Webový správce obrázků

Web-based image explorer

Pavel Košutek

Bakalářská práce
2009

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2008/2009

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Pavel KOŠUTEK**
Studijní program: **B 3902 Inženýrská informatika**
Studijní obor: **Informační technologie**

Téma práce: **Webový správce obrázků**

Zásady pro vypracování:

1. **Úvod do využitých technologií.**
2. **Kritické zhodnocení existujících řešení.**
3. **Zpracování funkčních požadavků na systém.**
4. **Realizace navrženého systému.**

Rozsah práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **KITTEL, Michael A. ASP.NET 2.0 cookbook. 2nd edition. Sebastopol, California : OReilly, 2006. 1014 s. ISBN 0-596-10064-7.**
2. **WENZ, Christian. Programming ASP.NET AJAX. 1st edition. Cambridge : OReilly, 2007. 454 s. ISBN 0-596-51424-7.**
3. **MACDONALD, Matthew, SZPUSZTA, Mario. ASP.NET 2.0 a C :Tvorba dynamických stránek profesionálně. 1. vyd. Brno : Zoner Press, 2006. 1376 s. ISBN 80-86815-38-2.**
4. **NORTHRUP, Tony, WILDERMUTH, Shawn. MCTS self-paced training kit (Exam 70-536) : Microsoft .NET Framework 2.0 - application development foundation. Redmond : Microsoft Press, 2006. 1039 s. ISBN 0-7356-2277-9.**
5. **JOHNSON, Glenn, NORTHRUP, Tony . MCTS Self-Paced Training Kit (Exam 70-528) : Microsoft .NET Framework 2.0 Web-Based Client Development. Redmond : Microsoft Press, 2007. 902 s. ISBN 0-7356-2334-1.**
6. **POWERS, Shelley. Adding Ajax. Sebastopol, California : OReilly & Associates, 2007. 382 s. ISBN 978-0-596-52936-9.**
7. **ZAKAS, Nicholas C., MCPEAK, Jeremy, FAWCETT, Joe. Ajax : Profesionálně. Brno : Zoner Press, 2007. 672 s. ISBN 978-80-86815-77-0.**

Vedoucí bakalářské práce: **Ing. Petr Šilhavý**
Ústav aplikované informatiky

Datum zadání bakalářské práce: **20. února 2009**

Termín odevzdání bakalářské práce: **1. června 2009**

Ve Zlíně dne 13. února 2009

prof. Ing. Vladimír Vašek, CSc.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

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

ABSTRAKT

Náplní práce je navrhnout a popsat webovou aplikaci správce obrázků v ASP.NET. Text je rozčleněn do dvou částí.

První část popisuje základní pojmy týkající se technologie ASP.NET, nejpoužívanější formáty počítačové grafiky a ukazuje algoritmy při zpracovávání obrazu počítačem. V druhé části práce jsou popsány funkce programu včetně nejdůležitějších zdrojových kódů.

Klíčová slova: ASP.NET, C#, počítačová grafika

ABSTRACT

The major scope of my work is to design and describe web application of picture administrator in ASP.NET. The text is divided into two parts.

First part gives the description of basic concepts respective to the ASP.NET technology, the most used formats of computer graphics and shows algorythms of picture processing via computer. The second part describes the program functions as well as the most important source codes.

Keywords:ASP.NET, C#, computer graphics

Rád bych poděkoval svému vedoucímu bakalářské práce Ing. Petru Šilhavému za jeho podněty a pomoc při korekci výsledné formy bakalářské práce.

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.

Ve Zlíně

…….
podpis diplomanta

OBSAH

ÚVOD

Dnešní svět si lze bez Internetu asi jen těžko představit. Jeho rozšíření vyneslo do popředí zájmu užívání webových aplikací. Pro vývoj těchto aplikací lze využít některých ze skriptovacích jazyků jako třeba ASP nebo PHP. Právě z prvně jmenovaného se postupem času vyvinula nová technologie ASP.NET, od základů vybudována jako objektová platforma.

Použití technologie ASP.NET pro vývoj dynamických stránek za poslední dobu velmi vzrostlo. Výhod ASP.NET před původním ASP můžeme nalézt několik. Například aplikace běží díky kompilovanému kódu rychleji, programátoři mají na výběr z více jazyků a bohatý výběr ovládacích prvků a knihoven tříd velmi urychluje vývoj.

Již dávno jsou pryč doby, kdy uživatel pracoval s počítačem pouze v textovém režimu. Používání grafiky umožnilo uživatelsky mnohem příjemnější přístup k této technice, což je jedním z důvodů masivní rozšířenosti. Vždyť již po zavedení téměř každého novějšího operačního systému se na obrazovce objeví jeho graficky zpracované uživatelské prostředí a grafické zpracování stránek ovlivňuje i jejich návštěvnost.

Pokud pohlédneme na počítačovou grafiku z technického hlediska, pak je to obor informatiky využívající počítače pro zobrazovaní a úpravu prostorových informací reálného světa nebo vytváření umělých grafických objektů. Umělecký pohled považuje počítačovou grafiku za samostatnou kategorii grafiky.

Počítačových programů, které pracují s grafikou můžeme nalézt nespočetné množství. Některé z nich umějí upravovat fotografie, v jiných zase můžeme vytvářet trojrozměrné modely reálných těles.

Cílem této práce má být poskytnutí základních grafických transformací, pro které bylo již napsáno nepřeberné množství desktopových aplikací, také uživateli aplikací webových. Například začlenění do některé webové služby poskytující ukládání a prezentování vlastních fotogalerií na Internetu, by mohl usnadnit vkládání a úpravu fotografií.

I. TEORETICKÁ ČÁST

1 ASP.NET

1.1 Seznámení s technologií

ASP.NET je technologie určená pro vývoj webových aplikací na jejímž vzniku se zasloužila firma Microsoft.

Webová aplikace je velmi často používaný pojem pro který existuje více definic. Například internetová encyklopedie Wikipedie podává toto vysvětlení : „Webová aplikace je aplikace, která je zpřístupněná webovému prohlížeči přes síť podobné internetu nebo intranetu“ ([5]). Webová aplikace je postavena na modelu klient/server a ke komunikaci mezi klientem a serverem slouží protokol http.Veškeré akce požadované klientem se tedy vykonávají na straně serveru. Je složená z webových stránek. ASP.NET je součástí .NET Frameworku a proto ho nelze stáhnout samostatně.

1.1.1 Verze ASP.NET

Prvním verzí se stalo ASP.NET 1.0 a v jeho rámci pak další verze 1.1, 2.0, a 3.5.

Z důvodu použití názvu .NET Framework 3.0 neexistuje verze ASP.NET 3.0. Název .NET Framework 3.0 patří totiž několika novým technologiím - WPF (Windows Presentation Foundation), WCF (Windows Communication Foundation) a WF (Windows Workflow Foundation), ale neobsahuje novou verzi CLR ani ASP.NET.

1.1.2 Multijazyčnost

Velkou výhodou ASP.NET je více jazyčnost. Lze totiž použít jakýkoliv programovací jazyk odpovídající určitým zásadám. Můžeme používat jazyk Visual Basic.NET, řízené C++, C# nebo J#.

Je to proto, že jakákoliv aplikace pro .NET běží v řízeném prostředí CLR(*common language runtime*).

1.1.3 Kompilace a spouštění aplikací

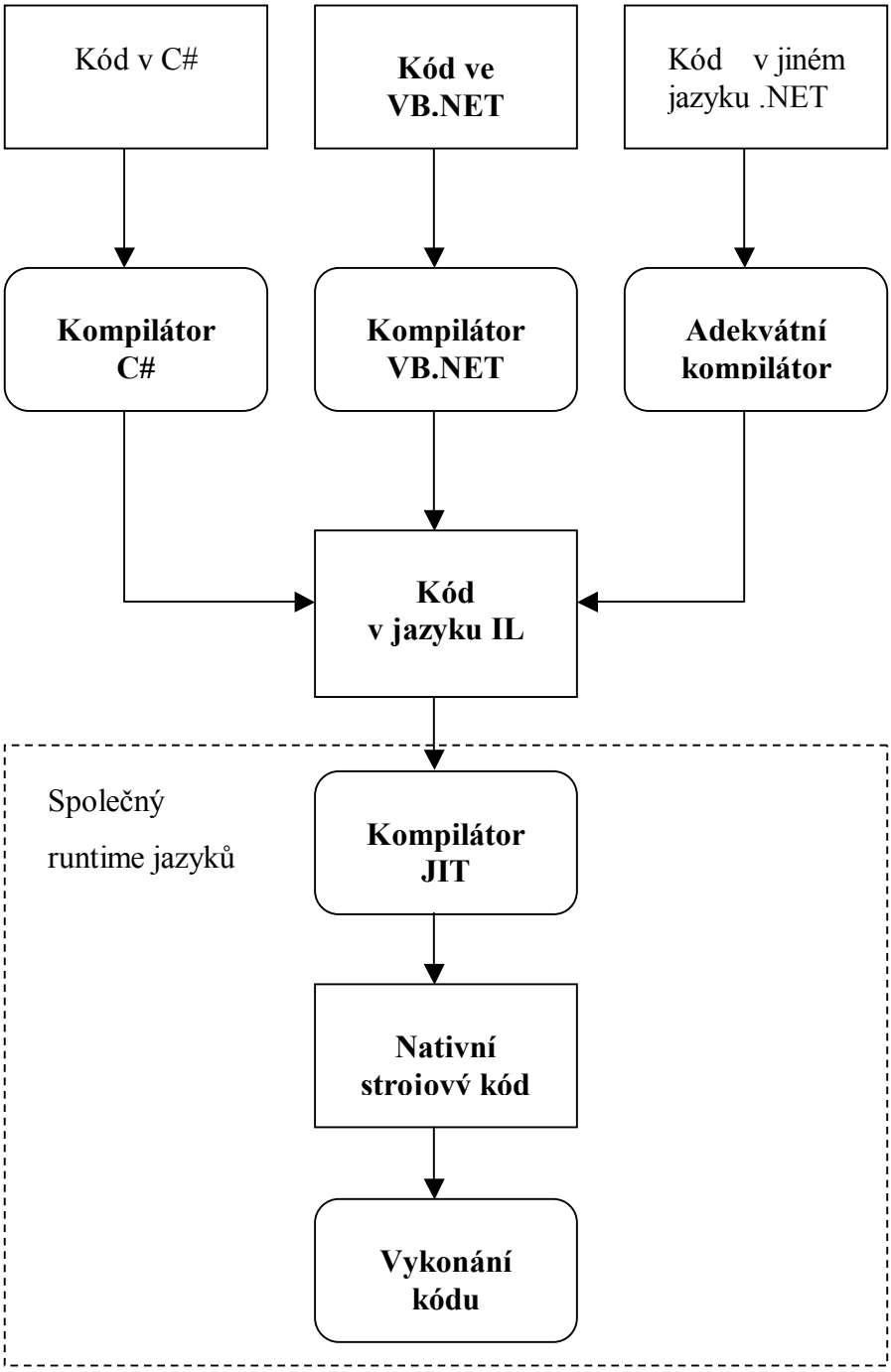
Nevýhodou staršího ASP je používání interpretovaných skriptovacích jazyků v kódu webových stránek což znamená, že při vykonávání aplikace musí server interpretovat kód a přeložit ho do strojového kódu nižší úrovně. U ASP.NET se aplikace neinterpretují, ale

vždy se kompilují. Kompilace má 2 části (obr.1). V té první se námi napsaný kód například v VB nebo C# zkompiluje do Microsoft Intermediate Language (zkráceně IL), což je přechodný jazyk. IL je jediný jazyk kterému rozumí CLR a soubor s tímto kódem se nazývá assembly.

Těsně před vykonáním stránky nastane druhá část kompilace kdy se kód IL zkompiluje do nativního nízkoúrovňového kódu (kompilace just-in-time).

1.1.4 Objektově orientovaný model

Nespornou výhodou ASP.NET je jeho objektová orientovanost a přístup ke všem objektům .NET Frameworku. Programátor může také například vytvářet nové třídy nebo již existující rozšiřovat děděním. Výborným příklad zapouzdření jsou serverové ovládací prvky se kterými může vývojář manipulovat, poskytovat jim data nebo mohou reagovat na události. Kromě toho není třeba psát jejich HTML kód ručně. Značky webových ovládacích prvků začínají vždy prefixem asp:. Například značka prvku Buton neboli tlačítko bude vypadat takto



Obr. 1. Kompilace webové aplikace v ASP.NET

1.2 Programování na straně klienta ASP.NET AJAX

Jak jsem již uvedl v předchozích kapitolách architektura ASP.NET je postavena na modelu odeslání dat z prohlížeče na server, kde se zpracuje a výsledek je odeslán zase zpátky prohlížeči, který aktualizuje stránku. Veškerá práce je tedy vykonávána na serveru. Tato architektura má své výhody. Například zaručuje kompatibilitu se všemi prohlížeči. Problém ovšem nastane pokud má nastat nějaká událost jako reakce na pohyb myši nebo na stisk klávesy. Je zřejmé, že v tomto případě není tento model příliš efektivní.

Jedním z řešení je použití JavaScriptu, který používají některé ovládací prvky. Pokud máme například tlačítko v menu u kterého se při najetí myší změní barva pak tuto akci vykoná prohlížeč. Data se na server odešlou až uživatel na tlačítko klikne.

Řešení, které umožňuje aby stránka zavolala server a aktualizovala se bez toho aby spustila celé odeslání na server je Ajax (Asynchronous JavaScript and XML).

Výhody

- urychluje uživateli práci, šetří datové přenosy

Nevýhody

- nepoužitelnost tlačítka Zpět. Při změnách na stránce pomocí AJAXu se nemění URL v adresním řádku prohlížeče. Proto není možné takto modifikovanou stránku poslat e-mailem nebo uložit do záložek. Tento problém řeší AJAXové aplikace tak, že se za URL dosazují identifikátory začínající na # (odkaz dovnitř stránky). Při opětovném vyvolání takového URL ho JavaScript zjistí a uvede stránku do příslušného stavu (tím se dá vyřešit i problém s tlačítkem Zpět). Problém je, že na cílovém počítači musí být dostupný onen JavaScript.

AJAX by se neměl používat místo klasických odkazů. Totiž například při kliknutí v menu na jednu z položek se nepřejde na novou stránku, ale pouze se nahraje nový obsah do stávající stránky. Toto řešení přináší problémy s přístupností i správou obsahu. Proto v tomto případě je lépe používat klasické odkazy.

Jako jeden z prvních kdo používal AJAX byl Google například pro našeptávač.

1.3 Ovládací prvky

Umožňují vytváření webových aplikací pomocí jednotlivých komponent jako je tomu u desktopových aplikací a skládá se z nich webová aplikace. Mohou být jak vizuální (např. text, obrázky, kalendář) nebo nevizuální (připojení k databázi). Zápis těchto prvků je do XML tagů, které automaticky renderují své HTML obrazy před odesláním klientovi a dokáží vyvolávat události na straně serveru.

- HTML Controls – HTML tagy s atributem runat="server"
- Web Controls – předpřipravené prvky, některé mají svůj obraz v HTML
- User Controls a Cumstom Controls – uživatelsky vytvářené prvky

ASP.NET nám poskytuje velký výběr ovládacích prvků. Od těch jednoduchých jako třeba Button (tlačítko) nebo TextBox (textový vstup) po inteligentní datové prvky mezi které patří GridView nebo ListView.

1.3.1 GridView

Verze ASP.net 1.x poskytovala ovládací prvek DataGrid. Ve verzi 2.0 se objevil nový ovládací prvek GridView, který je zdokonalením a rozšířením původního prvku DataGrid se zachováním zpětné kompatibility.

„GridView je neobyčejně flexibilní ovládací prvek pro zobrazování dat v mřížce skládajících se z řádků a sloupců." ([1]) Má celou řadu zabudovaných schopností jako například výběr, stránkování nebo editování bez nutnosti editovat zdrojový kód. Další výhodou je možnost rozšiřování pomocí šablon.

V aplikaci Webový správce obrázků jsem použil právě tento prvek pro vytvoření správce adresářů a souborů.

1.3.1.1 Sloupce

Tento prvek obsahuje funkci pro automatické generování sloupců. Případě, že použijeme tuto funkci prozkoumá GridView datový objekt a vyhledá všechny sloupce záznamu nebo

vlastnosti vlastního objektu. Tato vlastnost je vhodná například pro rychlé testování stránek ovšem pro praktické použití postrádá určitou flexibilitu.

Při nastavování GridView můžeme využít sedmi typů sloupců

- BoundField – zobrazuje text z pole ve zdroji dat
- ButtonField – pro každý prvek v seznamu zobrazí tlačítko
- CheckBoxField – pro každý prvek v seznamu zobrazí zaškrtávací políčko
- CommandField – zobrazí tlačítka pro výběr nebo editaci
- ImageField – zobrazí obrázek
- TemplářeField – poskytuje největší kontrolu nad zobrazením. Umožňuje specifikovat vlastní ovládací prvky nebo vícenásobná pole

1.4 Operace s adresáři a soubory

Třídy pro získávání informací o systému souborů nalezneme ve jmenném prostoru System.IO. Jsou to dvě sady tříd. Do první patří Directory, File a do druhé zase DriveInfo, DirectoryInfo a FileInfo. Rozdíl mezi sadami spočívá v tom, že abych mohli zavolat nějakou metodu z DirectoryInfo nebo FileInfo musíme nejprve vytvořit jejich objekty, zatímco metody z Directory a File jsou dostupné stále. DirectoryInfo a FileInfo provádějí testy bezpečnosti jenom jednou, při vytvoření instance objektu a programátor se nemusí zabývat uváděním názvu adresáře nebo souboru při volání nějaké metody. Pro jednorázové úlohy jsou ovšem vhodnější Directory a File.

Třída DriveInfo umožňuje získat informace o diskových jednotkách jako například názvy jednotek nebo zjištění kolik na jednotce zbývá volného místa.

V některých případech potřebujeme manipulovat s cestami uloženými jako textové řetězce. Pro tyto chvíle je určena třída Path ze jmenného prostoru System.IO. Pomocí této třídy můžeme předejít nejrůznějším chybám. Ulehčí nám práci pokud například potřebujeme získat z cesty název souboru nebo informace o názvu adresáře.

1.5 Správa stavu

Aby aplikace ASP.NET byly schopny obsluhovat velké množství požadavků, musí se klient po každém požadavku od serveru odpojit. Zároveň jsou odstraněny všechny objekty pro stránky vytvořené. Nevýhodou je nutnost používat techniky pro uložení dat, které musí přetrvat mezi jednotlivými požadavky.

1.5.1 Stav zobrazení

Slouží pro ukládání dat v rámci jedné stránky a jejich udržení mezi odesíláním požadavků na server. Může ukládat jednoduché datové typy, ale i objekty.

```
1 ViewState[“text”] = “text”;
2 string text = (string)ViewState[“text”] ;
```

Obr. 2. Uložení a následné načtení textového řetězce

1.6 Vývojová prostředí

Pro vývoj webových aplikací máme možnost sáhnout po někdy ne zrovna levných vývojových prostředích . Naštěstí ale existují i prostředí, která jsou k dispozici zdarma.

Jedním z těch placených je například Visual Studio od společnosti Microsoft, u kterého je také k dispozici verze Express, která je zdarma.

Další placenou možností může být Macromedia Dreamweaver, který ASP.NET podporuje od verze MX. Z těch volně dostupných bych zmínil Web Matrix, který je rovněž od společnosti Microsoft.

1.6.1 Visual Studio

Visual studio je profesionální vývojářský nástroj podporující sadu soustavy různých nástrojů. Ve verzi 2005 došlo ke změně možnosti vyvíjet aplikace bez projektu. Verze 2008 už poskytuje možnost vytváření aplikací bez projektu i s ním. Pokud sáhneme po vývoji založeném na projektu VS vytvoří soubor projektu do kterého uloží informace o souborech projektu a nastavení. Veškerý kód se poté po spuštění projektu zkompiluje do assembly

ještě před zobrazením ve webovém prohlížeči. Při druhé možnosti, vývoj bez projektu, jsou všechny soubory v adresáři webu je součástí aplikace. Kód se kompiluje až při požádání o zobrazení stránky.

1.6.1.1 Vývojové rozhraní

- Integrovaný webový server. Pro testování aplikace ASP.NET je potřeba webový server. Jedna z možností je její testování přímo na serveru některého webhostingu. Tato možnost je ale velmi nepraktická. Další možnost je připravit si vlastní webový server. Visual Studio vývojářům usnadňuje práci spouštěním vlastního serveru přímo z jeho prostředí.
- IntelliSence. Tato funkce označuje chyby a doporučuje opravy.
- Vícejazyčný vývoj. Visual Studio umožňuje napsat aplikaci i ve více jazycích.
- Ladící nástroje. Kód můžeme krokovat nebo si prohlížet informace z paměti počítače.

2 POČÍTAČOVÁ GRAFIKA

2.1 Grafické formáty

Stanovují pravidla pro uložení obrázků do souboru. Z hlediska ukládání obrazových dat můžeme grafické formáty rozdělit na rastrové ukládající data jako soustavu pixelů a vektorové, které ukládají soustavu matematických obrazců. Vektorová grafika je tedy tvořená matematickými vzorci opisujícími vzhled obrazce. Výhodou tohoto formátu je možnost modifikace obrazce bez toho aby došlo ke zhoršení kvality zobrazení. Vektorová grafika se ale nehodí na zobrazování fotorealistických obrazů. V souboru s vektorovou grafikou jsou uloženy posloupnosti příkazů podle kterých se vykresluje objekt kdežto rastrové soubory ukládají informace o obrazových bodech tvořících obrázek.

Další důležité rozdělení je podle metody komprese. Při kompresi totiž máme dvě možnosti, ztrátovou a bezztrátovou. Při první se část obrazové informace vynechá a u druhé odpovídá komprimovaný obraz původnímu.

2.1.1 JPEG

Je velmi často využívaný ztrátový grafický formát vhodný například pro ukládání grafiky s ostrým přechodem mezi barvami a poskytuje velmi dobrou kvalitu obrazu. Své velké rozšířenosti vděčí také z velké míry digitálním fotoaparátům. Podporuje grafickou hloubku 24 bitů.

2.1.2 PNG

PNG je formát vyvinutým jako náhrada za placený GIF. Grafickou informaci komprimuje bezztrátově. Podporovaná grafická hloubka je 32 bitů. K 24 bitovému obrázku přidává 8 bitový alfa kanál ve kterém jsou zaznamenány grafické efekty jako například stínování.

2.1.3 BMP

Bitmapové ukládání obrázků je jedním z nejjednodušších způsobů zaznamenání grafické informace. V tomto formátu jsou jednotlivé pixely zaznamenány tak, jak jdou za sebou a pro každý je zaznamenána barevná hloubka informace. Barevná hloubka u BMP je 24 a podporuje i alfa kanál. Výhodou bitmapového bezztrátového formátu je uložení grafické informace v maximální kvalitě, identické s předlohou. Výsledná velikost souboru BMP

záleží pouze na počtu pixelů. Pro každý pixel jsou vyhrazeny ve 24 bitové barevné hloubce 3 byty.

2.1.4 GIF

Tento původně patentovaný a placený formát určený pro bezztrátové ukládání obrázků s barevnou hloubkou 8 bitů je vhodný pro ukládání vektorové grafiky v rastrové podobě. Pro ukládání fotografií však nepostačuje jeho barevná hloubka. Rozšířený formát pod označením GIF89a podporuje ukládání více obrázků v jednom souboru pro vytvoření jednoduchých animací.

2.2 Barevné modely

Barevné modely jsou jedním ze způsobů organizace barev. Při rozkladu bílého světla na jeho barevné složky rozlišujeme 6 základních barev

- fialová
- modrá
- modrozelená
- zelená
- žlutá
- červená

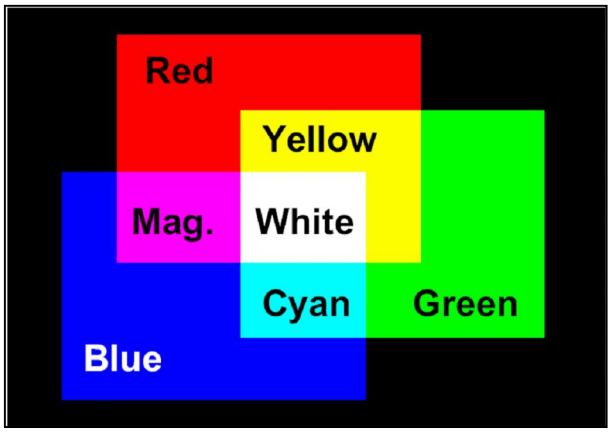
Barvy jsou potom v jednotlivých barevných modelech definovány poměry těchto složek.

2.2.1 RGB

Je založený na třech barvách – červená, zelená a modrá. Lidské oko má nejlepší citlivost právě pro jejich vlnové délky (630 nm, 530 nm a 450 nm).

Model RGB definuje tuto trojici barev každého pixelu, které tvoří jeho barevný vektor. Každá složka barevného vektoru v RGB je reprezentována 8 bitmi, tedy jedním bytem. Protože máme 3 složky dostaneme 24 bitů, tedy 3 byty.

U tohoto barevného modelu jde o aditivní způsob míchání barev, kdy se jednotlivé složky barev sčítají a vytvářejí tak světlo větší intenzity.

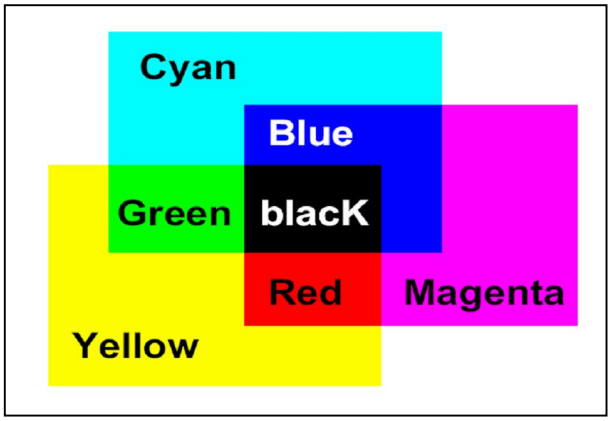


Obr. 3. Skládání barev modelu RGB

Pokud rozebereme jednu složku hlouběji znamená jejich osm bitů číselnou informaci vyjadřující jednu z 256 možných hodnot v intervale <0, 255>. Čím vyšší je celočíselná hodnota tím světlejší je výsledná barva pixelu. Například černou barvu můžeme vyjádřit jako [0, 0, 0] a bílou [255, 255, 255].

2.2.2 CMYK

Tento model obsahuje čtyři základní barvy – azurovou, purpurovou, žlutou a černou. Je založený na subtraktivním míchání barev, kdy se s každou další přidanou barvou ubírá část původního světla.



Obr. 4. Skládání barev modelu CMYK

2.3 Grafické efekty

2.3.1 Inverze barev

Výsledkem tohoto efektu je vytvoření opačné barvy k barvě původní.

Jednomu pixelu přiřadíme označení P1 a jeho barevný vektor bude mít podobu [R1, G1, B1]. Pokud provedeme inverzi toho bodu získáme nový bod P2 s barevným vektorem [R2, G2, B2].

V případě že budeme mít černou barvu pak výsledkem inverze bude barva bílá.

P2 [255 – 0, 255 – 0, 255 -0] = P2 [255, 255, 255]. Tento způsob lze aplikovat na všechny barvy. Pro převedení celého obrázku na inverzní proto musíme projít všechny obrazové body jeden po druhém a tuto operaci provést.

2.3.2 Převedení barev obrázku do odstínů šedé

V tomto případě musíme převést 24-bitovou informaci do 256 odstínů šedé barvy. Účelem algoritmu tohoto efektu je určit změnu hodnoty jasu, kterou získá násobením každé složky konstantou tak jak je tomu v následující rovnici.

$$I = 0{,}299 \times R + 0{,}587 \times G + 0{,}144 \times R \qquad (1)$$

2.4 Práce s grafikou v .NETu

Pro načtení a práce s rastrovou grafikou v .NET je určená třída Image umístěná ve jmenném prostoru System.Drawing. Od této třídy jsou potom odvozeny další dvě, Bitmap pro načítání bitových map a Metafile pro vektorovou grafiku.

2.4.1 Optimalizace grafických transformací

2.4.1.1 Transformační matice ColorMatrix

Ve jmenném prostoru System.Drawing.Imaging se nachází třída ColorMatrix. Tato třída definuje matici typu 5x5 uskutečňující transformaci barev v modelu RGBA. V něm je každá

barva stejně jako v RGB jednoznačně určena barevným vektorem tvořený třemi složkami (červená, zelená, modrá) a navíc je ještě přidána složka čtvrtá tzv. alfa kanál. Při 8-bitové kapacitě je výsledná barva zakódovaná pomocí 32 bitů. Část pro alfa kanál určuje jestli průhlednost pixelu. Pokud chceme aby byl obrázek průhledný bude hodnota alfa kanálu 0 a naopak při požadavku na sytou barvu 255.

Transformační matice je tabulka reálných čísel uspořádaných do pěti řádků a pěti sloupců. Základní tvar je jednotková matice. Matice obsahuje 25 prvků s indexací od nuly. Pokud se budeme zabývat prvky na hlavní diagonále budou prvky [0, 0], [1, 1] a [2, 2] reprezentovat složku červené, zelené a modré. Alfa kanál nejdeme na pozici [3, 3] a poslední prvek musí vždy rovný 1.

Při transformaci barevného vektoru potom vektor násobíme transformační maticí. Při násobení zmíněnou jednotkovou maticí zůstane vektor stejný. Pokud ale chceme získat nový barevný vektor musíme hodnoty na pozicích [0, 0], [1, 1] a [2, 2] změnit.

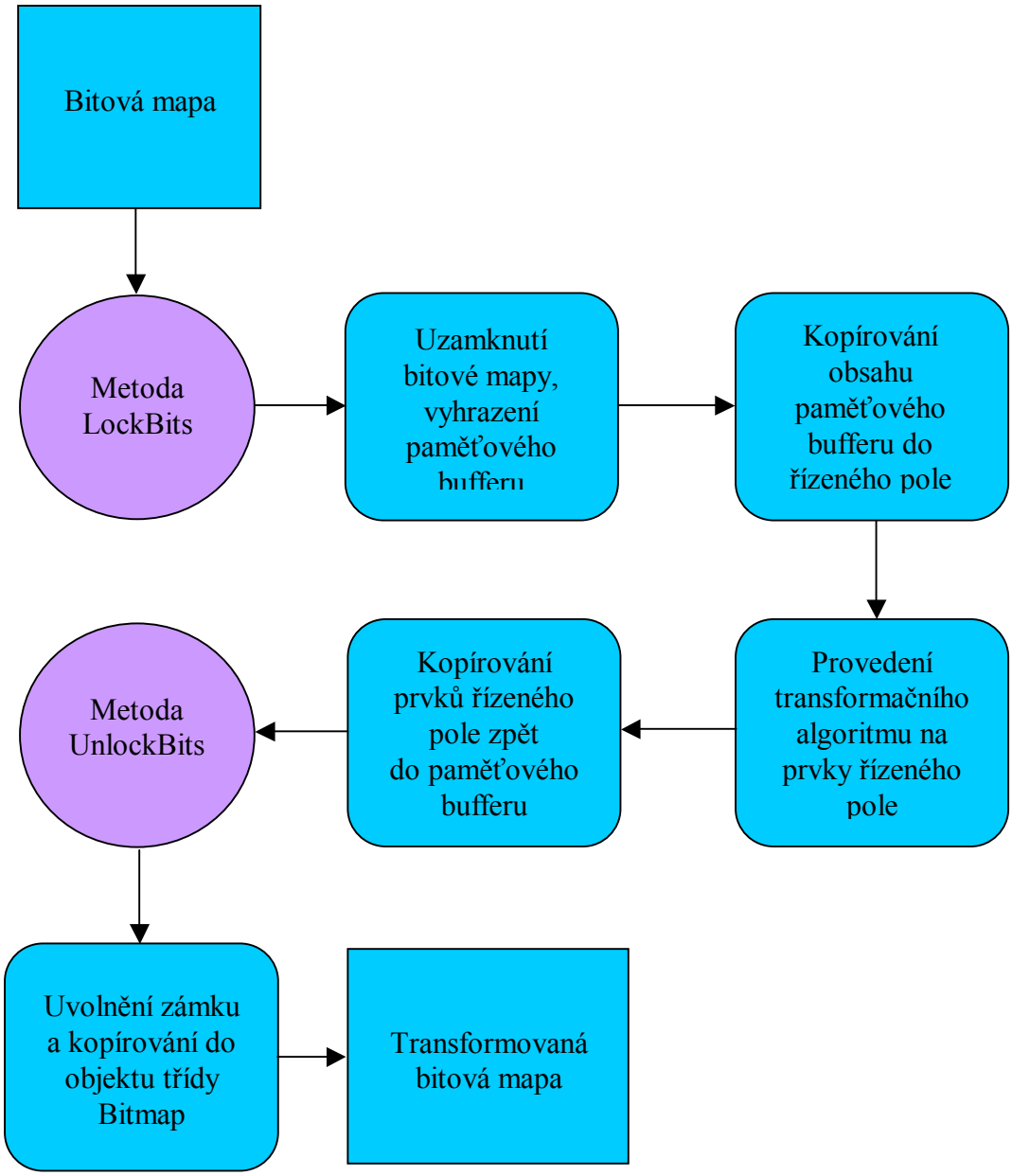
Instance třídy ColorMatrix tedy transformační maticí. Můžeme ji použít pro inverzi barev nebo převedení do odstínů šedé.

2.4.1.2 Technika uzamknutí bitové mapy v operační paměti

Pro tento způsob můžeme použít metodu LockBits() z třídy BitMap. Tato metoda poskytne dočasný paměťový buffer použitelný pro čtení a zápis rastrových údajů. Začátek tohoto bufferu je identifikovatelný pomocí paměťové adresy neboli ukazatele.

Přístup k rastrovým údajům přes ukazatele je velmi efektivní i když se ukazatele v jazyce C# téměř vůbec nepoužívají z důvodu nekorektního použití paměťových adres. Pokud chceme použít ukazatele v jazyce C# musíme je ohraničit do bloků nezabezpečeného kódu označených klíčovým slovem unsafe.

Metoda LockBits() nám vrátí odkaz na objekt třídy BitmapData, díky němuž zjistíme například informace o rozměrech bitové mapy nebo rastrovém formátu, které se uzamknou v paměti. Obsah paměti se poté nakopíruje do řízeného pole a na jeho prvky se provede transformace. Poté dojde ke zkopírování již upraveného řízeného pole zpět do paměťového bufferu, použitím metody UnlockBits() uvolníme zámek a nakopírujeme do objektu třídy BitMap již transformovanou bitovou mapu.



Obr. 5. Postup mechanizmu využívajícího uzamknutí bitové mapy

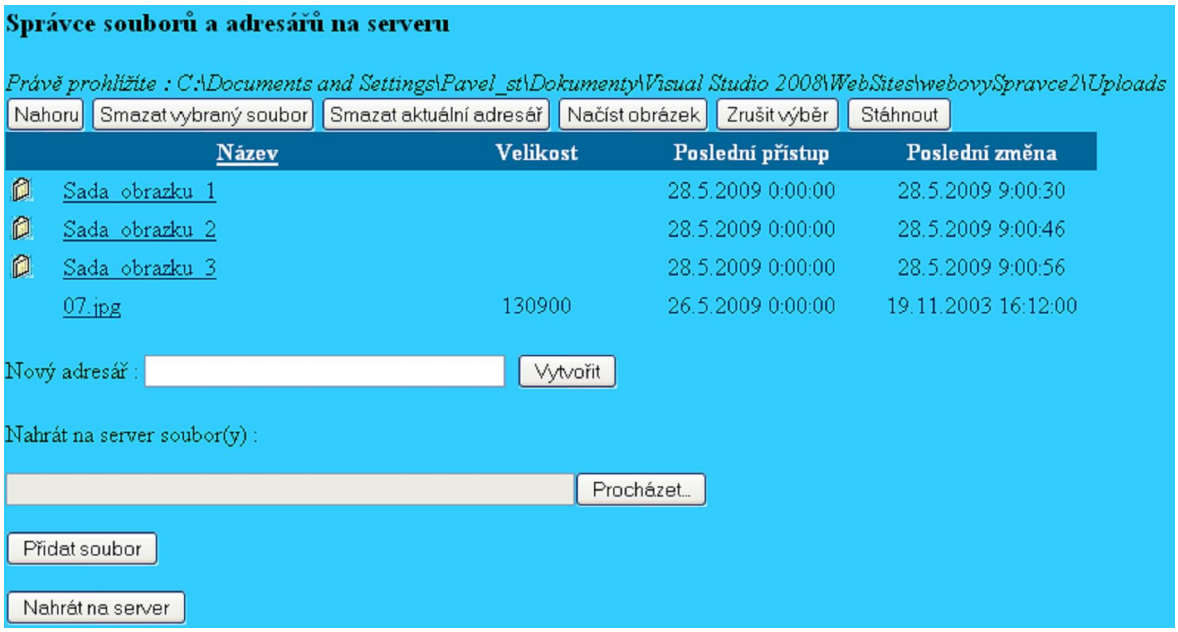
2.4.2 Provnání s grafickým programem Adobe Photoshop

Pokud porovnáme dobu vykonávání například převodu do odstínů šedé nebo inverzi barev s grafickým programem Adobe Photoshop zjistíme, že je tento program vykoná transformaci mnohem rychleji. Je to právě tím, že Photoshop nevykonává algoritmy v řízeném prostředí, ale využívá nízkoúrovňové funkce aplikačního rozhraní Win32 API.

II. PRAKTICKÁ ČÁST

3 SPRÁVCE SOUBORŮ A ADRESÁŘŮ

Umožňuje uživateli procházení adresářů a vytváření nových, nahrávat a mazat soubory, extrahovat zkomprimované soubory a vybírat obrázky pro úpravu. K tomu je určen adresář Uploads. Obsah tohoto adresáře se pomocí datového ovládacího prvku GridView zobrazí automaticky při načtení stránky .



Obr. 6. Správce souborů a adresářů

3.1 Zobrazení seznamu adresářů a souborů

K zobrazení jsou použité dva datové ovládací prvky GridView, jeden pro adresáře a druhý pro soubory.

Načtení souborů a adresářů jsem vyřešil funkcemi GetFiles() pro soubory a GetDirectories() pro adresáře z třídy DirectoryInfo. Pro naplnění prvku GridView informacemi, které vrátí tyto funkce musí být jeho vlastnost DataSource nastavená na objekt z této třídy a po zavolání DataBind() se prvek daty z objektu naplní. Bez zavolání této funkce se vůbec na stránce nezobrazí.

3.2 Procházení hierarchií adresářů

3.2.1 Posun o adresář výš

Pro tento krok musíme aktuální cestu zkombinovat s řetězcem ".." k čemuž můžeme využít funkce Combine() umístěnou v třídě Path.

Protože uživatel má možnost mazání souborů a adresářů je z důvodu správného chodu aplikace potřeba zajistit, aby se ve správci souborů a adresářů nedostal nad adresář jemu určený, v tomto případě Uploads.Tento problém je vyřešen podmínkou. Pokud cesta bude končit tímto adresářem, pak se funkce vykonávající načtení adresářů a souborů nevykoná.

```
1 if (!cesta.EndsWith("Uploads"))
2 {
3     cesta = Path.Combine(cesta, "..");
4     cesta = Path.GetFullPath(cesta);
5     cestaNaServer = cesta;
6     nacist(cesta, adresare, soubory, zobrazeniCesty);
7 }
```

Obr. 7. Posun o adresář výš

3.2.2 Načtení obsahu adresáře

Tím, že uživatel klikne na některý odkaz adresáře vyvolá událost SelectIndexChanged a její obsluha zobrazí jeho podadresář.

3.3 Operace s adresáři

3.3.1 Vytvoření nového adresáře

Vytvoření nového adresáře probíhá použitím funkce CreateSubdirectory() z třídy DirectoryInfo. Do parametru této funkce je vložená cesta k adresáři, ve kterém má být nový adresář vytvořen. Pro mazání adresářů je pak použita funkce Delete() ze stejné třídy.

3.3.2 Smazání adresáře

Uživateli je také poskytnuta možnost mazat adresář ve kterém se právě nachází v případě, že neobsahuje žádné soubory. Po kliknutí na tlačítko Smazat vybraný soubor je načtena aktuální cesta a metodou Delete třídy DirectoryInfo je adresář vymazán. Po smazání se do seznamu načte obsah adresáře, ve kterém se smazaný adresář nacházel.

3.4 Operace se soubory

3.4.1 Nahrání souboru na server

Správce samozřejmě musí uživateli dávat také možnost uložit na server svůj obrázek, který chce upravovat. Pomocí tlačítka může nahrát na server i více souborů najdou. Pokud totiž klikne na toto tlačítko objeví se na stránce pod prvkem Input (File) další tento prvek do kterého může vybrat další soubor. Obsluhu tlačítka pro přidávání souborů vykonává funkce JavaScriptu. Odeslání potom spustí tlačítkem Nahrát na server. Obsluha tlačítka pro odeslání pomocí cyklu for, který se vykoná tolikrát kolik je na stránce přidaných Input prvků.

3.4.2 Stahování obrázků ze serveru

Stahování souborů ze serveru uživatel provede výběrem daného souboru ve správci a kliknutím na tlačítko Stáhnout. Aby se nenačetla místo správce obrázků prázdná stránka, otevře se nová stránka download.aspx, která při svém načítání vykoná operaci pro stáhnutí.

Aby nová stránka vůbec věděla ze které stránky byla spuštěna musel jsem do ní vložit novou řídící direktivu `<%@ PreviousPageType VirtualPath="~/Default.aspx" %>`. Veřejná proměnná cestaKsouboru uloží cestu dvybraného souboru. Po kliknutí na tlačítko Stáhnout se vykoná metoda Server.Transfer(), která otevře stránku download.aspx. Díky vložené řídící direktivě může v této chvíli nová stránka zjistit cestu k souboru. Pomocí objektu Response, který je instancí třídy System.Web.HttpResponse může stránka uložit požadovaný soubor.

```
1 Response.Clear();
2 Response.AddHeader("Content-Disposition",
3                    "attachment; filename=" + soubor.Name);
4 Response.ContentType = "application/octet-stream";
5 Response.WriteFile(soubor.FullName);
6 Response.End();
```

Obr. 8. Postup při stáhnutí souboru ze serveru

3.4.3 Mazání souborů

Stejně jako v případě adresářů má i u souborů uživatel možnost smazat vybraný soubor. V případě, že uživatel chce ze serveru odstranit některý soubor musí tento soubor označit ve správci adresářů a souborů a kliknout na tlačítko Smazat vybraný soubor. Pro tuto akci je z třídy File použita metoda Delete().

3.4.4 Extrahování komprimovaných souborů

Platforma .Net podporuje komprimaci dat. Problém je ale v tom, že přes pravděpodobně stejný komprimační algoritmus není formát souboru stejný jako výstupní formát komprimačních nástrojů jiných výrobců. Proto jsem k extrahování použil externí knihovnu ICSharpCode napsanou pro platformu .Net, kde jednotlivé třídy jsou uložené ve jmenném prostoru SharpZipLib.

Ze jmenného prostoru Zip jsem pro dekomprimaci použil třídu FastZip a její funkci ExtractZip(). Jako parametry funkce se vkládá řetězec názvu souboru, který chceme dekomprimovat, řetězec cílového adresáře a řetězec filtru souborů.

```
1 FastZip z = new FastZip();
2 z.ExtractZip(soubor, cesta, "");
```

Obr. 9. Extrahování souboru.zip

V této třídě můžeme také extrahovat soubory komprimované s heslem, samozřejmě pokud heslo známe.

```
1 FastZip z = new FastZip();
2 z.Password = "nejaky text hesla";
```

Obr. 10. Získání hesla

Pokud uživatel klikne na soubor s příponou zip zobrazí se nad správcem souborů žluté tlačítko Extrahovat, které je při práci s nekomprimovanými soubory skryto.

3.5 Zobrazení náhledu obrázku

Pro zobrazení náhledu jsem použil prvek Image. Tento prvek vyžaduje určitý tvar cesty místa, kde je umístěný obrázek. Požadovaný tvar cesty musí být ve tvaru s normálními lomítky a cesta začíná až adresářem na serveru, kde jsou umístěné stránky. Funkce MapPath() vrátí celou cestu umístění daného obrázku na serveru. Proto nelze použít přímo cestu nastavenou seznamem souborů. Cestu je nutné zobrazit až od adresáře Uploads. Proto pomocí funkce IndexOf() zjistíme index na kterém se v řetězci cesty nalézá název adresáře a pomocí funkce Remove() odstraníme z původního řetězce část od počátku do toho indexu. K prohození zpětných lomítek lze využít funkce Replace(). Do parametrů níže uvedené funkce vkládáme dva řetězce. Prvním je celá cesta vrácení MapPath() a druhý je řetězec, který hledáme. Funkce vymaže z řetězce celé cesty část od začátku až po zadaný řetězec.

```
 1 public string vraceniCastiCesty(string celaCesta,
 2                                  string retezec)
 3 {
 4     int index;
 5     string castCesty;
 6     index = celaCesta.IndexOf(retezec);
 7     castCesty = celaCesta.Remove(0, index);
 8     castCesty = castCesty.Replace('\\','/');
 9     return castCesty;
10 }
```

Obr. 11. Funkce pro vymazání části cesty

4 SPRÁVCE OBRÁZKŮ

Poskytuje možnost provádí základních operací s obrázky jako například měnit jejich velikost, přidávat vodoznak nebo převést obrázek do odstínů šedé.

4.1 Úprava velikosti obrázku

Uživatel zadá do připravených TextBoxů novou výšku a šířku obrázku. Po kliknutí na tlačítko změnit se nejprve převádí text z Textboxů na hodnotu integer. Z obrázku původní velikosti se vytvoří objekt třídy Bitmap. Poté proběhne vložení tohoto objektu a nových rozměrů obrázku do nového objektu stejné třídy. Potom je ještě nutné obraz upravit použitím antialiasingu a nový obrázek uložit.

```
 1 string cesta = (string)ViewState["cesta"];
 2 int novaVyska = int.Parse(txtVyskaObrazu.Text);
 3 int novaSirka = int.Parse(txtSirkaObrazu.Text);
 4 lblAktualniCesta.Text = cesta;
 5
 6 Bitmap staryObraz = (Bitmap)ViewState["obraz"];
 7 Bitmap novyObraz = new Bitmap(staryObraz,
 8                               novaSirka,
 9                               novaVyska);
10
11 Graphics oGraphics = Graphics.FromImage(novyObraz);
12 oGraphics.SmoothingMode = SmoothingMode.AntiAlias;
13 oGraphics.InterpolationMode =
14           InterpolationMode.HighQualityBicubic;
15
16 oGraphics.DrawImage(staryObraz, 0, 0, novaSirka, novaVyska);
17 novyObraz.Save(cesta);
```

Obr. 12. Postup při změně velikosti obrázku

4.2 Rotace obrázků

Pro otáčení a zrcadlové převracení obrázků je připravena funkce RotateFlip(). Do jejího parametru pak můžeme volit otočení podle úhlů a převracení podle os. Například parametr `RotateFlipType.Rotate90FlipNone` otočí obrázek o 90 stupňů do prava nebo `RotateFlipType.RotateNoneFlipX` ho převrátí podle osy x. Po provedení změny se musí opět obrázek uložit.

4.3 Fotografické filtry

4.3.1 Převedení barev na odstíny šedé a invertování barev

Ve fotografických filtrech nesmí pochopitelně chybět funkce, která převede obrázek do odstínů šedé, tedy vytvoří z barevného obrázku černobílý ani ta která metodou inverze barev vytvoří z obrázku jeho negativ. Oba principy byly popsány již dříve v teoretické časti práce a proto se již v této kapitole s nimi nebudu zabývat.

4.3.2 Sépiový efekt

Tato jednoduchá funkce umožní uživateli vložit do obrázků historický nádech pomocí sépiového efektu.

Po uzamčení bitové se mapy převádí RGB složky jednotlivých pixelů. Funkce má jediný parametr, bitmapu upravovaného obrázku.

```
1 public unsafe Bitmap sepiovyEfekt(Bitmap b)
2 {
3       BitmapData bd = b.LockBits(new Rectangle(0, 0,
4                               b.Width,b.Height),
5                               ImageLockMode.ReadWrite,
6                               PixelFormat.Format32bppArgb);
```

```
 7      for (int y = 0; y < b.Height; y++)
 8      {
 9          int* ukzt = (int*)((int)bd.Scan0 + y * bd.Stride);
10          for (int x = 0; x < b.Width; x++)
11          {
12            Color c = Color.FromArgb(*ukzt);
13            int sepia = (int)(0.299 * c.R + 0.587 * c.G
14                                + 0.114 * c.B);
15            int r = (int)((sepia > 206) ? 255 : sepia + 49);
16            int g = (int)((sepia < 14) ? 0 : sepia - 14);
17            int bl = (int)((sepia < 56) ? 0 : sepia - 56);
18            Color sepiaColor = Color.FromArgb(r, g, bl);
19            *ukzt = sepiaColor.ToArgb();
20            ukzt++;
21          }
22      }
23      b.UnlockBits(bd);
24      return b;
25  }
```

Obr. 13. Sépiový efekt

4.3.3 Rozmazání obrázku

Funkce která provádí rozmazání obrázků musí být také označená jako unsafe. Jako její parametry se vkládá proměnná integer určující velikost šumu a bitová mapa obrázku, který bude upravován. Velikost šumu uživatel zadá do textového pole.

Nejprve se data bitové mapy uzamknou v paměti pomocí LockBits. Celé rozmazání pak probíhá pomocí generátoru náhodných čísel tak, že se vezme pixel a vygenerují se náhodné souřadnice x a y. Pomocí ukazatele se potom tento pixel uloží na nové souřadnice a stejným způsobem se pokračuje pro další pixely. Po skončení cyklu vrátí změněnou bitovou mapu.

```
 1 public unsafe Bitmap rozmazaniObrazku(int silaSumu, Bitmap b)
 2 {
 3   BitmapData bd = b.LockBits(
 4                           new Rectangle(0,0,b.Width,b.Height),
 5                           ImageLockMode.ReadWrite,
 6                           PixelFormat.Format32bppArgb);
 7  Random rand = new Random();
 8  for (int y =0; y<b.Height; y++)
 9    {
10   int* ukzt = (int*)((int)bd.Scan0 + y * bd.Stride);
11      for (int x = 0; x< b.Width; x++)
12      {
13       int x2 = Math.Max(Math.Min(b.Width-1,
14                 rand.Next(silaSumu,silaSumu)+x),0);
15       int y2 = Math.Max(Math.Min(b.Height-1,
16                 rand.Next(-silaSumu,silaSumu )+y),0);
17       int* ukzt2 = (int*)((int)bd.Scan0 + y2 *
18                     bd.Stride +x2*4);
19       Color c = Color.FromArgb(*ukzt);
20       *ukzt2 = c.ToArgb();
21       ukzt++;
22       }
23    }
24    b.UnlockBits(bd);
25    return b;
26 }
```

Obr. 14. Funkce pro rozmazání obrázku

4.4 Přidání vodoznaku

Aplikace také umožňuje přidávat do obrázku vodoznak. Uživatel ho může vložit buď jako text nebo si jako vodoznak může vybrat obrázek.

4.4.1 Vodoznak jako text

Ve správci souborů a adresářů si uživatel vybere obrázek do kterého bude chtít přidat vodoznak. Do TextBoxu zadá text a pomocí RadioButtonů vybere polohu textu v obrázku. Na výběr má ze tří poloh, nahoře, dole a uprostřed.

Nejprve se vytvoří objekt Image obrázku do kterého se bude vkládat vodoznak a do proměnných typu integer se uloží výška a šířka. Dalším krokem je vytvoření bitmapy s rozměry původního obrázku, která je vložená do objektu třídy Graphics. Po nastavení kvality renderování funkce DrawImage() do tohoto objektu vykreslí vytvořený objekt Image. Pro určení maximální velkosti písma se testuje více možností, které jsou uloženy v poli. To provádí cyklus for do té doby než je šířka textu menší než šířka obrázku. Následuje určení horizontálního a vertikálního umístění, určení štětce a vykreslení textu do obrázku. Funkce vrátí objekt s obrázkem obsahujícím vodoznak.

```
1  public System.Drawing.Image vodoznakText(
2                                  string pracovniAdresar,
3                                  TextBox txtTextVodoznaku,
4                                  int umisteni)
5  {
6    string textVodoznaku = txtTextVodoznaku.Text;
7    System.Drawing.Image imgPhoto =
8                System.Drawing.Image.FromFile(pracovniAdresar);
9    int phWidth = imgPhoto.Width;
10   int phHeight = imgPhoto.Height;
11   Bitmap bmPhoto = new Bitmap(phWidth, phHeight,
12                                PixelFormat.Format24bppRgb);
13   bmPhoto.SetResolution(imgPhoto.HorizontalResolution,
14                         imgPhoto.VerticalResolution);
15   Graphics grPhoto = Graphics.FromImage(bmPhoto);
16   grPhoto.SmoothingMode = SmoothingMode.AntiAlias;
```

```
17   grPhoto.DrawImage(imgPhoto,
18                     new Rectangle(0, 0, phWidth, phHeight),
19                     0,
20                     0,
21                     phWidth,
22                     phHeight,
23                     GraphicsUnit.Pixel);
24
25   int[] sizes = new int[] {60,56,50,46,40,36,30,28,24
26                            ,20,18,16,14,12,10,8,6,4};
27   Font crFont = null;
28   SizeF crSize = new SizeF();
29
30   for (int i = 0; i < 7; i++)
31   {
32      crFont = new Font("arial", sizes[i], FontStyle.Bold);
33      crSize = grPhoto.MeasureString(textVodoznaku, crFont);
34      if ((ushort)crSize.Width < (ushort)phWidth)
35         break;
36   }
37   int yPixlesFromBottom = (int)(phHeight * .9);
38   if (umisteni == 1)
39   {
40      yPixlesFromBottom = (int)(phHeight * .05);
41   }
42   if (umisteni == 2)
43   {
44      yPixlesFromBottom = (int)(phHeight * .5);
45   }
46   if (umisteni == 3)
47   {
48      yPixlesFromBottom = (int)(phHeight * .9);
49   }
```

```
50   float yPosFromBottom = ((phHeight - yPixlesFromBottom)
51                                   - (crSize.Height / 2));
52
53   float xCenterOfImg = (phWidth / 2);
54
55
56   StringFormat StrFormat = new StringFormat();
57   StrFormat.Alignment = StringAlignment.Center;
58
59   SolidBrush semiTransBrush2 =
60              new SolidBrush(Color.FromArgb(153, 0, 0, 0));
61
62   grPhoto.DrawString(textVodoznaku,
63                      crFont,
64                      semiTransBrush2,
65                      new PointF(xCenterOfImg + 1,
66                                 yPosFromBottom + 1),
67                      StrFormat);
68   SolidBrush semiTransBrush =
69   new SolidBrush(Color.FromArgb(153, 255, 255, 255));
70   grPhoto.DrawString(textVodoznaku,
71                    crFont,
72                    semiTransBrush,
73                    new PointF(xCenterOfImg, yPosFromBottom),
74                    StrFormat);
75   imgPhoto = bmPhoto;
76   grPhoto.Dispose();
77   return imgPhoto;
78 }
```

Obr. 15. Vložení textu do obrázku jako vodoznak

4.4.2 Vodoznak jako obrázek

Uživatel stejně jako v předchozím případě vybere obrázek ve správci souborů a adresářů.

Pomocí prvku FileUpload vybere obrázek, který chce vložit jako vodoznak a kliknutím na tlačítko Načíst nahraje tento soubor na server. Nový soubor se uloží do adresáře Vodoznaky. Pokud tento adresář neexistuje automaticky se vytvoří. RadioButtony opět dávají možnost vybrat umístění vodoznaku. Nadefinováno je pět možností z nichž čtyři jsou v každém rohu a jedna umístí vodoznak na střed. V prvních čtyřech možnostech dojde ke vložení vodoznaku 10 pixelů od okrajů z obou stran. Například pokud bude obrázek umístěný do pravého horního rohu bude souřadnice y rovná 10 a u souřadnice x zjistíme rozdíl šířky celého obrázku a šířky vodoznaku a od něho potom odečteme 10. Kliknutím na tlačítko Vložit dojde k uložení obrázku s vloženým vodoznakem.

```
 1 public System.Drawing.Image vodoznakObrazek(
 2                                     string cestaOriginal,
 3                                     string cestaVodoznak,
 4                                     int polohaVodoznaku)
 5 {
 6         System.Drawing.Image imgPhoto =
 7         System.Drawing.Image.FromFile(cestaOriginal);
 8         int phWidth = imgPhoto.Width;
 9         int phHeight = imgPhoto.Height;
10         Bitmap bmPhoto = new Bitmap(phWidth,
11                                phHeight,
12                                PixelFormat.Format24bppRgb);
13
14         bmPhoto.SetResolution(imgPhoto.HorizontalResolution,
15                                imgPhoto.VerticalResolution);
16         Graphics grPhoto = Graphics.FromImage(bmPhoto);
17         System.Drawing.Image imgWatermark =
18                    new Bitmap(cestaVodoznak);
19         int wmWidth = imgWatermark.Width;
20         int wmHeight = imgWatermark.Height;
```

```
21          grPhoto.SmoothingMode = SmoothingMode.AntiAlias;
22          grPhoto.DrawImage(
23              imgPhoto,
24              new Rectangle(0, 0, phWidth, phHeight),
25              0,
26              0,
27              phWidth,
28              phHeight,
29              GraphicsUnit.Pixel);
30
31          Bitmap bmWatermark = new Bitmap(bmPhoto);
32          bmWatermark.SetResolution(
33                          imgPhoto.HorizontalResolution,
34                          imgPhoto.VerticalResolution);
35          Graphics grWatermark =
36                              Graphics.FromImage(bmWatermark);
37
38          ImageAttributes imageAttributes =
39                                new ImageAttributes();
40
41          ColorMap colorMap = new ColorMap();
42
43          colorMap.OldColor = Color.FromArgb(255, 0, 255, 0);
44          colorMap.NewColor = Color.FromArgb(0, 0, 0, 0);
45
46          ColorMap[] remapTable = { colorMap };
47
48          imageAttributes.SetRemapTable(remapTable,
49                                  ColorAdjustType.Bitmap);
50          float[][] colorMatrixElements = {
51             new float[] {1.0f,  0.0f,  0.0f,  0.0f, 0.0f},
52             new float[] {0.0f,  1.0f,  0.0f,  0.0f, 0.0f},
53             new float[] {0.0f,  0.0f,  1.0f,  0.0f, 0.0f},
54             new float[] {0.0f,  0.0f,  0.0f,  0.3f, 0.0f},
55             new float[] {0.0f,  0.0f,  0.0f,  0.0f, 1.0f}};
```

```
56          ColorMatrix wmColorMatrix =
57                      new ColorMatrix(colorMatrixElements);
58
59          imageAttributes.SetColorMatrix(wmColorMatrix,
60                                  ColorMatrixFlag.Default,
61                                  ColorAdjustType.Bitmap);
62          int xPosOfWm = 0;
63          int yPosOfWm = 0;
64
65          if (polohaVodoznaku == 1)
66          {
67              xPosOfWm = 10;
68              yPosOfWm = 10;
69          }
70          if (polohaVodoznaku == 2)
71          {
72              xPosOfWm = ((phWidth - wmWidth) - 10);
73              yPosOfWm = 10;
74          }
75
76          if (polohaVodoznaku == 3)
77          {
78              xPosOfWm = phWidth / 2 - wmWidth / 2;
79              yPosOfWm = phHeight / 2 - wmHeight / 2;
80          }
81
82          if (polohaVodoznaku == 4)
83          {
84              xPosOfWm = 10;
85              yPosOfWm = (phHeight - wmHeight) - 10;
86          }
87          if (polohaVodoznaku == 5)
88          {
89              xPosOfWm = ((phWidth - wmWidth) - 10);
90              yPosOfWm = (phHeight - wmHeight) - 10;
91          }
```

```
 92          grWatermark.DrawImage(imgWatermark,
 93              new Rectangle(xPosOfWm,
 94                            yPosOfWm,
 95                            wmWidth,
 96                            wmHeight),
 97              0,
 98              0,
 99              wmWidth,
100              wmHeight,
101              GraphicsUnit.Pixel,
102              imageAttributes);
103
104          imgPhoto = bmWatermark;
105          grPhoto.Dispose();
106          grWatermark.Dispose();
107          return imgPhoto;
108  }
```

Obr. 16. Vložení obrázku jako vodoznak

5 UMÍSTĚNÍ APLIKACE NA INTERNET

Pro umístění aplikace na internet je důležité si vybrat správný webhosting. V dnešní době můžeme najít na internetu velký počet českých i zahraničních hostingů podporující technologii .NET mezi nimiž nalezneme i služby poskytované zdarma.

V hostingu je také potřeba nastavit přístupová práva na uživatelský adresář Uploads. Jeho další adresáře už práva potom dědí.

V případě aplikace Webový správce obrázků musí hosting pracovat s bloky unsafe použitými pro některé obrazové transformace.

ZÁVĚR

Cílem této práce bylo vytvořit webovou aplikaci správce obrázků v technologii ASP.NET a popsat její funkce. Teoretická část práce se snaží čtenáře seznámit se základy ASP.NET. Poté jsou popsány nejpoužívanější grafické formáty, barevné modely a základní algoritmy grafické transformace digitálního obrazu. V praktické části vysvětluji jak bylo postupováno při tvorbě jednotlivých funkcí aplikace a text je pro lepší představu doplněn ukázkami jejich zdrojových kódů.

ZÁVĚR V ANGLIČTINĚ

The main objective of this work was to create a web picture administrator application in the ASP.NET and to describe its functions. The theoretical part of the work tries to familiarize the readers with the very basics of ASP.NET. Subsequently the description of the most used graphic formats as well as the basic algorythms of graphic transformation of digital image follow. In the practical part I give an explanation of process of how each and individual functions of the application have been created and furthermore the text is supported with examples of source codes of the functions for better understanding.

SEZNAM POUŽITÉ LITERATURY

Monografie:

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

[2] ZAKAS, Nicholas C., MCPEAK, Jeremy, FAWCETT, Joe. *Ajax : Profesionálně*. Brno : Zoner Press, 2007. 672 s. ISBN 978-80-86815-77-0.

[3] NORTHRUP, Tony, WILDERMUTH, Shaw. MCTS self-paced training kit (Exam 70-536) : Microsoft .NET Framework 2.0 – application develoment foundation. Redmond : Microsoft Press, 2006. 1039 s. ISBN 0-7356-2277-9

Internetové zdroje:

[3] *Microsoft asp.net* [online]. [cit. 2009-04-28]. Dostupný z WWW : <http://www.asp.net>.

[4] *MSDN : Microsoft Developer Network* [online]. [cit. 2009-04-15]. Dostupný z WWW: <http://msdn.microsoft.com>.

[5] *WIKIPEDIE : Otevřená encyklopedie* [online]. [cit. 2009-04-06]. Dostupný z WWW : <http://cs.wikipedia.org/>.

SEZNAM OBRÁZKŮ