Implementace webových služeb v ASP.NET

Implementation of web services in ASP.NET

Petr Martinák

Bakalářská práce
2010

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2009/2010

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Petr MARTINÁK**
Osobní číslo:	**A07061**
Studijní program:	**B 3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Téma práce:	**Implementace webových služeb v ASP.NET**

Zásady pro vypracování:

1. **Proveďte rešerši technologie asp.net, zaměřenou na webové služby.**
2. **Stanovte výhody webových služeb.**
3. **Realizujte ukázku využívají webové služby.**
4. **Proveďte vyhodnocení.**

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **MACDONALD, Matthew, SZPUSZTA, Mario. ASP.NET 3.5 a C-Sharp 2008 : Tvorba dynamických stránek profesionálně. Jan Pokorný, Jan Gregor. 1. vyd. [s.l.]: Zoner Press, 2008. 1584 s. ISBN 978-80-7413-008-3.**
2. **EVJEN, Bill, HANSELMAN, Scott, RADER, Devon. ASP.NET 3.5 v jazycích C-Sharp a Visual Basic. [s.l.]: Computer Press, 2009. 1600 s. ISBN 978-80-251-2069-9.**
3. **BILL, Evjen, et al. ASP.NET 2.0: Programujeme profesionálně. Karel Voráček. Brno : Computer Press, a.s., 2006. 1224 s. ISBN 978-80-251-1473-5.**
4. **Illustrated C-Sharp 2008 . [s.l.]: [s.n.], 2009. 1333 s.**
5. **Programming .NET.3.5. [s.l.]: [s.n.], 2008. 478 s.**
6. **Webové programování v ASP.NET 2.0 . [s.l.]: [s.n.], 2008. 648 s.**

Vedoucí bakalářské práce: **Ing. Petr Šilhavý, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce: **5. března 2010**

Termín odevzdání bakalářské práce: **1. června 2010**

Ve Zlíně dne 5. března 2010

L.S.

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Hlavním cílem bakalářské práce je seznámit se s webovou službou, jakož to neodmyslitelnou součástí webových aplikací. Tato publikace je rozdělena do dvou částí. Kde zejména první část je obsáhlejší, která seznamuje čtenáře nejdříve z historií Internetu, s programovým vybavením pro realizaci webové služby, s neodmyslitelnými součástmi webové služby a v neposlední řadě přináší rozsáhlé informace ohledně webové služby samotné. Druhá část práce je zaměřena na názornou ukázku implementace webové služby do webových stránek.

Klíčová slova: Webová služba, XML, HTTP, ASP.NET, SOAP, WSDL

ABSTRACT

This bachelor thesis deals with the Web service as integral part of web applications. Thesis consists of two parts. At first part is more comprehensive, the history of the Internet, the software for implementing web services, detailed information about the web services and implementation of a web page is introduced in this the first part of thesis. The second part is focused on an illustrative example of the implementation of web services into web site.

Keywords: Web service, XML, HTTP, ASP.NET, SOAP, WSDL

Poděkování:

Poděkování patří mému vedoucímu bakalářské práce, panu Ing. Petru Šilhavému za odborné dohlížení a vedení nad mou prací. A za jeho rady a postřehy při realizaci mé bakalářské práce.

Motto:

„Non scholae, sed vitea discimus.“ (překlad. Neučíme se pro školu, ale pro život.)

Latinské přísloví

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně …….

podpis diplomanta

OBSAH

ÚVOD

S příchodem Internetu se velmi markantně začaly rozvíjet internetové stránky. Protože sloužily zejména firmám, které na Internetu prezentovaly a stále prezentují své výrobky, nabídky, služeb či samotnou firmu je zřejmé, že vyžadovaly co nejprofesionálnější vzhled a funkcionalitu takových to webových stránek. S tímto, ale tvůrci stránek měli svázané ruce, poněvadž využívali v začátcích pouze technologii nazvanou HTML(HyperText Markup Language), kterou zpracoval Berners-Lee v roce 1990. O takových to vytvořených webových stránek, se hovořilo jako o stránkách statických, což pro tvůrce webových stránek znamenalo velmi pracnou editaci každé vzniklé stránky.

Na tuto problematiku, ale zareagovali vývojáři společnosti Microsoft, kteří se snažili vytvořit dynamické stránky využívající nejnovějších programátorských technik. Toto se jim podařilo, když skloubili několik technologii jako je .NET Framework, (Cascading Sytle Sheets) neboli kaskádové styly a objektové programovací jazyky do jedná nazvané ASP.NET. A ta umožňuje vytvářet webové stránky s dynamickým obsahem. Takováto stránka je v podstatě HTML soubor s úseky programového jazyky například C# nebo Visual Basic, u nichž využíváme vkládání objektů do stránky a realizujeme tím i dynamiku takovéto webové stránky.

S tímto souvisí i využití webových služeb na stránkách, o kterých ani nevíme a přesto se s nimi setkáváme vesměs na každé webové stránce. Můžeme se s takovouto webovou službou setkat například u internetových obchodů, kde se platí pomocí karty nebo úplně jednoduše při poslání e-mail ze stránky. Taková to služba je realizovaná díky XML souborům, které odvádí největší práci.

Proto tato bakalářskou práce je věnována implementaci webové služby za pomocí ASP.NET. Začátek práce seznamuje čtenáře s historií Internetu. Následně na to navazuje sekce s technologiemi využívající webové služby a další nutné informaci k pochopení webové služby. Poté se práce zaměřuje a podrobně rozebírá jednotlivé prvky webové služby, jako například jaké standardy takováto webová služba využívá, co je zapotřebí při realizaci webové služby a samozřejmě nedílnou součástí webové služby je i práce s XML soubory. Což je podle mého názoru největší výhoda, kterou této webové služby nabízí. XML soubory jsou čitelné na velkém množství platforem, a tedy není divu, že je využívají právě firmy při jednoduché komunikaci mezi sebou.

Závěr bakalářské práce se věnuje návrhu webových stránek, na kterém by mohla být realizovatelná webová služba. Názorná ukázka webové služby, jako takové na webových stánkách. K tomu bylo využito nabytých poznatků, při vytváření bakalářské práce. Výsledkem jsou tedy webové stránky, které prezentují firmu a kde je zaimplementována webová služba pro kontrolu odesílání e-mailů.

I. TEORETICKÁ ČÁST

1 HISTORICKÝ VÝVOJ

1.1 Webové služby

Webové služby představují novou technologii, můžeme však hodně pochopit již z nedávné historie. Objektově orientované programování se připojilo k hlavnímu programovacímu proudu někdy v osmdesátých letech minulého století. Objektově orientovaný kód sliboval začlenění objektů do kódu, díky čemuž mohli vývojáři vytvářet komponenty, které byly opakovaně použitelné, rozšiřitelné a snadno udržovatelné.

V devadesátých letech minulého století se objevila technologie, která umožnila budovat aplikace sestavováním komponent. Vznikly dvě dominantní technologie založené na komponentách – COM (Component Object Model) a COBRA (Common Object Request Broker Architecture). Pro tyto komponenty byly vytvořeny standardy, použity u distribuovaných komponent tak, aby aplikace mohla pracovat s objekty umístěnými v různých počítačových zapojeních do sítě. COM i COBRA jsou sice velice sofistikované komponenty avšak obtížně nastavitelné. Dalším problémem bylo to, že tyto dvě technologie společně nemohly fungovat. S objevením Internetu se pak logicky objevil exponenciální nárůst dalších problémů. To mělo za následek, že ikdyž jsou technologie COM a COBRA velice komplexní a současně proprietární, ani jedna z těchto technologií se neprosadila.

Webové služby jsou novou technologií, která má za cíl tyto nedostatky odstranit za pomocí rozšíření technologií objektů a komponent do webového prostředí. Na rozdíl od dřívějších technologií jsou webové služby navrženy výhradně pro vytváření mnohem modulárnějších aplikací. [1]

1.2 HTML, CSS

1.2.1 HTML

V roce 1989 spolupracovali Tim Berners-Lee a Robert Caillau na projektu se zaměřením na propojení informačního systému pro společnost CERN (z franc. Conseil Européen pour la recherche nucléaire). V této době se pro tvorbu dokumentů nejčastěji využívaly jazyky TeX, PostScript a také SGML (Standard Generalized Markup Language). Berners-Lee v roce 1990 navrhl jazyk HTML (HyperText Markup Language) a protokol pro jeho přenos v počítačové síti – HTTP (HyperText Transfer Protocol). V téže době Tim Berners-Lee

napsal první webový prohlížeč, který nazval WorldWideWeb. Následoval rychlý rozvoj webu, takže bylo nutné pro HTML definovat standardy. [13]

1.2.2 CSS

CSS (Cascading Sytle Sheets) neboli kaskádové styly, vznikly jako souhrn metod pro úpravu vzhledu stránek. Je to tedy jazyk, pro popis způsobu zobrazení stránek napsaných v jazycích HTML, XHTML (eXtensible HyperText Markup Language) nebo XML (eXtensible Markup Language).

Jazyk byl navržen standardizační organizací W3C (World Wide Web Consortium). Autorem prvotního návrhu byl Håkon Wium Lie. Byly vydány prozatím dvě úrovně specifikace CSS1 (Cascading Sytle Sheets, level 1) a CSS2 (Cascading Sytle Sheets, level 2). Dokončuje se revize CSS 2.1 (Cascading Sytle Sheets, level 2 revision 1) a pracuje se na verzi CSS3 (Cascading Sytle Sheets, level 3). Hlavním smyslem je umožnit návrhářům oddělit vzhled dokumentu od jeho struktury a obsahu. Původně to měl umožnit už jazyk HTML, ale v důsledku nedostatečných standardů a konkurenčního boje výrobců prohlížečů se vyvinul jinak. [12]

1.3 ASP.NET

Poprvé se technologie ASP.NET, tehdy ještě pod označením ASP (Active Server Pages) se objevila v roce 2002 jako součást první beta verze vývojového prostředí Visual Studio.NET. V dalších beta verzích a samozřejmě i ve finální verzi Visual Studia.NET se označení této technologie ustálilo na ASP.NET. [1]

1.3.1 ASP.NET 1.0

Na počátku vývojáři Microsoftu vycítili svou šanci v ASP.NET v němž viděli budoucnost pro daleko modernější webové aplikace. V porovnání s jinými dřívějšími vývojovými platformami, zprostředkovával lepší funkcionalitu:

- ASP.NET totiž nabídl úplný, objektově orientovaný programovací model, který obsahuje architekturu řízenou událostmi, založenou na ovládacích prvcích, což podporuje zapouzdření kódu a jeho opětovné využívání.
- ASP.NET umožňuje psát kód v jakémkoliv z podporovaných jazyků .NET. Nejznámější jsou zejména Visual Basic či C# (C Sharp).

- ASP.NET slouží jako platforma pro budování webových služeb, což jsou opětovně využitelné jednotky kódu, které mohou volat jiné aplikace přes platformy a hranice dané počítači. [1]

1.3.2 ASP.NET 2.0

Po vydařené první verzi ASP.NET 1.0 následovaly kroky Mircosoftu k vydání druhé verze ASP.NET 2.0, který měl za cíl zredukovat počet řádků kódu, které se musely napsat, až o 70%. Oficiálně měl být ASP 2.0 zpětně kompatibilní z předešlou verzí ASP 1.0, v realitě tato kompatibilita, ale nebyla stoprocentní. Vyzvedl bych několik důležitých novinek jako například:

- **Vzory stránek** – velmi výhodné je funkcionalita, kde můžeme implementovat jednotný vzhled na více stránkách. Využíváme k tomu „master page“, kde si nadefinujeme šablonu a bez dalšího, většího úsilí ji opětovně využíváme. Šablonou si zajistíme, že každá webová stránka bude mít v aplikaci stejné záhlaví, zápatí a jednotné navigační ovládací prvky.
- **Ovládací prvky pro zdroje dat** – s novým modelem ovládacích prvků pro zdroje dat můžeme ve stránce deklarativně definovat, jak má stránka komunikovat se zdrojem dat. Využíváme je tedy zejména při práci s SQL (Structured Query Language) serverem. Kromě **GridView** ASP.NET 2.0 dále nabízí nové ovládací prvky pro zobrazení dat, mezi které patří **DetailView** a **FormView.**
- **Personalizace** – většina webových aplikací pracuje s daty, které jsou specifické pro daného uživatele. Pokud například vytváříme web internetového obchodu, bude asi potřeba, abychom ukládali a podle potřeby načítali poštovní adresu aktuálního uživatele, jeho preference ohledně prohlíženého obsahu webu, obsah nákupního košíku atd. ASP.NET 2.0 se s tímto vypořádal pomocí personalizace, což je vlastně API (Application Programming Interface) určené pro zacházení s informacemi specifickými pro jednotlivé uživatele, které jsou uložené v databázi. Myšlenka personalizace je založena na tom, že ASP.NET vytvoří objekt profilu, v němž pak můžeme kdykoliv přistupovat k informacím specifickým pro daného uživatele.
- **Bezpečnost a členství** – ASP.NET 2.0 rozšiřuje formulářovou autentizaci o nové schopnosti. ASP.NET předně obsahuje automatickou podporu sledování přihlašovacích dokladů uživatele (credentials), bezpečným způsobem ukládá hesla, a provádí autentizaci uživatele na přihlašovací stránce.

- **Bohatě vybavené ovládací prvky** – ASP.NET představuje více než 40 nových ovládacích prvků. Mnohé z nich podporují nové funkcionality, jako jsou prvky zasvěcené bezpečnosti, nebo ovládací prvky webových částí (web parts), které jsou určeny pro portály. [2]

1.3.3 ASP.NET 3.5

V ASP.NET 3.5 se spíše jedná o postupnou evoluci. Nové funkce, které se v nové verzi ASP.NET vyskytují jsou soustředěny do dvou oblastí, a to: LINQ a Ajax.

- **LINQ (Language Integrated Query)** - je sada rozšíření pro jazyky C# a Visual Basic, které umožní psát kód C# nebo VB, který manipuluje s daty v paměti podobně, jako když pokládáme dotazy na databázi.
- **AJAX** - je vícevrstvá technologie, která vývojářům poskytuje široký obsah voleb pro integraci funkcí Ajaxu do stránek ASP.NET. Na základní úrovni lze pomocí AJAX psát mocnější JavaScript, na vyšších úrovních je možné používat serverové komponenty. AJAX Vám umožní integrovat funkce jako: automatické dokončování, přetahovaní myší (drag-and-drop) a animace. [3]

1.4 .NET Framework

1.4.1 .NET Framework 1.0

Je spojen s vývojovým prostředím Visual Studio.NET 2002, které bylo následníkem populárního vývojového prostředí Visual Studio 6.0. Poprvé se objevují jazyky C# 1.0 a Visual Basic. NET. Předtím byl kód napsán v programovacím jazyce Visual Basic překládané ve dvou krocích. Nejprve se generoval mezikód, na který se aplikoval kompilátor jazyka C++. Na scénu přichází i platforma ASP. NET 1.0 pro vývoj webových aplikací. [1]

1.4.2 .NET Framework 2.0

Verze je spojena s vývojovým prostředím Visual Studio 2005. Významnou novinkou je možnost vývoje aplikací s využitím technologie ASP.NET 2.0, podpora sériového portu, objektové prostory, přístupová práva k souborům. S novou verzí frameworku přišla i nová verze programovacích jazyků C# 2.0, Visual Basic 2005, nové třídy v BCL (Based Class Library), podpora pro generika v CLR (Common Language Runtime). S platformou .NET

Framework 2.0 je spojena i platforma ASP.NET 2.0 pro vývoj webových a intranetových aplikací. [2]

1.4.3 .NET Framework 3.5

Nová verze nabízí efektivnější nástroje pro vizuální návrh uživatelského prostředí a vylepšené možnosti ladění, takže si oprávněně zaslouží přívlastek RAD (Rapid Application Development). Vývojové prostředí je úzce spojeno s technologickou platformou. NET Framework 3.5. V současné době je k dispozici opravný balíček SP1 (Service Pack 1) pro Visual Studio 2008 i pro platformu .NET Framework 3.5. Jednotlivé novinky bych vyzvedl:

- ADO.NET Entity Framework
- ASP.NET Silverlight controls
- ADO.NET Data Services
- ASP.NET Dynamic Data
- Podpora pro SQL Server 2008
- LINQ
- .NET Framework Client Release ("Arrowhead"), - 2007 Ribbons [3]

2 PROGRAMOVÉ TECHNOLOGIE

2.1 ASP.NET

ASP.NET společnosti Microsoft, je přepracovaná verze jazyka ASP (Active Server Pages), která umožňuje vytvářet webové stránky s dynamickým obsahem. Stránka ASP je v podstatě HTML soubor s úseky programového jazyky buď C# nebo Visual Basic. Je to skriptovací jazyk, podobně jako tomu je u jazyk PHP (Hypertext Preprocessor). Vyskytuje se zde ovšem zásadní rozdíl, a to ten, že ASP.NET je založen na plné integraci do systému .NET Framework což zaručuje podporu jazyka C#. Další rozdíl mezi PHP a ASP.NET je v tomto:

- PHP (Hypertext Preprocesor) je v podstatě alternativa k ASP, která je Open Source. Neustále se pracuje na rozšiřování tohoto programovacího jazyka, a zatím poslední nejnovější verze je s označením 5. U PHP se jedna o skriptovací jazyk, založený na jazyku C/C++, Javě a Perlu, který je interpretovaný serverem.
- Dalším rozdílem je zcela odlišné psaní kódu. Rozdíl je uveden ve vzorovém příkladu, který vypíše do prohlížeče textový řetězec „Ahoj světe“: [1]

  - Kód ASP.NET:

```
<asp:Label ID="Label1" runat="server" Text="Ahoj světe">
</asp:Label>
```

Obr. 1. – Ukázka kódu, který realizuje prvek **Label** ve webovém prohlížeči

  - Kód PHP:

```
<?php
      echo "Ahoj, světe!";
?>
```

Obr. 2. – Výpis textového řetězce „Ahoj světe“, v jazyce PHP

2.2 Html, Css

2.2.1 HTML

HTML (HyperText Markup Language), který nám říká, že se jedná o značkovací jazyk pro hypertext. Je jedním z jazyků pro vytváření stránek v systému World Wide Web, který umožňuje publikaci dokumentů na Internetu. Při psaní stránky se využívají, příkazy zvané tagy, které dělíme na párové a nepárové. Tyto tagy jsou uzavřeny do ostrých závorkách "< >". Párové tagy vymezují nějakou část textu nebo stránky a nepárové se používají jen pro něco jednorázového (zalomení řádku, obrázek, vložení čáry). [13]

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<!-- Hlavička stránky -->
<title></title> <!-- Párový tag -->
</head>
<body>
Text text text <br> <!-- Nepárový tag -->
text
<!-- Tělo stránky -->
</body>
</html>
```

Obr. 3. – Struktura HTML stránky

2.2.2 CSS

V CSS (Cascading Style Sheets) se píše čím dál častěji. Stále více a více webmasterů kaskádové styly využívá na svých stránkách, a v současnosti patří mezi neodmyslitelnou součást designu stránek.

Syntaxe:

Kaskádové styly mají několik pravidel. Každé pravidlo obsahuje selektor a blok deklarací. Každý blok deklarací pak obsahuje seznam deklarací a každá deklarace sestává z vlastnosti, následuje dvojtečka „:“ a hodnota vlastnosti. Každá deklarace je zakončena středníkem „**;**“ [12]

```
body { font-family: "Georgia","Comic sans ms";
background-color: #005000;
color: #f5f5f5;
text-align: center; }
```

Obr. 4. – Názorný příklad kaskádových stylů

2.3 C#, Visual Basic

2.3.1 C#

C# (C Sharp) je nově vyvinutý jazyk pro .NET, který kombinuje vlastnosti známých a oblíbených programovacích jazyků a přidává k nim některé své nové vlastnosti. I přes to, že si jsou jednotlivé programovací jazyky pro tuto platformu rovny, je C# Microsoftem prosazován jako jazyk hlavní. C# je silně objektově orientovaný jazyk vycházející z programovacích jazyků Java a C++. Mezi nejdůležitější vlastnosti silně objektově orientovaného jazyka C#, které se považují za důležité jsou například:

- **Třídy** – základní stavební prvek při tvorbě objektově orientovaných aplikací obsahující akce (metody) a atributy
- **Struktury** – lze je chápat jako zjednodušené třídy, jejich užitím jsou nejčastěji popisovány vlastní datové struktury.
- **Vlastnosti** – někdy označované jako chytré proměnné
- Pole a jejich „chytrá“ verze nazývaná indexery
- **Zástupci** – typově bezpečné ukazatele na funkce
- **Události** – druh zástupců sloužící ke zpracování asynchronních operací [5]

```
using System;

namespace AhojSvete
{
      class AhojSveteClass
      {
            public static void Main(string[] args)
            {
                  Console.WriteLine("Ahoj svete!");
                  Console.ReadLine();
            }
      }
}
```

Obr. 5. – Názorný příklad "Ahoj světe" v jazyku C#

2.3.2 Visual Basic

Visual Basic se posledních letech stal hlavním programovacím prostředkem na platformě produktů firmy Microsoft. Různé mutace tohoto programovacího jazyka se používají nejen pro programování samostatných (i síťových) aplikací, ale také pro tvorbu maker v balíku programů Microsoft Office (Visual Basic for Applications). Při programování internetových aplikací, spouštěných na straně serveru (ASP) i klienta (Visual Basic Script). Programování ve Visual Basicu je počítáno mezi objektově orientované. Programátor může používat velké množství předdefinovaných objektů, jako jsou formuláře, textová pole pro zadávání a zobrazování dat, příkazová tlačítka, menu, popisky a mnoho dalších objektů. Souhrnně tyto objekty nazýváme ovládacími prvky (anglicky „controls"). Velkým rozdílem oproti programovacímu jazyku C# je ten, že VB (Visual Basic) nepoužívá na konci řádku při psaní kódu středník „**;**", jako tomu je právě u programovacího jazyka C#. [14]

```
Private Sub Button1_Click(ByVa sender As System.Object, ByVal
e As System.EventArgs) Handles Button1.Click
        Label1.Text = "Ahoj světe"
End Sub
```

Obr. 6. – Názorný příklad "Ahoj světe" v programovacím jazyku Visual Basic

2.4 XML

2.4.1 Úvod do XML

V současné době se o XML (Extensible Markup Language) hovoří nejčastěji v souvislosti s Webem a považuje se za nástupce dnes používaného jazyka HTML. Výhoda XML spočívá v tom, že autor stránky může používat vlastní tagy, které dokážou mnohem přesněji označit význam prezentovaných informací. To má velký význam především pro vyhledávání informací. Dnešní Web je informacemi přehlcen a nalézt konkrétní informací je stále obtížnější a mnohdy i nemožné. Tento problém nevyřeší sebelepší prohledávací servery, pokud jim nepomohou autoři stránek, kteří pomocí XML uloží do stránek mnohem více metainformací. XML-dokument bez CSS (Cascading Style Sheets) stylů nelze zobrazit, autoři tak ztratí důvod pro vytváření špatně strukturovaných stránek, které však dosahují požadovaného vzhledu. Základním požadavkem na správně strukturovaný XML-dokument je, aby byl celý uzavřen mezi počátečním a ukončovacím tagem.

Všechny tagy musí být párové, to znamená, že pro počáteční tag `<tag>` musí vždy existovat i ukončovací tag `</tag>`. Tagy jsou samozřejmě v ostrých závorkách „<>“. Výjimku tvoří prázdné elementy, které nemají žádný obsah. Jim odpovídající tag však musí být ukončen znaky „/>“ místo pouhého „>“. [10]

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE dopis SYSTEM "knihovna.dtd">
<knihovna>
  <autor>
    <jméno>Jan</jméno>
    <prijmeni>Novák</prijmeni>
    ...
  </autor>
  <kniha>
   <nazev>XML</nazev>
   <vydavatelstvi>Albatros</ vydavatelstvi >
   ...
  </kniha>
</ knihovna >
```

Obr. 7. – Struktura XML-dokumentu

2.4.2 XML v .NET

V .NET Frameworku společnosti Microsoft zastává XML významnou pozici a dává aplikacím ASP.NET bohatou sadu schopností pro používání dat XML a pro manipulaci s nimi. Mnohé schopnosti ASP.NET využívají XML v pozadí. Jako tomu je u webových služeb, kde webové služby používají model vyšší úrovně, který je nadstavbou infrastruktury XML. XML by se mělo využívat v souvislosti s ASP.NET v těchto obecných situacích:

- Potřebujeme manipulovat s nějakými daty, které jsou už uložena jako XML. To může nastat, pokud se vyměňují informace s nějakou existující aplikací, která používá nějakou speciální odrůdu XML.
- Kdy potřebujeme ukládat svá data jako XML a chceme nechat otevřené dveře pro případnou budoucí integraci. Pokud využijeme XML, jsme si vědomi, že aplikace jiných výrobců mohou být navrženy tak, aby uměly tyto data přečíst.
- Pokud chceme použít nějakou technologii, která je závislá na XML. Například webové služby používají všelijaké standardy, které jsou všechny založeny na XML. [3]

2.4.3 Výhody XML

Když se objevilo XML, hlavní příčinou jeho úspěchu bylo to, že bylo velmi jednoduché. Pravidla XML jsou totiž mnohem stručnější a jednodušší než pravidla jeho předchůdce, kterým bylo SGML (Standard Generalized Markup Language). V průběhu let se však do XML přimíchalo mnoho dalších podporovaných standardů, což mělo za následek to, že používání XML v nějaké profesionální aplikaci rozhodně není jednoduchá. Bez ohledu na to všechno je dnes XML, mnohem prospěšnější než kdykoliv dříve a využíváme jej v moderních aplikacích. Mezi hlavní výhody XML patří:

- **Osvojení** – XML je všudypřítomné. Mnohé firmy používají XML pro ukládání dat nebo o tom aktivně uvažují. Kdykoliv je zapotřebí sdílet nějaké data, je XML automaticky první volbou, která se prozkoumává.
- **Rozšiřitelnost a flexibilita** – XML nevnáší žádná pravidla týkající se sémantiky dat a nesvazují firmy používáním proprietárních sítí. Důsledkem toho je, že XML se dá použít na jakýkoliv druh dat a levněji se implementují.

- **Příbuzné standardy a nástroje** – XML nástroje například „parsery“ a okolní standardy, jako je XML Schema, XPath, které vypomáhají při vytváření a zpracování dokumentů XML. Důsledkem je, že programátor má téměř v každém jazyku k dispozici hotové komponenty pro čtení XML, pro ověřování, zdali XML vyhovuje nějaké sadě pravidel, pro hledání XML, či pro transformaci z jednoho formátu XML do druhého. [2]

3 WEBOVÉSLUŽEBY

3.1 Úvod do webových služeb

Webové služby (anglicky Web Services) jsou základní stavební bloky pro přechod k distribuované výpočetní technice na Internetu. Tato technologie je nezávislá na platformě a založená na standardech W3C. Otevřené standardy a zaměření na komunikaci a spolupráci mezi lidmi a aplikacemi vytvořilo prostředí, ve kterém se webové služby stávají platformou pro integraci aplikací. Aplikace jsou konstruovány k využívání více webových služeb z různých zdrojů a jejich spolupráci, bez ohledu na to, kde jsou umístěny nebo jak byly implementovány. Webová službu bych definoval jako softwarovou službu, vystavenou na web prostřednictvím protokolu SOAP (Simple Object Access Protocol), popsaného souborem WSDL (Web Service Description Language) a registrovaného v UDDI (Universal Description, Discovery, and Integration). Vysvětlení jednotlivých zkratek naleznete v další části práce.

Výhody webových služeb:

- Hlavní výhodou architektury webových služeb je to, že programy mohou být napsány v různých jazycích na různých platformách a že spolu navzájem komunikují standardním způsobem.
- Další podstatnou výhodou webových služeb proti předcházejícím pokusům je to, že pracují se standardními webovými protokoly – XML, HTTP a TCP/IP (Transmission Control Protocol/Internet Protocol). [2]

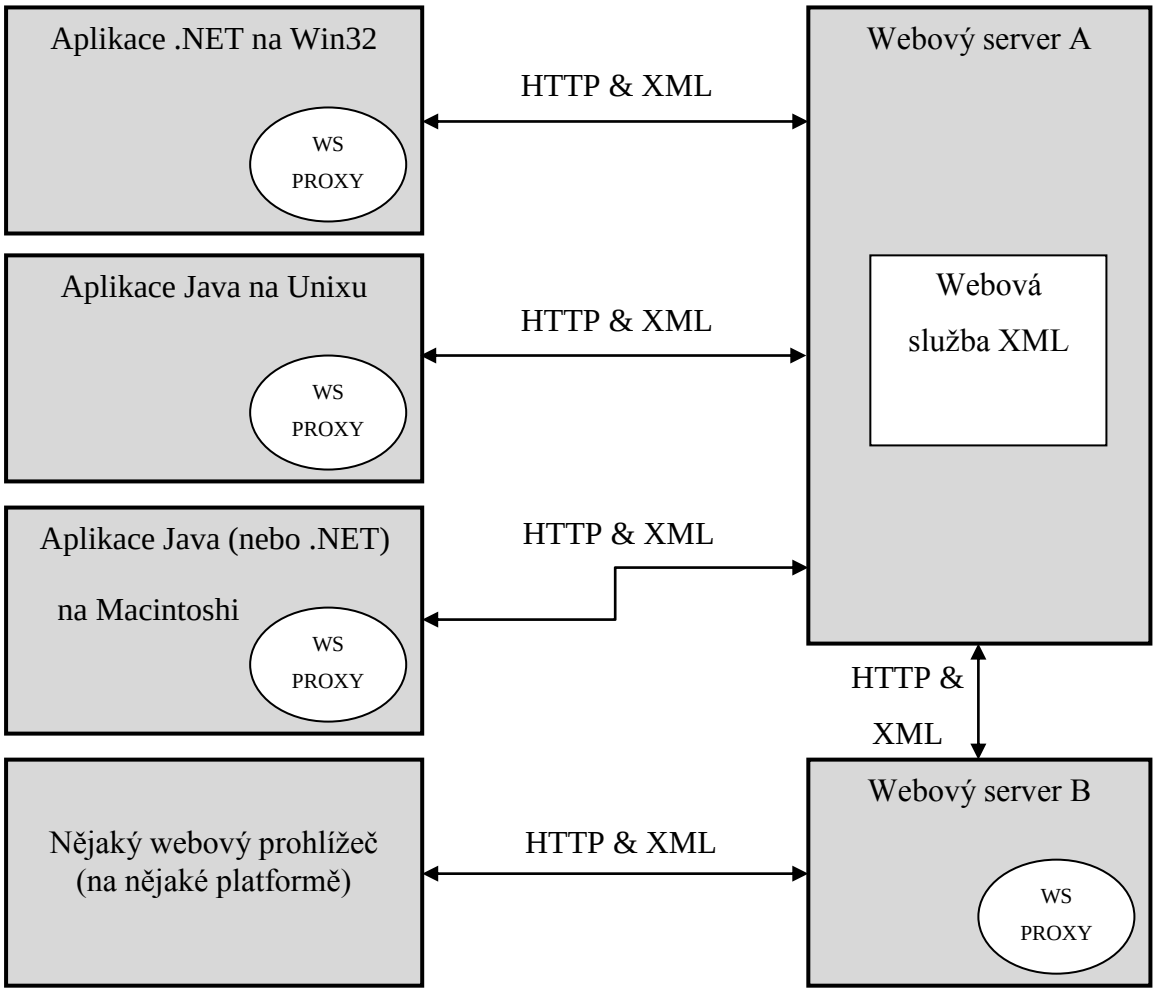
3.2 Co je to webová služba?

Webová služba je nová technologie, která má za cíl efektivně vyřešit problém s vkládáním objektů a komponent do webového prostředí. Webová služba je v zásadě jednotkou aplikační logiky, neboli jinak řečeno komponentou, která může být vzdáleně volána přes Internet. Díky tomu získáváme hotové aplikace, které sdílejí funkcionalitu a jsou aplikacemi modulárnějšími. Toto usnadní zejména práci vývojářům v .NET. Ti budou moct být schopni sdílet zkompilované assembly mezi jednotlivými aplikacemi. Dále budou moct sdílet funkcionalitu mezi aplikacemi, které běží na různých platformách a které jsou poskytovány různými společnostmi. [8]

3.3 Definice klienta webové služby

Jedním z aspektů webové služby, který nemusí být hned od počátku zřejmý, je to, že konzumentem webové služby nemusí být webová stránka. Žádné takové omezení tu není. Webovou službou mohou zrovna tak snadno využívat i klienti v podobě konzolových aplikací nebo formulářových aplikací Windows. Ve všech případech komunikuje konzument webové služby se vzdálenou webovou službou nepřímo, pomocí prostředníka, jímž je nějaký typ proxy serveru.

Proxy webové služby vypadá a chová se jako skutečný vzdálený objekt a vystavuje stejnou sadu členů. Pod kůrou webové služby najdeme ovšem implementační kód, který předává proxy požadavky dál webové službě pomocí standardního HTTP. Proxy také mapuje přicházející proud XML zpět do datových typů specifických pro .NET. [1]



Obr. 8. – Webová služba v akci [2]

3.4 Rozbor webové služby

```
<%@ WebService Language="C#" Class="WebService" %>

using System;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;

[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
// [System.Web.Script.Services.ScriptService]
public class WebService  : System.Web.Services.WebService {

    [WebMethod]
    public string AhojSvete() {
        return "Ahoj světe";

    }
```

Obr. 9. – Názorný příklad webové služby ASP.NET "Ahoj světe" napsaný v programovacím jazyku C#

3.4.1 Jmenné prostory

3.4.1.1 Úvod do jmenných prostorů

Pokud potřebujeme do XML ukládat nějaký druh informací, vytvoříme si pro tento účel obvykle vlastní značkovací jazyk, vlastní sadu značek. Formální zápis je za pomocí nějakého jazyka pro popis schématu dokumentu jako jsou DTD (Document Type Definition), XML schémata (XML Schema). Problém ovšem nastane, chceme-li v jednom dokumentu použít více sad značek. Může dojít například ke konfliktu názvů elementů apod. Tento problém řeší jmenné prostory. Jedná se o samostatný standard, který velmi úzce doplňuje samostatný standard XML. Dokonce se plánuje, že v dalším vydání standardu XML bude vše shrnuto v jednom dokumentu.

3.4.1.2 Princip jmenných prostorů

Každý element a atribut může být přiřazen ke jmennému prostoru, který je identifikován URI adresou. Pro zkrácení zápisu se pak v XML dokumentu deklarují pro použité jmenné prostory krátké prefixy, které se používají pro přiřazení elementu nebo atributu do určitého jmenného prostoru. Z následující ukázky je patrné, jak se zápis zkrátí, neboť poměrně dlouhé URI je v dokumentu uvedeno jen jednou. [11]

Ukázka jmenného prostoru:

```
<a:x xmlns:a="urn:x-pokus:a" xmlns:b="urn:x-pokus:b">
  <a:y>Obsah elementu Y ve jmenném prostoru urn:x-pokus:a</a:y>
  <b:y>Obsah elementu Y ve jmenném prostoru urn:x-pokus:b</b:y>
</a:x>
```

Obr. 10. – Zkrácený zápis zapříčiněn tím, že je URI uvedeno

v dokumentu pouze jednou

3.4.1.3 .NET zaměřené na webové služby

Tab. 1. – Tabulka jmenných prostorů webové služby [2]

Jmenný prostor	Význam
`System.Web.Services`	Obsahuje základní typy, které nutně potřebujete, když budujete nějakou webovou službu XML (včetně mimořádně důležitého atributu [WebMethod]).
`System.Web.Services.Configuration`	Obsahuje typy, které umožňují nakonfigurovat chování webové služby ASP.NET při běhu.
`System.Web.Services.Description`	Obsahuje typy, které umožňují programátorsky nakonfigurovat s dokumentem WSDL popisující danou webovou službu.
`System.Web.Services.Discovery`	Obsahuje typy, které umožňují webovému konzumentovi zjistit, jaké webové služby jsou nainstalované na daném stroji.
`System.Web.Services.Protocols`	Definuje četné typy, které reprezentují základ různých protokolů webové služby (HTTP GET, HTTP POST a SOAP)

3.4.2 Popis atributu [WebService]

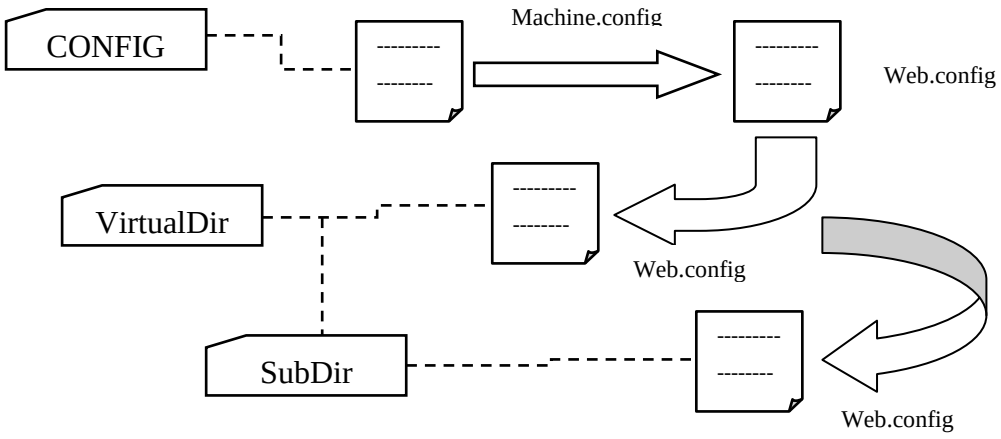
Webová služba může být volitelně kvalifikována atributem [WebService]. Tento atribut podporuje pár vlastností, jenž jedna z nich je Namespace. S její pomocí se dá zařídit název jmenného prostoru XML, který se má používat uvnitř WSDL. Runtime ASP.NET pro daný soubor *.asmx standardně požije jmenný prostor XML `http://tempuri.org/`. [1]

3.4.3 Soubor web.config

ASP.NET používá vícevrstvý konfigurační systém, který umožňuje používat pro různé části aplikace různá nastavení. Chcete-li tento systém používat, musíte přidat do svého virtuálního adresáře dodatečné podadresáře. Ty pak mohou obsahovat své vlastní konfigurační soubory web.config s dodatečnými nastaveními. ASP.NET používá dědění těchto konfigurací, takže každý podadresář obdrží nastavení z rodičovského adresáře. Dejme tomu, že máme webový požadavek http://localhost/A/B/C/Default.aspx, kde A je kořenový adresář webové aplikace. V tomto případě se do hry zapojují nastavení z několika úrovní:

1. Nejprve se použije nastavení z machine.config
2. Pak se aplikuje nastavení web.config z kořenového adresáře počítače. Tento soubor web.config je ve stejné složce CONFIG jako soubor machine.config.
3. Je-li nějaký soubor web.config v kořenu aplikace A, jeho nastavení se uplatní nyní.
4. Je-li nějaký soubor web.config v podadresáři B, jeho nastavení se uplatní nyní.
5. Je-li nějaký soubor web.config v podadresáři C, jeho nastavení se uplatní poslední.

Je důležité připomenout, že ačkoliv můžete mít neomezený počet podadresářů, jsou významná zejména nastavení použitá v krocích 1. a 2. Je to proto, že některá nastavení se dají použít pouze na úrovni machine.config a jiná nastavení lze zase použít na úrovni webové aplikace. [2]



Obr. 11. – Dědění konfigurace

3.5 Výhody webových služeb

Webové služby řeší problémy těsně spojených technologií, jako jsou například COBRA a DCOM (Distributed Component Object Model). Mluvíme zejména o těchto problémech:

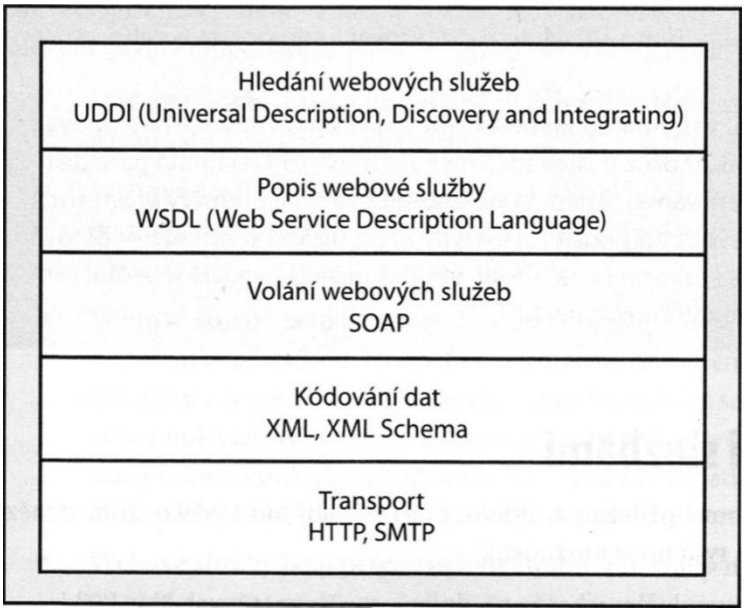
- Průchod přes firewall
- Zpracování složitých transportních protokolů nižší úrovně
- Integrace různorodých platforem

Zde nastupují právě webové služby, které tyto problémy řeší a představují další logický krok ve vývoji distribuovaných technologií založených na komponentách. Webové služby jsou nejsilnější zejména v následujících případech:

- **Jednoduchost** – webových služeb spočívá v tom, že mohou být snadno podporovány širokou škálou platforem.
- **Volnost metod** – webové služby mohou rozšířit svá rozhraní a přidat nové metody, aniž by to jakkoliv ovlivnilo klienty.
- **Bezstavovost** – znamená pro webové služby to, že klient vznese vůči webové službě požadavek, ta mu vrátí výsledek a spojení se ukončí. Neexistuje tedy permanentní spojení. To umožňuje se snadno přizpůsobit mnoha klientům a k poskytování webových služeb se využívá serverová farma
- **Přívětivost k firewall bráně** – pro webové služby to znamená, že ikdyž firewally mohou pro technologie distribuovaných objektů představovat problém a jediné co přes firewally může prakticky projít vždy, je http provoz na portech 80 a 443. Tohoto problému jsou, ale webové služby zcela zbaveny poněvadž zcela standardně používají http protokol. Mohou tedy bez problému procházet přes firewally, aniž by k tomu potřebovaly specifické nastavení firewallu. [1]

3.6 Standardy webových služeb

Webové služby jsou založeny na otevřených standardech. Za těmito standardy stojí významné společnosti, jako jsou například: Microsoft, IBM či Sun. Právě tyto společnosti musí, ale nejdříve všechny tyto standardy implementovat kompatibilním způsobem. [1]



Obr. 12. – Standardy, které dnes používají

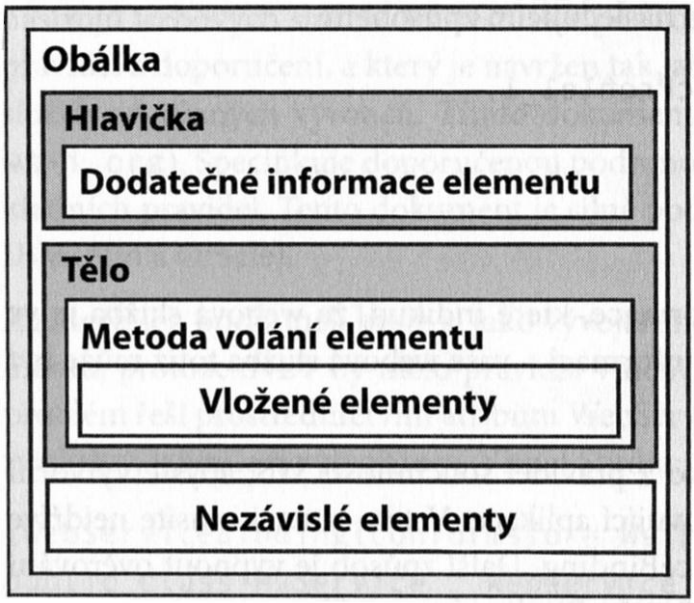
webové služby [1]

3.6.1 Co je to SOAP?

SOAP (Simple Object Access Protocol) je meziplatformový standard, který je používán pro formátování zprávy, zasílání mezi webovými službami a klientskými aplikacemi. Předností standardu SOAP spočívá v jeho flexibilitě. SOAP může využít nejenom pro zasílání jakýchkoliv typů XML dat, ale můžeme jej využít i pro práci s jinými transportními protokoly, než je nám známé klasické HTTP. Využití standardu SOAP najdeme i při zasílání zpráv přes přímé TCP/IP spojení. [1]

3.6.1.1 Struktura SOAP

Princip SOAP zprávy je jednoduchý. Jak již bylo řečeno SOAP zpráva je dokumentem XML. Tento XML dokument má jediný kořenový prvek <Envelope>, který slouží jako SOAP obálka. Zbytek zprávy je uložen uvnitř obálky, která v sobě obsahuje tělo a záhlaví. [1]



Obr. 13. – Struktura SOAP zprávy [1]

3.6.1.2 Verze SOAP

V dnešní době je nejpoužívanější verzí SOAP verze SOAP 1.1. Další použitelnou variantou je novější SOAP 1.2, které objasňuje mnoho aspektů SOAP standardů, zavádí několik menších vylepšení a formalizuje model rozšiřitelnosti. Bez ohledu na to, jakou verzi SOAP použijeme, jsou schopnosti webové služby stejné. V podstatě jediným důvodem, proč by jsme se měli zajímat o to, jakou verzi SOAP budeme používat je ta, abychom zajistili kompatibilitu s klienty, kteří .NET nepoužívají. [1]

3.6.2 Co je to UDDI?

UDDI (Universal Description, Discovery, and Integration) je to standard pro vytváření obchodních rejstříků, které registrují společnosti a webové služby, které nabízejí, a také odpovídají URL (Uniform Resource Locator) adresy jejich WSDL kontaktů. Tento UDDI centralizovaný adresář, kam různé společnosti vkládají webové služby, které následně nabízejí klientům. Tento potenciální klient vyhledá služby podle svých specifických potřeb. [1]

3.6.3 Co je to WSDL?

WSDL (Web Service Description Language) je jazyk založený na XML, který se používá k popisu veřejného rozhraní webové služby a komunikačních protokolů, které podporuje. WSDL dokumenty jsou v podstatě kontakty, který sděluje klientovi vše, co potřebuje

vědět, aby mohl spolupracovat s webovou službou. WSDL dokument má v podstatě stejnou úlohu jako typová knihovna u COM komponenty. Poněvadž obsahuje informace pro komunikaci mezi webovou službou a klientem. Neobsahuje žádné informace, které by jakkoliv se týkaly kódu nebo implementace metod webové služby. Tyto informace nejsou nezbytně nutné pro chod webové služby a mohly by narušit bezpečnost takovéto webové služby.

Síla WSDL spočívá v tom, že není svázán se žádnou specifickou platformou nebo objektovým modelem. Je to čistý jazyk XML, který poskytuje rozhraní pro webové služby v rámci všech platforem. [1]

3.6.3.1 Struktura WSDL

WSDL dokumenty se skládají z pěti hlavních elementů, které se vzájemně kombinují a popisují webovou službu. První tři jsou abstraktní a definují výměnu zpráv. Poslední dva jsou konkrétní, definují protokol a týkají se informací. Tři abstraktní elementy, `<types>`,`<message>` a `<portType>` se vzájemně kombinují a definují rozhraní webové služby. To znamená, že definují metody, parametry a vlastnosti webové služby. Zbývající dva elementy `<binding>` a `<service>`, se kombinují a poskytují nejenom protokol SOAP přes HTTP, ale také adresní informace URI (Uniform Resource Identifier) webové služby.

```
<?xml version="1.0" encoding="utf-8" ?>
<definitions>
  <types></types>
  <message></message>
  <portType></portType>
  <binding></binding>
  <service></service>
</definitions>
```

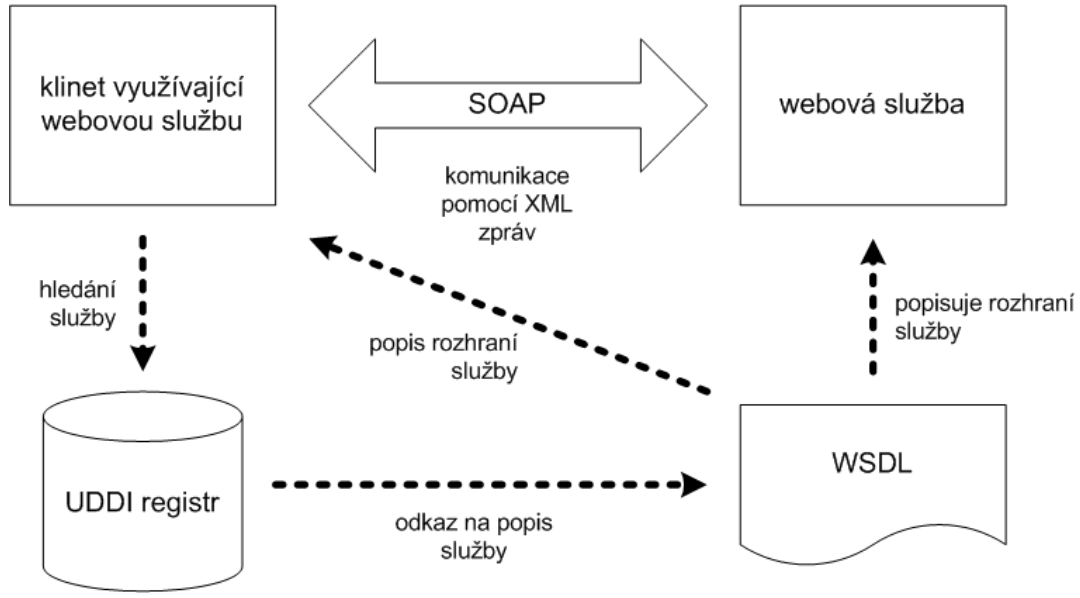
Obr. 14. – Elementy dokumentu WSDL

- **Types** – v této sekci jsou definovány všechny datové typy webové služby.
- **Message** – sekce, která poskytuje podrobné informace o zprávách, o požadavku a odpovědi, které jsou používány při komunikaci s webovou službou.
- **PortType** – sekce, kde se seskupují zprávy do dvojice vstupních a výstupních zpráv. Každá dvojice reprezentuje metodu.
- **Binding** – tato sekce poskytuje informace o transportních protokolech podporovaných webovou službou.
- **Service** – je sekce, která poskytuje koncové body (adresy URI) webové služby. [1]

3.6.4 Komunikace webových služeb

Jak již bylo řečeno, webové služby umožňují jednoduchou komunikaci mezi aplikacemi ve velmi heterogenním prostředí. Komunikace webových služeb je založena na platformě nezávislých standardech, a to především na jazyce XML a protokolu HTTP. Aplikace si mezi sebou posílají XML zprávy, které přenášejí dotazy a odpovědi jednotlivých aplikací.

Ke každé webové službě by měl být k dispozici její formální popis v jazyce WSDL. Z tohoto popisu již jde automaticky vygenerovat SOAP požadavek. Ve větších systémech nebo přímo v otevřeném prostředí Internetu se popis webové služby může zaregistrovat do UDDI registru. Ten slouží jako jakýsi telefonní seznam, který vyhledává služby s určitými parametry. Nakonec tedy klient, který chce využívat webovou službu, získá přístup k této službě buď přes UDDI, nebo přímo přes její popis. Z tohoto popisu je jasné, jakou strukturu má mít SOAP zpráva a kam se má webové službě poslat, aby ji rozpoznala. Vzájemné vztahy mezi standardy webové služby znázorňuje následující obrázek: [11]



Obr. 15. – Vztah tří základních technologií SOAP, WSDL a UDDI webových služeb [11]

3.6.5 Co je to HTTP?

Protokol HTTP (Hypertext Transfer Protocol) je základním protokolem na úrovni aplikace používaný ke zprostředkování přenosu dat „**na**“ a „**z**“ webového serveru. Prostřednictvím tohoto protokolu tedy probíhá veškerá komunikace webových služeb. Jak již bylo řečeno, využívá jej standard SOAP pro přenos zpráv přes HTTP kanály. HTTP poskytuje

jednoduchý a rychlý způsob definování interakce mezi klientem a serverem. Tento protokol definuje, jak klient musí žádat o data ze serveru a jak tyto data server klientu vrátí zpět.

3.6.5.1 Historie HTTP protokolu

První verze protokolu HTTP, známá jako verze 0.9, byla použita již roku 1990. HTTP verze 1.0 definovaná v RFC 1945 je podporovaná většinou serverů i klientů. HTTP 1.0 však nezpracovává správně efekty hierarchických proxy serverů ukládání do mezipaměti pro zprostředkování virtuálních hostitelů. Mimoto má HTTP 1.0 značné potíže s výkonem způsobené otevíráním a ukončováním velkého množství připojení pro jednu webovou stránku. Aktuální verze, HTTP 1.1, definovaná v RFC 2616 řeší mnoho z problémů minulých verzí protokolu. [15]

3.6.5.2 Princip HTTP protokolu

Uživatel nejdříve vyžádá dokument z webového serveru definováním adresy URL. Během tohoto kroku může proběhnout vyhledání názvu domény, které přeloží název, jako například: www.seznam.cz na adresu IP – 77.75.72.3. Pokud se vyhledávání názvu domény nezdaří, je vrácená chybná zpráva. Po nalezení serveru vytvoří prohlížeč správný požadavek HTTP a odešle jej na server, umístěném na adrese definované adresou URL, tedy již změněná adresa www.seznam.cz [2]

3.6.6 Co je to DISCO?

Jedná se o zkratku discovery. Tak zvaný discovery dokument poskytuje odkazy na více koncových bodů webové služby. DISCO standard je specifický pro Microsoft a posléze bude nahrazen podobným standardem pojmenovaným jako WS-Inspection. Standard DISCO vytváří jediný soubor, který seskupuje seznam příbuzných webových služeb. Společnost může na svém serveru zveřejnit soubor DISCO, který obsahuje odkazy na všechny poskytované webové služby. Klienti, kteří chtějí najít všechny dostupné webové služby, musí o tento soubor pouze požádat. Tento postup je užitečný, když klient už zná společnost, která webové služby nabízí a chce zjistit, které služby jsou zpřístupněny, a také najít odkazy k podrobnějším informacím o těchto službách. Vyhledávání webových služeb přes Internet není sice moc efektivní, nicméně se to může hodit u lokálních sítí, kde klient připojí na server a hned vidí, které služby jsou k dispozici. [2]

3.7 Zabezpečení webových služeb

Webové služby by mohly v ideálním světě pracovat zcela nechráněny, jelikož by se pojaly jako knihovna tříd s určitými schopnostmi, kde by jsme se nestarali o zajištění autentizace uživatelů či s logikou zabezpečení. Jelikož žijeme v realitě a webové služby se umisťují na Internet, kde se střetávají s neustálým útokem zvenčí, musí se zabezpečení věnovat větší pozornost. Dále jsou také webové služby, které jsou založeny na přihlašování společně s využitím mikroplateb, a kde se musí ochránit citlivá data klienta.

K ochraně webových služeb můžeme použít některou z technik používaných k ochraně webových stránek. Jako například IIS s podporou SSL, kde své klienty nasměruje na adresu webové služby, která začíná předponou https://. IIS můžeme rovněž použít pro autentizaci Windows, ačkoliv je pravda, že musíme provést ještě několik dalších kroků.[1]

3.7.1 IIS – Internet Information Services

3.7.1.1 Základní informace

IIS je v podstatě služba Windows. Má na starosti zpracování požadavků, které obdrží na konkrétních portech. K tomuto účelu běží v systému služba nazvaná World Wide Web Publishing Service v překladu tedy, Služba publikovaná na WWW. Tato služba naslouchá na několika síťových portech TCP/IP, obvykle na portu 80 pro normální HTTP a na portu 443 pro HTTPS. Tuto službu zajišťuje konzola IIS, která vytváří různé weby. Pro každý web může zaregistrovat více portů, nutný je však minimálně jeden. [1]

3.7.1.2 Bezpečnost IIS

Než se k runtime ASP.NET dostane přístupový požadavek, IIS ověřuje zabezpečení na základě vlastní konfigurace. IIS poskytuje několik bezpečnostních mechanismů, které fungují jako strážní, ještě předtím než ASP.NET začne obsluhovat požadavek. Mluvíme v zásadě o těchto bezpečnostních mechanismů:

- **Autentizace** – IIS podporuje základní autentizaci, digestní (souhrnnou) autentizaci, pasportovou autentizaci, autentizaci Windows, a také autentizaci pomocí certifikátu prostřednictvím kanálu SSL. Výsledkem jakékoliv autentizaci, kterou vykonává ISS, je autentizovaný uživatel Windows. IIS proto podporuje autentizaci uživatelů Windows.

- **Autorizace** – IIS má integrovanou podporu pro omezení IP adres a vyhodnocení ACL Windows.
- **Důvěrnost** – Šifrování může být vynuceno pomocí SSL. [1]

3.7.2 SSL – Secure Socket Layer

SSL (Secure Socket Layer) se k zabezpečení webových služeb nabízí jako šifrovací protokol. Jeho využití ve webových službách je však z následujících důvodu nevhodný:

- SSL je navržen tak, aby zabezpečoval spojení typu bod-bod. To nestačí pro webové služby, neboť u nich je potřeba zabezpečit komunikaci mezi koncovými účastníky, mezi kterými muže existovat několik uzlů.
- SSL pracuje na transportní vrstvě a ne na úrovni zprávy. Data jsou tedy chráněna, když putují po drátě, ale ne na pevném disku.
- HTTPS v současné podobě nepodporuje nepopiratelnost. Ta je důležitá zvláště u komerčních webových služeb.
- SSL neumožňuje podepisování a šifrování elementu v XML dokumentu.[2]

3.8 Řešení zabezpečení

3.8.1 Bezpečnost díky XML

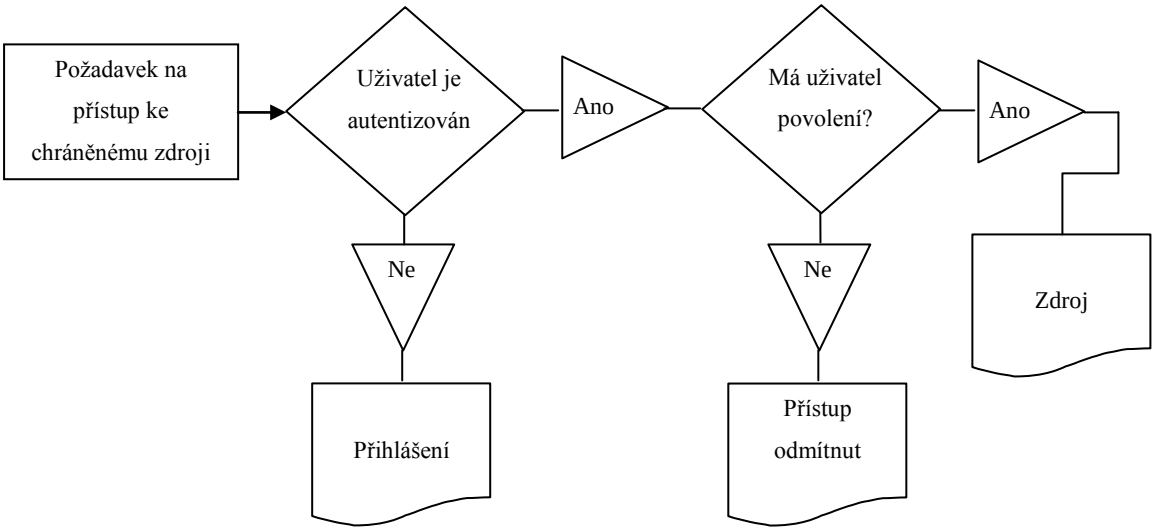
Bezpečnost při výměně zpráv typicky představuje autenticitu, integritu dat, nepopiratelnost a utajení. Digitální podpis zajišťuje první tři požadavky. Čtvrtý zajišťuje šifrování dat. W3C (World Wide Web Consortium) definovalo specifikace XML Signature a XML Encryption pro zabezpečení komunikace založené na XML. Kromě toho společnosti IBM, Microsoft a VeriSign společně vytvořili doplňující specifikace jako WS-Security (Web Services Security), které jsou postaveny na těchto specifikacích W3C. Celkem bylo v průběhu posledních let vytvořeno několik bezpečnostních schémat založených na XML, aby poskytly úplné a jednotné bezpečnostní schémata pro webové služby. Tyto schémata zahrnují:

- XML digital signature
- XML Encryption
- WS-Security (Web Services Security)
- SAML (Secure Assertion Markup Language)
- XKMS (XML Key Management Specification)

- XACML (Extensible Access Control Markup Language)
- ebXML Message Service [10]

3.8.2 Autentizace Windows

Autentizace Windows je dobrým řešením pro webové služby, pokud máte malý okruh uživatelů, kteří již mají účty Windows. Princip této autentizace je jednoduchý. Autentizace Windows funguje velmi podobně jako autentizace u webové stránky. Rozdíl je v tom, že webová služba je vykonává jinou aplikací, nikoliv přímo prohlížečem. Z toho důvodu zde není k dispozici žádný vestavěný způsob, jak vyzvat uživatele, aby zadal své uživatelské jméno a heslo. Tyto informace doplňuje již samotná aplikace webové služby, kde si tyto informace načte buď z konfiguračního souboru, z databáze, nebo vyzve přímo uživatele k jejímu doplnění. Nevýhoda této autentizace je ta, že už tak dobře nefunguje u rozsáhlých veřejných webových služeb. Řešením je navržení vlastní autentizace. [1]



Obr. 16. – Žádost o přístup k webové stránce vyžadující autentizaci a autorizaci [1]

3.8.3 Vlastní autentizace založena na lístcích

Správně navržený systém založený na autentizaci pomoci lístků má řadu výhod. Poskytuje absolutní flexibilitu, optimalizuje také výkon a zaručuje rozšiřitelnost, protože je možné lístek cachovat. Princip této metody spočívá v tom, že uživatel si musí zavolat specifickou webovou metodu webové služby a poté musí dodat své přihlašovací údaje. Jako je kombinace uživatelského jména a hesla. Metoda pro přihlášení zaregistruje uživatelskou relaci a vytvoří nový, jedinečný lístek. Od této doby se uživatel může opakovaně připojovat k webové službě tím, že jakékoliv další metodě předloží tento lístek. Díky této

metodě využijeme výhod standardu SOAP záhlaví, které zajišťuje, aby proces řízení lístků byl přehledný, stejně jako autorizace klienta.[1]

II. PRAKTICKÁ ČÁST

4 REALIZACE WEBU

Součástí zadání bakalářské práce je i názorná ukázka webové služby. Aby takováto služba mohla být realizovatelná, musely se nejdříve navrhnout webové stránky, na kterých by tato služba mohla běžet. Pro realizaci webu byly využity programy Adobe Photoshop CS4 a Visual Studio.

4.1 Grafický návrh

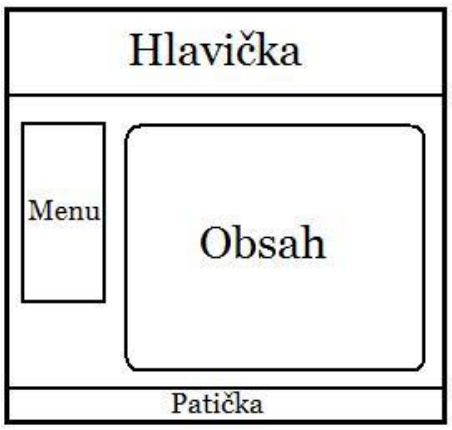
V současné době i malé internetové stránky mají za cíl vypadat co nejlépe. S tím souvisí design stránek. Jelikož nejsem grafik, grafický návrh byl problematický úkol. Nabízela se možnost jít cestou kaskádových stylů a celé stránky nastylovat, nebo se pokusit zrealizovat malý web, který by měl trochu grafického cítění. Nakonec byla zvolena cesta grafického návrhu, který se implementoval do stránek. Pro vypracování grafického návrhu byl využit program Adobe Photoshop CS4.

4.1.1 Využití Adobe Photoshopu CS4

Adobe Photoshop CS4 byl zvolen z toho důvodu, poněvadž byl uživatelsky nejpřívětivější. Zejména protože již při spuštění se uživateli nabízí, pracovní plocha aplikace Adobe Photoshop CS4. Je uspořádaná tak, aby vám pomohla soustředit se na vytváření a úpravy obrazů, grafických návrhů webů atd. Pracovní plocha obsahuje nabídky a řadu nástrojů a panelů pro prohlížení, úpravy a přidávání prvků do obrazů.

4.1.2 Rozvržení stránky

Rozvržení stránky bylo nejdříve zpracováno čistě na papír. Navrženo bylo několik variant, jak by budoucí návrh stránky mohl vypadat. Nakonec byla vybrána tato:



Obr. 17. – Rozvržení webové stánky

Vycházelo se z toho, že budoucí stránky mají mít informativní charakter neboli prezentační. V takových to případech někdy málo znamená více. A proto zvítězila varianta, která je rozdělena do tří sekcí **hlavička**, **obsah** a **patička**. Vysvětlení následujících sekcí:

- **Hlavička** – je zde umístěn název firmy, logo firmy a čeho se firma týká
- **Obsah** – v této sekci najdete tlačítkové menu a obsah samotný
- **Patička** – je prezentací autora stránek, kdy byly vytvořeny a kontakt na autora

4.1.3 Výsledný vzhled stánky před implementací

Na názorném obrázku se Vám má přiblížit, jak vypadá finální verze grafického návrhu. Před jeho implementací do ASP.NET. Tedy předtím než se tento návrh rozřezal do zmíněných sekcí.

Obr. 18. – Výsledný grafický návrh stránek

4.2 Implementace grafického návrhu do ASP.NET

Proto, abychom mohli tento grafický návrh vložit do stránky *.aspx budeme k tomu potřebovat ještě znalost kaskádových stylů a také znalost prostředí ASP.NET.

Na stránce byl vytvořen jeden hlavní **div** s id **stranka**. Jeho šířka je 770 pixel, aby mohl být zobrazen i na obrazovce s menším rozlišením než je 1024x768px. Do tohoto divu se umístily další 3 **div**y - **hlavička**, **obsah** a **patička**. Hlavička obsahuje pouze název firmy, její logo s popiskem. Do obsahu byla vložena komponenta **ContentPlaceHolder**, místo kde se budou dosazovat jednotlivé stránky. A dále se do obsahu vložil id **menu.** V patičce se nachází copyright a informace o autorovi.

4.2.1 Master page

Master page slouží pouze jako kostra stránek. Tento soubor MasterPage.master se nedá zobrazit v prohlížeči. Obsahuje tedy jen kostru stránek a zmíněnou komponentu **ContentPlaceHolder,** která má svou id **ContentPlaceHolder1**, do které se odkazují jednotlivé obsahy s **Content** ostatních stránek *.aspx.

Master page je vlastně jakousi šablonou určující vzhled stránek, jež ji používají. Typická master stránka definuje hlavičku, patičku a levý pruh webové stránky. Stránky, jež tento master používají, pak zpravidla doplňují „vnitřek“ stránky k vnějším oblastem definovaných masterem.

Obr. 19. – Rozdělení stránky MasterPage.master

```
<%@ Master Language="C#" AutoEventWireup="true"
CodeBehind="MasterPage.Master.cs" Inherits="Vistamont.MasterPage"
%>

<body>
    <form id="form1" runat="server">
     <div id="stranka">
     <!-- Hlavička -->
        <div id="hlavicka">
        </div>
     <!-- Obsah -->
      <div id="Obsah">
         <div id="innerContent">
            <asp:ContentPlaceHolder ID="ContentPlaceHolder1"
                  runat="server">
            </asp:ContentPlaceHolder>
         </div>
         ...
      </div>
     <!-- Patička -->
        <div id="Paticka"> ... </div>
     </div>
    </form>
</body>
```

Obr. 20. – Výtažek kódu, ze stránky MasterPage.master

Z tohoto obrázku je patrné, že jednotlivé divy či oblasti definované na masteru jsou aktivní, se kterými můžeme na stránce pracovat tak, jak jsme zvyklí – vytvářet zde ovládací prvky a podobně. Všimněte si, že komponente **ContentPlaceHolder1** je prázdná. To je z toho důvodu, že do tento obsah je dynamický. Vkládá se do něj obsahy z jednotlivých Contentu a tím pádem je buď vyšší, nebo nižší.

4.2.2 Content page

Content page má jediný úkol, a to ten, že veškerý kód, který je obsažen v komponentě Content se zobrazí v **ContentPlaceHolder1**, která je umístěna v souboru MasterPage.master. **ContentPlaceHolder1** je speciální prvek, který bude vyplněn skutečným obsahem na stránkách používajících příslušný master (na každé master stránce může být i více pojmenovaných prvků typu ContentPlaceHolder).

```
<%@ Page Title="O nás" Language="C#"
MasterPageFile="~/MasterPage.Master" AutoEventWireup="true"
CodeBehind="Onas.aspx.cs" Inherits="Vistamont.WebForm2" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head"
runat="server">
</asp:Content>
<asp:Content ID="Content2"
ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
    <div id="Posun"> ... </div>
</asp:Content>
```

Obr. 21. – Content page s ContentPlaceHoldrem, ve kterém je umístěn obsah stránky

Obr. 22. – ContentPlaceHolder s vyplněným obsahem

Vidíme, že oblasti definované na masteru jsou zašedlé a nemůžeme je upravovat. Naproti tomu ContentPlaceHolder označuje oblast, se kterou můžeme na stránce pracovat tak, jak jsme zvyklí – vytvářet zde ovládací prvky a podobně. A zde je patrné, že v ContentPalceHolder1 je umístěný obsah z Contetu.

4.2.3 Tvorba menu

Menu bylo vytvořeno v soboru MasterPage.master díky skriptu vloženého v tabulce. Nebyla využita komponenta SiteMapPath, díky němuž se velmi jednoduše menu realizuje. Díky programu Adobe Photoshop CS4 byla vytvořena tlačítka. Které se pak pomocí

tabulky a jednoduchého skriptu ne straně serveru rozhýbala. Jejich princip je velice jednoduchý, je to klasická záměna *.jpg obrázků pomocí skriptu při najetí myši na dané tlačítko.

```
<a href="Prace.aspx" onmouseover="document['Prace'].src =
'App_Themes/Default/Picture/Button/Tl_PraceSt.jpg' ;"
onmouseout="document['Prace'].src =
'App_Themes/Default/Picture/Button/Tl_Prace.jpg' ;">
<img src="App_Themes/Default/Picture/Button/Tl_Prace.jpg"/
name="Prace">
</a>
```

Obr. 23. – Výtah kódu, pro záměnu tlačíteka **Práce**

Obr. 24. – Menu v základním stavu před použitím události pro změnu obrázku tlačítka

Obr. 25. – Menu ve změněném stavu při najetí ukazatelem myši na tlačítka

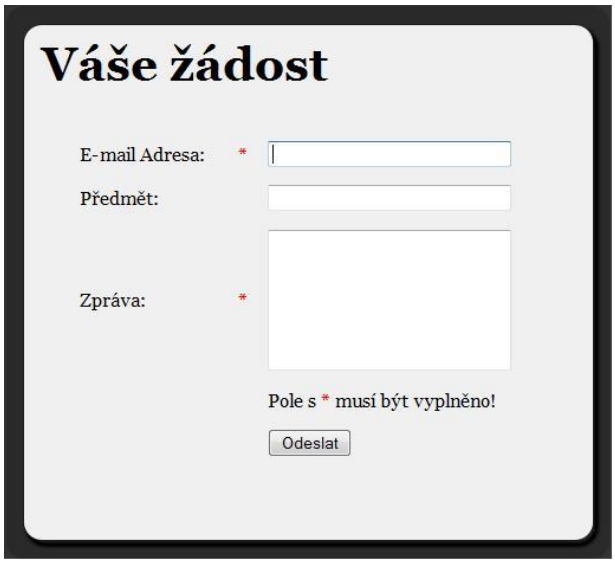
5 IMPLEMENTACE WEBOVÉ SLUŽBY

Pro nejefektivnější ukázku webové služby byla zvolena služba **ValidateEmail**, která má za úkol ověřit správnost zadané e-mail adresy, kterou uživatel zadá do formuláře pro odeslání e-mailu. Tato služba se na webových stránkách využívá ve velkém množství.

5.1 Realizace stránky EmailBox

Při realizaci stránky pro odesílání e-mailu na nastavený adresátovy e-mail bylo využito opět contentu. Kde do contentu se vložila tabulka a do ní se umístily prvky:

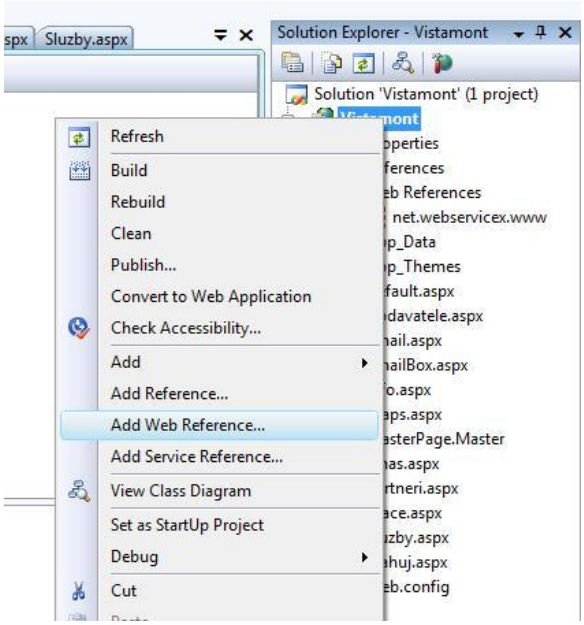
- **TextBox** – v tabulce jsou umístěny 3 prvky TextBox (txtEmail, txtPredmet a txtZprava) s níž prvek **txtZprava** má nastavenou vlastnost **TextMode** na **MultiLine**, proto aby při psaní zprávy mohl využívat víceřádkový řetězec v tomto TextBoxu.
- **ValidationSummary –** tento prvek kontroluje TextBoxy - txtEmail a txtZprava proto, aby uživatel musel vložit správně zadaný e-mail a také proto, aby se neodeslala prázdná zpráva.
- **Label** – do tohoto prvku se vypisují, informativní zprávy, zda byl e-mail odeslán, zda se odesílání e-mailu nepodařilo či zda uživatel vložil neověřenou e-mailovou adresu
- **Button** – tento prvek je nejdůležitější, v tomto jsou nastaveny veškeré události obsluhující tuto webovou službu



Obr. 26. – Výsledek e-mailového formuláře

5.2 Využití webové služby ValidateEmail

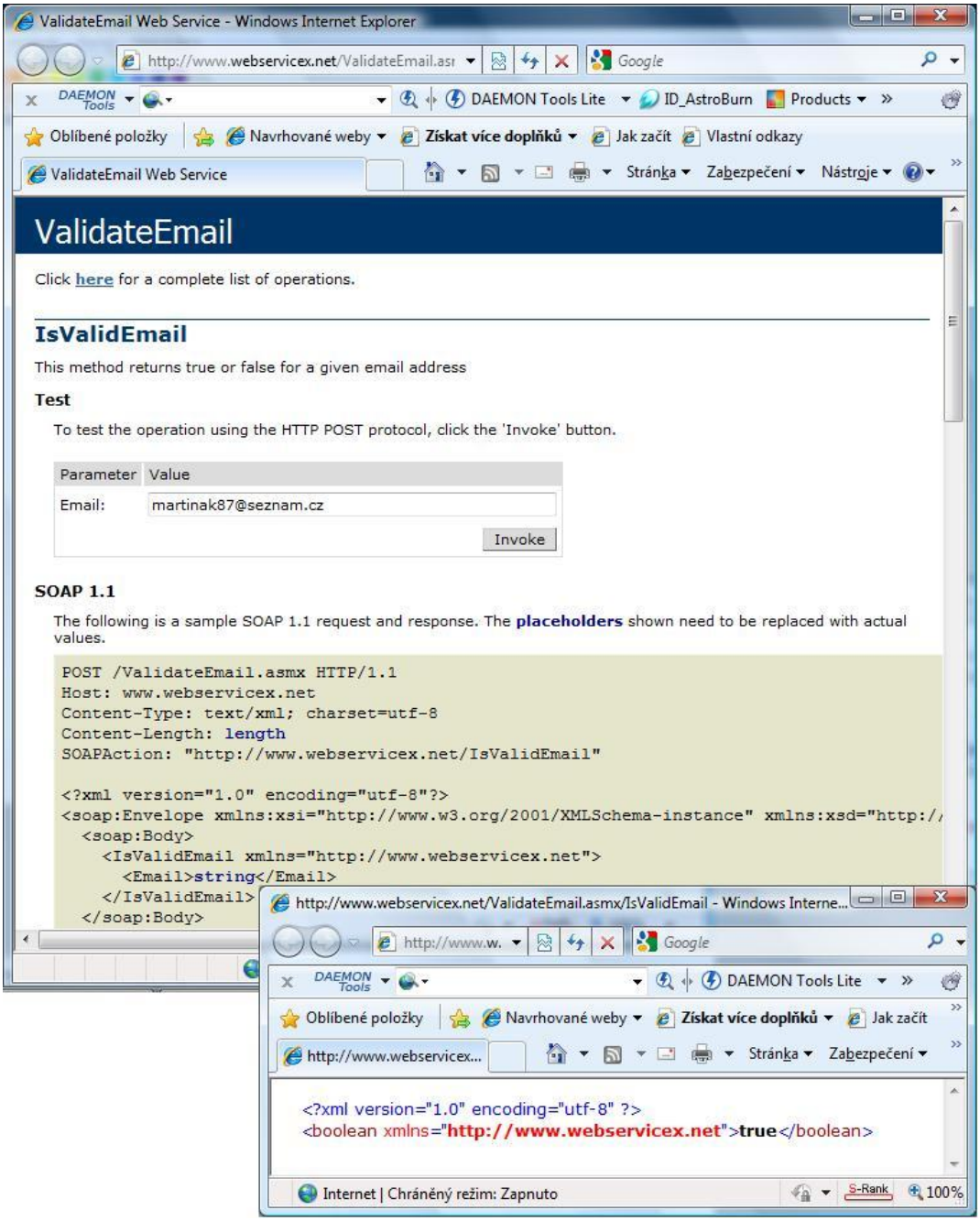
Tato webová službu byla použita a implementována za pomocí funkce „**Add web reference**“ z adresy **http://www.webservicex.net/ValidateEmail.asmx** na níž je umístěná a kterou následně využívá, pro ověřování e-mailové adresy.



Obr. 27. – Přidání webové reference do projektu

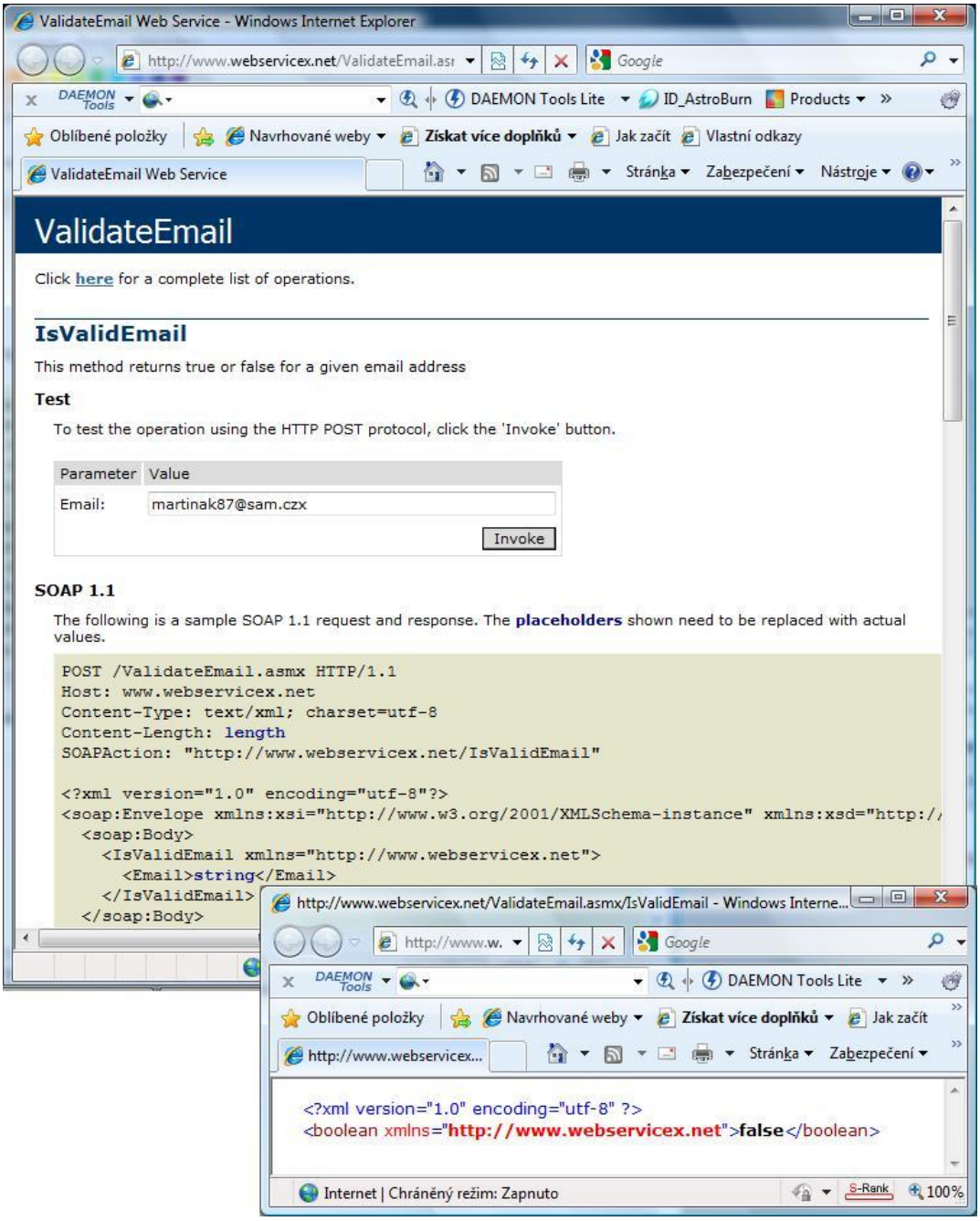
5.2.1 Práce webové služby ValidateEmail

Na následních obrázcích je názorně ukázáno jak webová služba pracuje na adrese http://www.webservicex.net/ValidateEmail.asmx. Při vyhodnocování zadané e-mail adresy a jaká výsledky tato webová služba zpětně vrací.



Obr. 28. – Kontrola webové služby ValidateEmail s pravdivým výsledkem

Na obrázku výše je zachycena kontrola e-mailové adresy s pravdivým výsledkem. Tedy když uživatel zadá svou e-mail adresu, tak webová služba **ValidateEmail** tuto adresu zkontroluje a vyhodnotí s výsledkem true nebo false. Zde vidíte, že e-mail adresa **martinak87@seznam.cz** je vyhodnocena jak správná tedy s boolean hodnotou **true**.



Obr. 29. – Kontrola webové služby ValidateEmail s nepravdivým výsledkem

Na obrázku výše je zachycena kontrola e-mailové adresy s nepravdivým výsledkem. Tedy když uživatel zadá svou e-mail adresu, tak webová služba **ValidateEmail** tuto adresu zkontroluje a vyhodnotí s výsledkem true nebo false. Zde vidíte, že e-mail adresa **martinak87@sam.czx** je vyhodnocena jako nepravdivá, tedy s boolean hodnotou **false**.

5.3 Nastavení události tlačítka odeslat

Nastavení události tlačítka se muselo umístit do souboru EmailBox.aspx.cs. Jmenné prostory **using System.Net.Mail; using System.Drawing;** a **using Vistamont.net.webservicex.www;** Poté bylo zapotřebí nastavit hlavně události tlačítka při načtení e-mailové adresy z TextBoxu txtEmail, načtení řetězce z TextBoxu txtZprava a následné předání výsledku pro danou službu. Následně se musely nastavit ošetření výjimek, které můžou nastat, když se nezadá email nebo, když se nevloží textový řetězec do txtZprava.

```
MailMessage EmailMessage = new MailMessage();
EmailMessage.From = new MailAddress(txtEmail.Text);
EmailMessage.To.Add(new MailAddress("martinak87@seznam.cz"));
EmailMessage.Subject = "Zprava Vistamont: " + txtPredmet.Text;
EmailMessage.Body = txtZprava.Text;
EmailMessage.IsBodyHtml = false;
EmailMessage.Priority = MailPriority.Normal;
SmtpClient client = new SmtpClient();
client.Send(EmailMessage);
lblZprava.Text = "Zpráva odeslána úspěšně.";
lblZprava.ForeColor = Color.Green;
```

Obr. 30. – Události, které nastanou při stisku tlačítka Odeslat

5.4 Nastavení souboru Web.config

V souboru web.config bylo zapotřebí nastavit vlastnosti SMTP serveru, aby bylo možné přijímat e-maily, které uživatel odešle z webové stránky. Proto je nutné nakonfigurovat SMTP server pro aplikace ASP.NET prostřednictvím administračního webu, aby se do souboru web.config dané aplikace přidaly nezbytné konfigurační položky.

```
  <system.net>
    <mailSettings>
      <smtp from="VistamontSolar@seznam.cz">
        <network host="smtp.seznam.cz"
userName="VistamontSolar@seznam.cz" password="*************"
port="25" />
      </smtp>
    </mailSettings>
  </system.net>
```

Obr. 31. – Nastavení SMTP serveru v souboru web.config

6 VYHODNOCENÍ POZNATKŮ

Webové služby patří v současné době k potenciálně novým technologiím. Ve světe webových stránek, webových služeb a aplikací má již neodmyslitelné místo. Z důvodu toho, že využívá standardní webové protokoly – XML, HTTP a TCP/IP, je velmi jednoduše implementovatelné a také programátorsky přívětivá.

Mluvíme tu o nové technologii, jenž vyřešila problém vkládání objektů a komponent do webového prostředí. Vývojáři ji mohou volat pomocí protokolu HTTP přes internet, díky tomu získávají hotové aplikace, které sdílejí funkcionalitu a jsou modulárně přijatelnější. Programátor tuto službu využije vzdáleně. Tedy rychle a efektivně zpracuje zadaný úkol. Toto usnadní zejména práci vývojářům v .NET. Ti budou moct být schopni sdílet zkompilované assembly mezi jednotlivými aplikacemi. Dále budou moct sdílet funkcionalitu mezi aplikacemi, které běží na různých platformách a které jsou poskytovány různými společnostmi.

Bezpečnost webové služby stojí také za zmínku. Jelikož pracují s citlivými informacemi ohledně klienta, který platí kartou v internetovém obchodu, hrozí takovému to klientu z vnějšku nebezpečí. Na to, ale webové služby myslely a efektivně si vypomohli opět za pomocí XML souborů. Bezpečnost při výměně dat se zabezpečuje hlavně autenticita, integrita dat, nepopiratelnost či utajení. Za pomocí XML, která využívá digitálního podpisu ochrání první tři zmíněné požadavky. Čtvrtému požadavku věnovala pozornost společnost World Wide Web Consortium a definovalo specifikace XML Signature a XML Encryption pro zabezpečení komunikace založené na XML. Kromě toho společnosti IBM, Microsoft a VeriSign společně vytvořili doplňující specifikace jako WS-Security (Web Services Security), které jsou postaveny na těchto specifikacích W3C.

Výhody webových služeb je hned několik. Ne jenže řeší problémy těsně spojených technologií, jako jsou například COBRA a DCOM. Kde byl hlavním problémem průchod webové služby přes firewall, zpracování složitých transportních protokolů nižší úrovně a také integrace různorodých platforem.

Toto všechno webové služby velmi prakticky zvládají za spolupráce standardů SOAP, WSDL a UDDI. Ale také to, že jsou využívané jako služby, do nichž lze přidávat objekty, a tím vyhotovíme plně dynamickou aplikaci.

Hlavně díky těmto aspektům, se staly webové služby tak oblíbenými nástroji pro vývojáře webových stránek. Díky nimž se vyvarují zdlouhavým programováním a odstraňováním výjimek, které brzdí vývojáře v další práci.

ZÁVĚR

Bakalářská práci je zaměřena na informace týkající se problematiky s webovou službou. Práce je rozdělena do dvou bloků a to s teoretickou a praktickou částí. V každém bloku je několik okruhů pro danou problematiku, týkající se webových služeb.

Rozdělení bakalářské práce:

- **Teoretická část**
  - Historický vývoj
  - Programové technologie
  - Webová služba
- **Praktická část**
  - Realizace webu
  - Názorná ukázka webové služby
  - Vyhodnocení výsledků

Teoretická část je zaměřena nejenom na webové služby samotné, ale také na věci jím blízké. Zabývá se historickým vývojem Internetu, webovým prostředím a službám. Dále popisuje pomocí jakých technologii, lze webové služby napsat tedy naprogramovat. A v závěru teoretické části se zaměřuje již na webové služby podrobně. Vysvětluje jejich přínos pro webové stránky, jaké standardy využívají, její princip při práci na Internetu. Také se zmiňuje o XML souboru, s kterým je provázán a to doslova s webovou službou. Tato práce se snaží zachytit co nejpodrobnější hlavní aspekty webové služby. Také ale se věnuje formulaci, jak tuto problematiku co nejjednodušeji vysvětlit.

V praktické části se zmiňuje o realizaci webu, na kterém se názorně ukazuje webová službu. Bylo využito výhod grafických šablon. Jedna byla zpracována v programu Adobe Photoshop CS4. Tato šablona se následně rozdělila na tři hlavní části a za pomoci ASP.NET se implementovala do webových stránek. Následně na to bylo využito webové služby http://www.webservicex.net/ValidateEmail.asmx, pro kontrolu odesílání e-mailu z webových stránek. Závěr je věnován vyhodnocení získaných poznatků o webových službách.

ZÁVĚR V ANGLIČTINĚ

This bachelor thesis deals with information about the Web service. Thesis consists of two blocks, theoretical and practical parts. Several topics are described in two blocks.

The consists of two blocks:

- **The first theoretical part consists of**
  - Historical development
  - Software Technologies
  - Web Service
- **The second practical part consists of**
  - Implemented Web
  - Example of web services
  - Interpretation of results

The theoretical part is focused on not only the Web service itself, but also on things close to them. The thesis described historical development of the Internet, the Web environment, services.

Programming languages which you can write a Web service is the described thesis. In the end of this thesis Web service is the discuss thoroughly, and the mention of neither XML file is not forgot.

The practical part is worked on Web site, where is implementation web service. I known advantages of graphical templates, and one am prepared on Adobe Photoshop CS4. This template was subsequently consists of three main sections and using implanted into ASP.NET Web pages.

The thesis inserted on Web service from http://www.webservicex.net/ValidateEmail.asmx into web pages. In conclusion, is the conducted on an evaluation of knowledge gained about Web services.

SEZNAM POUŽITÉ LITERATURY

[1] MACDONALD, Matthew, SZPUSZTA, Mario. ASP.NET 3.5 a C-Sharp 2008 : Tvorba dynamických stránek profesionálně. Jan Pokorný, Jan Gregor. 1. vyd. [s.l.] : Zoner Press, 2008. 1584 s. ISBN 978-80-7413-008-3.

[2] MACDONALD, Andrew Troelsen. C# a .NET 2.0 profesionálně. Jan Pokorný. 1. vyd. [s.l.] : Zoner Press, 2006. 1200 s. ISBN 80-86815-42-0.

[3] EVJEN, Bill, HANSELMAN, Scott, RADER, Devon. ASP.NET 3.5 v jazycích C-Sharp a Visual Basic. [s.l.] : Computer Press, 2009. 1600 s. ISBN 978-80-251-2069-9.

[4] BILL, Evjen, et al. ASP.NET 2.0 : Programujeme profesionálně. Karel Voráček. Brno : Computer Press, a.s., 2006. 1224 s. ISBN 978-80-251-1473-5.

[5] Illustrated C-Sharp 2008 . [s.l.] : [s.n.], 2009. 1333 s.

[6] Programming .NET.3.5. [s.l.] : [s.n.], 2008. 478 s.

[7] Webové programování v ASP.NET 2.0 . [s.l.] : [s.n.], 2008. 648 s.

[8] Webové služby. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 11. 11. 2006, last modified on 25. 3. 2010 [cit. 2010-05-26]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Webov%C3%A1_slu%C5%BEba>.

[9] *Zaměřeno na informační technologie* [online]. 06. 08. 2005 [cit. 2010-05-26]. Programute.com. Dostupné z WWW: <http://programujte.com/?akce=clanek&cl=2005081704-zaklady-xml-webovych-sluzeb>.

[10] Extensible Markup Language. In Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, 2. 12. 2007, last modified on 4. 5. 2010 [cit. 2010-05-26]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Extensible_Markup_Language>.

[11] KOSEK, Jiří . *Vše, co jste kdy chtěli vědět, ale báli jste se zeptat* [online]. 2006 [cit. 2010-05-26]. Téměř vše o www.kosek.cz. Dostupné z WWW: <http://www.kosek.cz/diplomka/html/websluzby.html>.

[12] *O tvorbě, údržbě a zlepsování internetových stránek* [online]. 2006 [cit. 2010-05-31]. Jakpsatweb. Dostupné z WWW: <http://www.jakpsatweb.cz/css/css-prakticky.html>.

[13] *HyperText Markup Language. In Wikipedia* : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, 16.8.2009, last modified on 22.5.2010 [cit. 2010-05-31].

Dostupné z WWW:

<http://cs.wikipedia.org/wiki/HyperText_Markup_Language>.

[14] *Visual Basic. In* Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, 19.2.2006, last modified on 31.3.2010 [cit. 2010-05-31]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Visual_Basic>.

[15] *HTTPS. In Wikipedia : the free encyclopedia [online]*. St. Petersburg (Florida) : Wikipedia Foundation, 2.1.2006, last modified on 15.3.2010 [cit. 2010-05-31]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/HTTPS>.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

API	Application Programming Interface
ASP	Active Server Pages
BCL	Based Class Library
CERN	Conseil Européen pour la recherche nucléaire
CLR	Common Language Runtime
COBRA	Common Object Request Broker Architecture
COM	Component Object Model
CSS	Cascading Sytle Sheets
CSS1	Cascading Sytle Sheets, Level 1
CSS2	Cascading Sytle Sheets, Level 2
CSS 2.1	Cascading Sytle Sheets Level 2 Revision 1
CSS3	Cascading Sytle Sheets, Level 3
C#	C Sharp
DCOM	Distributed Component Object Model
DTD	Document Type Definition
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IIS	Internet Information Services
LINQ	Language Integrated Query
PHP	Hypertext Preprocessor
RAD	Rapid Application Development
SAML	Secure Assertion Markup Language
SGML	Standard Generalized Markup Language
SOAP	Simple Object Access Protocol

SP1	Service Pack 1
SQL	Structured Query Language
SSL	Secure Socket Layer
TCP/IP	Transmission Control Protocol/Internet Protocol
UDDI	Universal Description, Discovery, and Integration
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VB	Visual Basic
W3C	World Wide Web Consortium
WSDL	Web Service Description Language
WS-Security	Web Services Security
XACML	Extensible Access Control Markup Language
XHTML	eXtensible HyperText Markup Language
XKMS	XML Key Management Specification
XML	eXtensible Markup Language

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: NÁZEV PŘÍLOHY

P I CD s veškerými obrázky, kódy a dokumentací v elektronické podobě