 8.5

Budeme-li pracovat s datovou množinou *DataSet*, musíme naimportovat jmenný prostor `System.Data`.

Odpověď na kontrolní otázku 8.6

DataGrid je obdobou Grid View, jde to poznat ze zdrojového kódu, byl implementován v předchozí verzi ASP.NET a tak je Grid View jeho nástupcem. Pokud bychom chtěli používat Data Grid, museli bychom mnoho funkcionalit obhospodařit ručně, což znamená psaní množství kódu.

Data Grid neobsahuje v *DataGridTasks* možnosti jako editování, mazání záznamů, řazení a stránkování, a tak ke všem těmto událostem musíme dopsat procedury ručně.

Klíč k řešení – kapitola 9

Odpověď na kontrolní otázku 9.1

Pokud chceme přizpůsobit obsah zobrazovaných buněk svým potřebám, přidávat do nich různý HTML kód či ovládací prvky, nevystačíme s vázanými sloupci a musíme definovat svou vlastní šablonu ve sloupci TemplateField.

Pokud v *GridView Tasks* klikneme na položku Edit Templates, máme k dispozici volby pro editaci šablon. Tuto editaci můžeme provést i přímo ve zdrojovém kódu za použití IntelliSense. Šablony mají obdobnou funkci jako vázané sloupce a jejich přehled uvádí tabulka.

Šablona	Charakteristika
HeaderTemplate	Určuje vzhled hlavičky
FooterTemplate	Určuje vzhlad zápatí
ItemTemplate	Určuje vzhled jednotlivých buněk tabulky
AlternatingItemTemplate	Určuje odlišný vzhled lichých řádků
EditItemTemplate	Určuje vzhled editace Grid View

Odpověď na kontrolní otázku 9.2

Ovládací prvek Repeater je charakteristický tím, že je definován bez defaultního vzhledu, vše zajišťujeme sami pomocí šablon. Tyto šablony určí výslednou podobu zobrazených dat. Šablon repeatru je pět:

- *ItemTemplate* – Povinná šablona Repeatru vytvářející pro datový záznam jeden řádek výstupu. Vlastní data jsou vázána podle popsaného mechanismu výše.
- *AlternatingItemTemplate* – Šablona pro odlišení lichých a sudých řádků.
- *SeparatorTemplate* – Zobrazuje oddělovač mezi jednotlivými řádky dat.
- *HeaderTemplate* a *FooterTemplate* – Zajišťuje zobrazení hlavičky a zápatí tabulky.

Odpověď na kontrolní otázku 9.3

Ovládací prvek Data List je mezistupněm mezi Repeatrem a Grid View. Do jisté míry se podobá Repeatru, je také založen na šablonách, obsahuje navíc možnosti komunikace s uživatelem při modifikaci záznamů. Základní šablony jsou shodné s Repeatrem, obsahuje však navíc dvě šablony:

- *SelectedItemTemplate* – Obsahuje další elementy pro výběr položky.
- *EditItemTemplate* – Definuje rozložení Data Listu v editačním módu.

Odpověď na kontrolní otázku 9.4

Vícesloupcové zobrazení záznamů je možné pouze v Data Listu, tuto vlastnost nemá ovládací prvek Grid View. Proto budeme-li potřebovat zobrazit více záznamu vedle sebe či pod sebe, použijeme v Data Listu vlastnosti:

```
RepeatColumns="PočetSloupců" RepeatDirection="Horizontal|Vertical"
```

Odpověď na kontrolní otázku 9.5

Hierarchické vázání docílíme pomocí ovládacího prvku Tree View. Ten dokáže zobrazit úplnou strukturu XML dokumentu. Vázání dat se provádí v sekci `<Databindings>` prostřednictvím `<asp:TreeNodeBinding>`, který určuje vázaný uzel.

Odpověď na kontrolní otázku 9.6

XPath (XML Path Language) je jazyk, pomocí kterého lze adresovat části XML dokumentu. Pomocí tohoto jazyka lze z XML dokumentu vybírat jednotlivé elementy a pracovat s jejich hodnotami a atributy. XPath se používá v mnoha aplikacích XML, mezi nejvýznamnější patří využití v XSLT. Jazyk XPath je standardem vydaným organizací W3C. Základní součástí jazyka je path expression, „výraz popisující cestu“. Taková cesta se zapisuje jako posloupnost přechodů mezi jednotlivými sadami uzlů, oddělených lomítky. Každý přechod je určen pomocí tří složek (některé ovšem nemusí být uvedeny, pokud mají implicitní hodnotu): osa (axis), test (node test), predikát (predicate).

Klíč k řešení – kapitola 10

Odpověď na kontrolní otázku 10.1

Útok injektáží je postaven na principu, že se do aplikace dostanou data od uživatele, které vývojář neočekával. Proto se dynamicky vytvářející SQL dotaz změní na útočný kód, ze kterého může útočník zjistit citlivá data, či je dokonce zničit.

Jaká je vhodná obrana? Můžete omezit maximální délku vloženého řetězce ve vlastnostech textboxu, můžete omezit zadávání speciálních znaků, avšak nejlepším způsobem obrany je použití parametrizovaných dotazů a uložených procedur.

Odpověď na kontrolní otázku 10.2

Parametrizovaný dotaz je příkaz, který používá v SQL řetězci zástupné symboly. Tyto zástupné symboly jsou definovány v sekci <Parameters> a jsou zasílány přímo příkazům SQL.

Odpověď na kontrolní otázku 10.3

Visual Studio poskytuje intuitivní průvodce i pro tvorbu SQL dotazů za použití parametrů. Pohledem na datový zdroj zjistíme několik skutečností. Vlastnost SelectCommand je ve tvaru parametrizovaného dotazu a vygenerovaná sekce `<SelectParameters>` obsahuje parametr, který jsme uvedli v průvodci.

Odpověď na kontrolní otázku 10.4

Uložená procedura je dávka obsahující jeden či celou sadu SQL dotazů. Je uložena přímo v databázi. Můžeme je chápat jako zapouzdřené funkce, které prostřednictvím vstupních parametrů přebírají data a poskytují ucelenou sadu výsledků. Jejich použití je velmi doporučováno. Největšími přednostmi uložených procedur jsou:

- **Přehledná a snadná údržba**

  Protože uložené procedury nejsou součástí zdrojového kódu aplikace, nemusíme při jejich editaci aplikaci znovu kompilovat. Výhodou je rovněž přehledná administrace na jednom místě v Server Exploreru.

- **Bezpečnost aplikace využívající databázi**

  V úvodu kapitoly již byly zmíněny některé nepříliš bezpečné situace při generování SQL dotazů přímo ve zdrojovém kódu aplikace.

- **Zvýšení výkonu**

  Uložené procedury jsou dávkovým proudem příkazů, který může při jednom připojení na databázový server zastat mnoho práce.

Odpověď na kontrolní otázku 10.5

Uložené procedury jsou součástí databázového mdf souboru, takže je najdeme v Server Exploreru. Podobně jako tabulky jsou součástí rozbalovacího menu, a pravým tlačítkem na složce *Stored Procedures* můžeme provádět nabízené akce, jako např. vytvoření nové uložené procedury.

Při konfiguraci datového zdroje pomocí průvodců máme možnost výběru uložené procedury ze seznamu uložených procedur vytvořených pro aktuální datové spojení.

Klíč k řešení – kapitola 11

Odpověď na kontrolní otázku 11.1

Webová služba je novým mechanismem pro výměnu dat mezi aplikacemi. Pomocí XML a dalších technologií umožňuje komunikaci aplikace s jinou aplikací, která ji konzumuje vzdáleně bez ohledu na tom, na jakém hardwaru a softwaru je implementována.

Webové služby umožňují sdílení komponent a jejich funkcí. Je to v podstatě nejlépe demonstrovaná možnost znovupoužití kódu. Nespornou výhodou je údržba vaší aplikace. Nemusíte již instalovat řady programů a aplikačních objektů. Jednoduše je vzdáleně využíváte.

Odpověď na kontrolní otázku 11.2

- WSDL
  Web Service Description Language je jazyk založený na formátu XML. Popisuje doslova všechno, co je třeba, aby se klient o nabízené webové službě dověděl. Jakým způsobem se k webové službě přistupuje, jaké má metody, jaké parametry a datové typy. To vše je nezbytně nutné, abychom mohli vzdálené webové službě porozumět a efektivně využít její funkcionalitu.
- SOAP
  Protokol *Simple Object Access Protocol* je poměrně novým standardem na poli informačních technologií. Umožňuje klientským aplikacím odesílat data na server a přijímat data ze serveru. Využívá formátu XML, který má před http formátem žádosti POST a GET mnohem širší použití. Mohou se tak na místo parametrů a jejich hodnot odesílat složitější struktury dat, objekty a další. SOAP lze také používat i s jinými protokoly než http.
- DISCO
  Je to standard odvozený ze slova *Discovery* (objevování) a vytváří pouze jeden soubor s touto příponou. V tomto souboru jsou umístěné odkazy na přístupné web služby, které daná společnost uveřejňuje na svém serveru.
- UDDI
  Universal Description, Discovery and Integration je centralizovaný adresář nabízených webových služeb. Zde vývojářské firmy nabízejí své služby. Do UDDI adresáře je nutno se zaregistrovat.

Odpověď na kontrolní otázku 11.3

Proxy třída je komponenta, která umožňuje volání webové služby. Díky této přístupové třídě máme možnost zavolat metodu webové služby, jako by šlo o metodu v lokální komponentě. Třída proxy obsluhuje formát SOAP zpráv a řídí jejich přenos pomocí HTTP. Proxy třídu můžeme vytvořit dvěma způsoby:

- Pomocí nástroje příkazového řádku a programu *wsdl.exe.*

  Příkaz bude mít následující syntaxi:

  wsdl /language:VB *http://localhost/nazevprojektu/nazev.asmx?WSDL*

  pokud se bude jednat o vzdálenou službu, místo lacalhost použijeme URL vzdáleného WSDL souboru.
- Pomocí webové reference ve Visual Studiu.
  V tomto případě přidáme webovou referenci v Solution Exploreru projektu. Pomocí průvodce se během několika kliknutí myší přidá do projektu do složky App_WebReferences reference na webovou službu.

Odpověď na kontrolní otázku 11.4
Abychom mohli vyvíjet klientskou aplikaci, musíme mít vytvořenou proxy třídu. Jinak bychom museli psát mnoho zdrojového kódu na nižší úrovni aplikace. Jakmile máme vytvořenou proxy třídu můžeme jednoduchým a známým způsobem postupovat při tvorbě klientské aspx stránky. Konzumujeme tak metody webové služby, jako by byly součástí projektu.

Odpověď na kontrolní otázku 11.5

Standard WSDL zajišťuje samopopisnost webových služeb, popisuje jakým způsobem se k webové službě přistupuje, jaké má metody, jaké parametry a datové typy. Přesto máme možnost webovou službu detailněji popsat. To provedeme prostřednictvím atributu Description, který náleží jak webové službě, tak webové metodě.

Klíč k řešení – kapitola 12

Odpověď na kontrolní otázku 12.1

V *Solution Exploreru* klikneme pravým tlačítkem na název otevřeného projektu a zvolíme položku *Add Web Reference*. Poté co referenci přidáme, uvidíme ve složce projektu *App_WebReferences* hierarchickou strukturu cesty k *wsdl* web služby.

Odpověď na kontrolní otázku 12.2

IntelliSense nám pomáhá při tvorbě instance objektu webové služby její kompletní cestou v hierarchii web reference a také poskytnutím všech metod k danému objektu.

Odpověď na kontrolní otázku 12.3

URL jednoznačně charakterizující wsdl soubor webové služby.

Např: http://www.infoaccelerator.net/net/infoaccelerator/pointCalc/PointCalculator.cfc?wsdl

Odpověď na kontrolní otázku 12.4

Nejprve v projektu přidáme referenci. V Solution Exploreru klikneme pravým tlačítkem na název otevřeného projektu a zvolíme položku *Add Web Reference*. Otevře se průvodce, který umožňuje přidat referenci na webovou službu do projektu. Do pole URL adresy vložíme URL wsdl webové služby, které chceme použít. Potvrdíme tlačítkem *Go*. V textové oblasti na pravé straně vidíme výsledek této akce. Na dané URL adrese bude nalezena webová služba a zobrazen její název. V textboxu níže si můžeme všimnout webové reference v obrácené notaci (www je až na konci). V levém okně průvodce uvidíme popis nalezené služby a zobrazené metody, které služba poskytuje.

V *Solution Explorer v*e složce App_WebReferences uvidíme složku s hierarchickou strukturou cesty, např. *com – freewebs – www* s odkazem na webovou službu popsanou souborem wsdl.

Pokračujeme dále a vytvoříme klientskou aplikaci, neboli aspx stránku s formulářem pro konkrétní úlohu. Poté vytvoříme obslužný skript, kde nadefinujeme instanci objektu webové služby a vhodným způsobem konzumujeme potřebné metody.

Odpověď na kontrolní otázku 12.5

Web Service Description Language je jazyk založený na formátu XML. *DISCO* je standard pro objevování web služeb a je odvozený ze slova *Discovery. Simple Object Access Protocol* je protokol, který využívají klientské aplikace. *Universal Description, Discovery and Integration* je centralizovaný adresář nabízených webových služeb.

Rejstřík

Ch

I

J

K

M

O

P

R

S

Š

T

U

V

W

X

Z