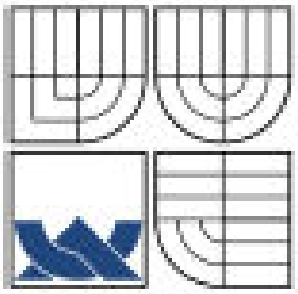
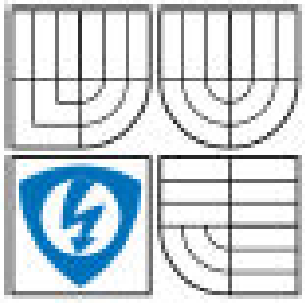


VYSOKÉ UCENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

ROZDÍLY VE VÝVOJI DYNAMICKÉHO OBSAHU PŘI POUŽITÍ TECHNOLOGIÍ PHP A ASP.NET

THE DIFFERENCES BETWEEN PHP AND ASP.NET IN DYNAMIC WEB DEVELOPMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE RADAN JÁNOŠ
AUTHOR

VEDOUCÍ PRÁCE Ing. PETR KOVÁŘ
SUPERVISOR

BRNO 2008

Anotace (česky)

Bakalářská práce analyzuje možnosti vývoje dynamického webového obsahu pomocí technologií PHP a ASP.NET, pokouší se o jejich srovnání a stanovuje základní rozdíly. Celá práce je rozdělena do čtyř hlavních kapitol. První, úvodní část, pojednává o vývoji dynamického webového obsahu, používaných technologiích a zasvěcuje čtenáře do dané problematiky. Druhá kapitola se věnuje již analýze a srovnání výše zmíněných technologií. Tato část začíná základním představením a historií technologií, následuje praktická část „instalace, nastavení, konfigurace“, kde je čtenář seznámen s potřebnými prostředky pro rozběhnutí každé z technologií. Dále pojednává o způsobu zápisu kódu včetně syntaxe jazyků, ve kterých je PHP srovnáváno s jazykem C# a popisuje objektovou orientaci jazyků. Kapitola obsahuje příklady přístupu obou technologií k datovým zdrojům, zda využívá funkcí či objektů, speciálních knihoven apod. Shrnuje podporu platforem pro probírané technologie a popisuje vývojová prostředí, kterých je možno při realizování aplikací pomocí těchto technologií využít. Třetí kapitola se zabývá návrhem a praktickou realizací webové aplikace jednoduchého redakčního systému v obou výše popsaných technologiích. Redakční systém využívá příslušných databázových systémů a obsahuje základní administrační rozhraní, které je spravováno a obsluhováno pomocí webového rozhraní. Poslední kapitola obsahuje závěrečné zhodnocení a sumarizaci zjištěných závěrů, shrnuje učiněné poznatky a snaží se zodpovědět otázku, v kterých případech dát přednost ASP.NET před PHP a naopak.

Klíčová slova:, PHP, ASP.NET, databáze, web, vývoj

Abstract (English)

This bachelor thesis analyses the possibilities of dynamic web content development by the help of technologies PHP and ASP.NET, attempting to compare both of them and establishes basic differences. All work is divided into the four main chapters. First, introductory part, treat of dynamic web content development, used technologies and initiatress reader to the given problems. The second chapter is already about analysing and comparing the two technologies mentioned above. This part begins by the basic introduction and history of technologies, follows practical part „ installation, setting, configuration", where is reader familiarized with needed development tools for starting each of technology. Further treat of the way of writing server code including syntax of languages, in which is PHP compared with language C# and describes the object orientation of languages. Chapter includes the samples of data access by both of the technologies, whether it uses functions or objects, special libraries etc. Summarises the support of platforms for mull over technologies and describes development environments that can be used to realize applications by the help of these technologies. Third chapter is engaged in design and practical realization of simple editorial system web application in both above described technologies. The editorial system is using appropriate database systems and includes basic content management interface, which is administred and managed through the web interface. Last chapter contains final estimation and summarization of ascertained conclusions, recapitulate downright findings and tries to answer the question, in which cases give priority to ASP.NET before PHP and on the contrary.

Keywords: PHP, ASP.NET, database, web, development

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Rozdíly ve vývoji dynamického webového obsahu při použití technologií PHP a ASP.NET“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č.121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č.140/1961 Sb.

V Brně dne 30. května 2008

……..……..

Radan Jánoš

PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské práce Ing. Petru Kovářovi z Ústavu telekomunikací za velmi užitečnou metodickou pomoc a cenné rady při zpracování bakalářské práce. Dále bych rád poděkoval rodičům za morální a psychickou podporu v době psaní této práce.

V Brně dne 30. května 2008

……..…….

Radan Jánoš

Obsah

1 Úvod

Zatímco ve svých počátcích plnily webové aplikace jen velmi jednoduché informační funkce, dnes jsou schopny zpracovávat značnou část podnikové agendy. Někteří IT odborníci proto přirovnávají jejich nástup k milníku, srovnatelnému například s příchodem architektury klient- server počátkem 90. let. Tyto aplikace lze ovšem postavit na řadě různých technologií, z nichž každá se vyznačuje určitými klady i zápory.

1.1 Vývoj webového obsahu

Web (či World Wide Web, WWW) je jednou ze služeb Internetu, a tak začneme právě zde. Internet je z technického hlediska tvořen desítkami tisíc počítačů, trvale spuštěných a propojených v síti. Počítače v síti Internet pracují jednak jako servery, jednak jako klientské zdroje – servery své služby poskytují, klientské počítače tyto služby využívají. Přestože je služeb na internetu pěkná řádka, můžeme pohled na ně velice zjednodušit: je jím zaslání dat směrem ke klientovi na jeho žádost. Většinou se jedná o data, která jsou uložena na pevném disku serveru, může se však také jednat o data, která jsou programem běžícím na serveru za běhu vytvořena a odeslána. Samozřejmě také cestují data směrem od klienta k serveru – jsou to především veškeré žádosti a dotazy, ale také to mohou být údaje o uživateli, jako je třeba jeho poštovní adresa nutná pro zaslání zboží [1].

Princip webu spočívá především ve zrození klientského programu, který je spuštěn a běží na počítači uživatele. Jedná se o tzv. prohlížeč (anglicky *browser*), který dekóduje data a jednoduché instrukce zaslané serverem, sestaví z nich stránku jako ze stavebních prvků, a tu pak zobrazí ve svém hlavním okně .

Nikdo neurčuje, jakým směrem půjde vývoj webu. Samozřejmě existuje několik konsorcií a výborů, které je spíše sledují a vydávají doporučení. Standardy se dříve rodily opravdu spontánně, tj. někdo novinku vymyslí a zavede do své praxe, tato novinka se natolik zalíbí, že ji začne podporovat mnoho dalších a posléze se stane standardem. Občas vznikaly zmatky či navzájem si odporující standardy, ale právě díky tomuto je vývoj webu velmi rychlý.

Poprvé byla spuštěna služba World Wide Web byla v roce 1990 na půdě výzkumného centra CERN, kdy jsme si vystačili s pouhými třemi technologiemi tvorby webových stránek. První z nich byl jazyk HTML (Hyper Text Markup Language), který sloužil k zápisu stránek. HTML je dodnes (spolu s XML a následovníkem HTML - XHTML) ústřední technologií Webu.

Základní principy a vlastnosti HTML:

- Jazyk HTML je vždy a pouze textový formát. Pokud se na stránce vyskytují obrázky či animace atd., existují na ně odkazy, tj. tato data nejsou umístěna do souboru obsahujícího HTML kód.
- Příkazy tohoto jazyka jsou spolu se svými parametry uzavírány do špičatých závorek, jedná se o příkazy, které nějakým způsobem definují formátování elementů na stránce. Vždy hned za první, otevírající závorkou je jméno příkazu, dále jsou pak jeho parametry.
- V jazyku HTML používáme párové nebo nepárové příkazy. Párový příkaz slouží k formátování elementů stránky – např. nadpis, text, tabulka. Nepárový příkaz se vztahuje na celý dokument nebo na element, který už je sám o sobě přesně vymezený – např. obrázek.

Další nezbytnou technologií je přenosový protokol **HTTP** (Hyper Text Transfer Protocol), který zajišťuje přenos HTML stránek z WWW serveru (místa, kde jsou stránky uložené) do prohlížeče. Původní verze HTTP byly velmi jednoduché. V důsledku zvýšených požadavků postupně vznikly nové verze HTTP, které podporují všechny nejvýznamnější WWW servery a prohlížeče. Dnes se k HTTP přidává další protokol SSL, který odesílaná data šifruje.

Poslední technologií nezbytnou pro implementování služby WWW (aby fungoval) jsou **URL** (Uniform Resource Locator). Každý objekt přístupný na Webu má svojí jedinečnou URL adresu, která slouží k vytváření odkazů na daný objekt.

Spojení těchto tří technologií nenabízí mnoho, umožňuje pouze prohlížení elektronických dokumentům které jsou provázány systémem odkazů. Další vývoj se tedy ubíral k dnešní podobě Webu, který je interaktivně reaguje na požadavky uživatele.

1.2 Dynamický webový obsah

Webové aplikace, které obsahují stránky s dynamicky generovaným obsahem, jsou aplikace, které komunikují s uživatelem nebo jiným systémem prostřednictvím protokolu HTTP. Jedná se o software typu "klient/server", kdy za klienta je považován nejčastěji webový prohlížeč, tedy např. Internet Explorer, Mozilla, Opera. Klientem však můžou být i automatizované systémy, například roboti fulltextových vyhledávačů, ti pak přistupují s pomocí **HTTP** agentů.

Na základě interakce s uživatelem zasílá klient serveru jednotlivé požadavky a následně zobrazuje obdržené webové stránky zapsané zpravidla v jazyce **(X)HTML**. Rozsah aplikací poskytovaných služeb se může pohybovat od jednoduchých úloh, jakými je například kniha hostů, až po sofistikované programy typu komplexních on-line obchodů či bankovních aplikací.

Webová aplikace:

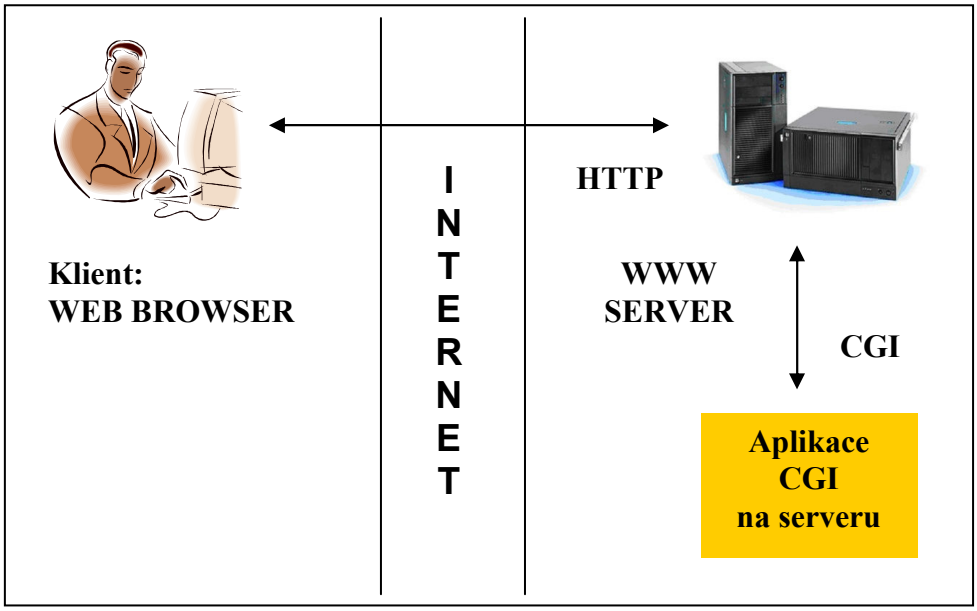
- je postavena na modelu klient/server
- pro komunikaci mezi klientem a serverem používá protokol http
- rozhraním pro koncového uživatele je webový prohlížeč (výjimečně wapový nebo jiný hypertextový prohlížeč)
- požadavky klienta zpracovává webový server (někdy nazývaný HTTP server)

Způsobů, jak se lze na prostředky vývoje webových aplikací dívat, je mnoho, což je dáno velkou rozmanitostí použitelných technologií. Pro účely této práce bude asi nejvhodnější chronologický, respektive „evoluční" pohled.

1.2.1 CGI skripty

Možnost automatického generování byla první inovace stránek, které obsahují informace proměnlivé v čase. Každá HTML stránka představuje soubor uložený na disku WWW serveru, který má své URL. To však nebrání tomu, aby URL ukazovalo na nějaký spustitelný soubor (aplikaci), který vygeneruje HTML stránku.

CGI skripty jsou tedy aplikace na které se uživatel dotazuje v podobě URL. Dotazovaný skript stránku uživateli zpětně vygeneruje (obr.1). Vše probíhá přes tzv. CGI-rozhraní. CGI-rozhraní definuje předání dat mezi www-serverem a CGI-programem (CGI-skriptem).



Obr.1: Zjednodušené schéma předávání dat při CGI skriptování

CGI skripty jsou standardním nástrojem, který podporuje každý slušný web-server. CGI skripty také poskytují výhody z hlediska možnosti výběru jazyka a platformové kompatibility. Jakmile se klient pomocí HTTP obrátí na server, probíhají následující kroky (upozorňuji, že uvedený systém je zjednodušený):

1. Server vytvoří CGI-proces
2. CGI-proces převede HTTP požadavky do proměného prostředí. Dál vytvoří kanál (*data pathways*) mezi serverem a externím CGI-programem
3. Následně CGI-program zpracovává požadavek a použije vytvořený kanál na odpověď serveru. V závěrečné fázi server odešle informace na klienta.

CGI skripty a jejich kód byl místy nepřehledný, což by u větších a rozsáhlejších projektů bylo velkou nevýhodou. Ve své původní verzi bylo CGI rovněž málo výkonné. Při každém požadavku se program musel znovu zavádět do paměti, což bylo problémem při velkém množství nebo frekvenci požadavků. CGI je však standardním nástrojem, který podporuje každý slušný web-server. Jeho skripty jsou programátorům příjemné také z hlediska možnosti výběru jazyka a platformové kompatibility. Tento rys možnosti výběru jazyka je svým způsobem ojedinělý a podobný „komfort“ nabízí až o mnoho let později technologie .NET.

1.2.2 Skriptovací technologie

Tyto technologie využívají vsuvek v HTML kódu. Mezi těmito vsuvkami se nachází kód webové aplikace, který je zpracováván serverem a ten generuje dynamicky výstup již v čistém HTML kódu prohlížeči. Mezi tyto technologie patří PHP, ASP, SSI, SAPI… Tyto technologie jsou dnes velmi rozšířené, výhodou je především jednoduchost zápisu kódu. Velmi jednoduchým a transparentním způsobem píše jak HTML kód, tak určitou funkčnost v místech, kde je potřeba výstup generovat dynamicky. Dnes je nejvíce rozšířenou technologií skriptovací jazyk PHP, je používáno i ASP, ale vzhledem k vyvinutému ASP.NET je již od klasického ASP upouštěno. V dnešní době ovládají skriptovací technologie velkou část trhu, proto také nalézáme velkou podporu těchto technologií, především tedy PHP. Rozšířené

technologie mají automaticky velkou podporu komunit – na Internetu lze najít velké množství už hotových skriptů nebo cenných informací, které by jinak bylo těžké shánět. Základní vlastnosti obou technologií jsou uvedeny v tab.1.

Tab.1: Základní charakteristiky skriptovacích jazyků

Základní jazyky	Označení	Charakteristika	Omezení
CGI	Common Gateway Interface	Základní výměnné prostředí pro příjem vstupu a publikování výstupu jakéhokoliv programu (třeba z Pascalu nebo Perlu)	Je potřeba nějaký další jazyk.
CGI + Perl	Perl je jazyk	Perl umí výborně pracovat se soubory a řetězci, proto je velmi oblíbený pro psaní CGI. Interpretovaný jazyk.	Relativně komplikovaný jazyk, hůře se pracuje s HTML.
CGI + Python	Python je jazyk	Python je přehledný objektový skriptovací jazyk.	Málokdo to umí, je málo dobré literatury, hůře se pracuje s HTML.
SSI	Server-side Include	Prostá vsuvka, která na straně serveru vloží do stránky nějaký objekt, nejčastěji jiný soubor nebo výstup programu.	Nelze předávat parametry, malé možnosti.
ASP	Active Server Pages	Vsuvky v obyčejné HTML stránce, které jsou serverem při odeslání zpracovávány. Umožňují přistupovat k databázím, souborům, poště atd.	Časté bezpečnostní díry, omezení na platformu Windows (na serveru!).
PHP	Hypertext preprocesor		Bez omezení
JSP, Cold fusion atd.	Různé další technologie	Víceméně totéž, co ASP	Malá rozšířenost, z čehož vyplývají nejrůznější trable.
Servlety	Plnohodnotné aplikace v Javě, které generují HTML kód pro klienta		Obtížná instalace, obtížnější jazyk, problémy s češtinou.

Serverové skriptovací jazyky mají také své nevýhody – nejsou příliš vhodné (lépe řečeno tak vhodné a tak připravené) na tvorbu velkých webových projektů a aplikací. Nejsou orientovány tak objektově a nemají tak pokročilý framework jako budou dále probírané objektové technologie ASP.NET. Ve verzi PHP 5 však podporu OOP již najdeme, nemá ovšem srovnatelně pokročilou infrastrukturu a přístup k objektům jako ASP.NET.

1.2.3 Objektové technologie

Problémy a nevýhody skriptovacích technologií se snaží řešit vývojově nejmladší technologie, tedy především ASP.NET. Využívá pokročilou infrastrukturu a tedy velmi obsáhlý .NET Framework. K dispozici jsou vývojářům mnohé již vyhotovené a odladěné nástroje a postupy vývoje webových aplikací, např. šifrování, různé transakce na webu atd. Skriptovací technologie nabízí také mnoho hotových aplikací a nástrojů – někdy však ne tak dokonalých a někdy s potřebou doupravení a přizpůsobení těchto aplikací. Technologie se může opřít o svůj opravdu silný Framework v pozadí a podporují vývoj aplikací objektovým způsobem. To je hlavní a charakteristická odlišnost od technologií skriptovacích.

2 PHP a ASP.NET

2.1 Historie a představení jazyka

2.1.1 Historie PHP

Základy jazyka položil v roce 1995 dánský programátor Rasmus Lerdorf, který vytvořil jednoduchou sadu skriptů v jazyce Perl pro zpracování záznamů o přístupech k jeho webu. Tuto sadu potom nazval Personal Home Page. Historii PHP popisuje webová stránka http://cz.php.net/history/, ze které vybírám několik zajímavých postřehů.

PHP/FI

Zkratka PHP/FI znamená Personal Home Page / Forms Interpreter. Jak rostla potřeba větší funkčnosti PHP vytvořil Rasmus Lerdorf mnohem rozsáhlejší implementaci v jazyku C, která byla schopna komunikovat s databázemi a umožňovala uživatelům vyvíjet jednoduché dynamické aplikace pro Web. Rasmus se rozhodl uvolnit zdrojový kód PHP/FI pro všechny, takže kdokoli ho mohl používat, stejně jako opravovat chyby a vylepšovat kód.

Druhou implementaci PHP/FI 2.0, psanou v jazyku C, začali používat uživatelé po celém světě. Oficiální verze PHP/FI 2.0 byla uvolněna až v listopadu roku 1997. Krátce nato následovala první alfaverze PHP 3.0.

PHP 3

Tato verze se již velmi blížila dnešnímu PHP. Vytvořili ji Andi Gutmans a Zeev Suraski v roce 1997 jako kompletně přepsaný celek. Ve snaze spolupracovat a zahájit budování nad existující uživatelskou základnou PHP/FI se rozhodli Andi, Rasmus a Zeev pracovat společně a prohlásit PHP 3.0 za oficiálního nástupce PHP/FI 2.0. Vývoj PHP/FI 2.0 byl v podstatě zastaven.

Obrovské rozšíření PHP 3.0, poskytnutí pevné infrastruktury pro mnoho různých databází, protokolů a API koncovým uživatelům, přilákalo také mnoho vývojářů, kteří se připojili a vytvořili nové rozšiřující moduly. Tyto okolnosti, podpora objektově orientované syntaxe, mnohem silnější a konzistentnější syntaxe jazyka, se podílely na obrovském úspěchu PHP 3.0. Nový jazyk byl nazván pouze 'PHP', což je rekurzívní akronym - PHP: Hypertext Preprocessor.

PHP 4

V roce 1998 začali programátoři Andi Gutmans a Zeev Suraski pracovat na přepsání jádra PHP s cílem zvýšit výkon pro složité aplikace a zlepšit modularitu kódové báze PHP. Toto jádro nazvali 'Zend Engine' (sestaven z jejich křestních jmen, Zeev a Andi). PHP 4.0, založené na tomto engine, bylo doplněno celou škálou nových prvků, jako je podpora pro mnoho WWW serverů, HTTP sessions, buffering výstupu, bezpečnější způsoby zpracování vstupů uživatele a mnoho nových jazykových konstruktů. Dnes používají PHP stovky tisíc vývojářů a nainstalované PHP hlásí několik milionů serverů - tj. přes 20 % domén na Internetu.

PHP 5

Hlavním přínosem PHP 5 je zlepšení objektové orientace jazyka a syntaxe vůbec. Vydáno bylo v roce 2004. Je řízeno hlavně svým jádrem, Zend Engine 2.0, s novým objektovým modelem a mnoha novými vlastnostmi. Přibyla další rozšíření např. konstruktory, destruktory, obsluha výjimek, vylepšení dereference objektů. Bližší informace o jádru viz stránky http://www.zend.com/zend/future.php.

2.1.2 Jazyk PHP

PHP je jazyk, který pracuje na straně serveru, dovede ukládat a měnit data vašich stránek přímo prostřednictvím prohlížeče. PHP - původně Personal Home Page vzniklo v roce 1996. Od té doby prošlo velkými změnami a nyní tato zkratka znamená Hypertext Preprocessor. Celý produkt je šířen jako freeware - to znamená bezplatně. Pokud se Vám na tomto způsobu šíření softwaru něco nezdá, můžeme se zeptat takhle: „Jaký je v současnosti nejpoužívanější webový server?“. Správná odpověď zní Apache. Apache je používán téměř na 50% všech WWW serverů a je to taky freeware. Dalšími úspěšnými freeware produkty jsou např.: MySQL, Linux a TEX.

Data vašich stránek jsou uloženy na webovém serveru. Nejprve se skript provede na serveru a potom se odešle výsledek prohlížeči. V jazyku PHP je možné vytvořit mnoho aplikací, jako je např. diskuzní fórum, knihu návštěv, počítadlo, anketu, graf a navíc máte možnost propojit vaše stránky s databází, např. MySQL a i dalšími.

Pro vytvoření jednoduché aplikace nazvané „Hello world“, nám postačí takto jednoduchý úsek zdrojového kódu:

```
<?php echo „Hello world“; ?>
```

PHP je opravdu samo o sobě velmi známé, je považováno za nejrozšířenější technologii vývoje webových aplikací. Pro většinu začínajících vývojářů, kteří nemají přílišné zkušenosti s jinými programovacími jazyky a to především objektově orientovanými (C++…), se PHP může stát hlavní technologií pro vývoj jejich aplikací.

Technologii PHP je možné využívat zdarma, spadá do kategorie Open Source a navíc je pro ni velká podpora v oblasti hostingu webových aplikací. Mnoho webových serverů nabízí i slušný hosting zdarma, řádově okolo 50 MB s nainstalovanými technologiemi PHP, webovým serverem Apache a databázemi MySQL.

2.1.3 ASP.NET

První verze ASP (Active Server Pages) byla vydána na konci roku 1996 a kromě podobného data vydání toho měla s PHP společného opravdu hodně. Obě dvě tyto technologie řadíme do tzv. skriptovacích jazyků a technologií popisovaných již v předchozí kapitole.

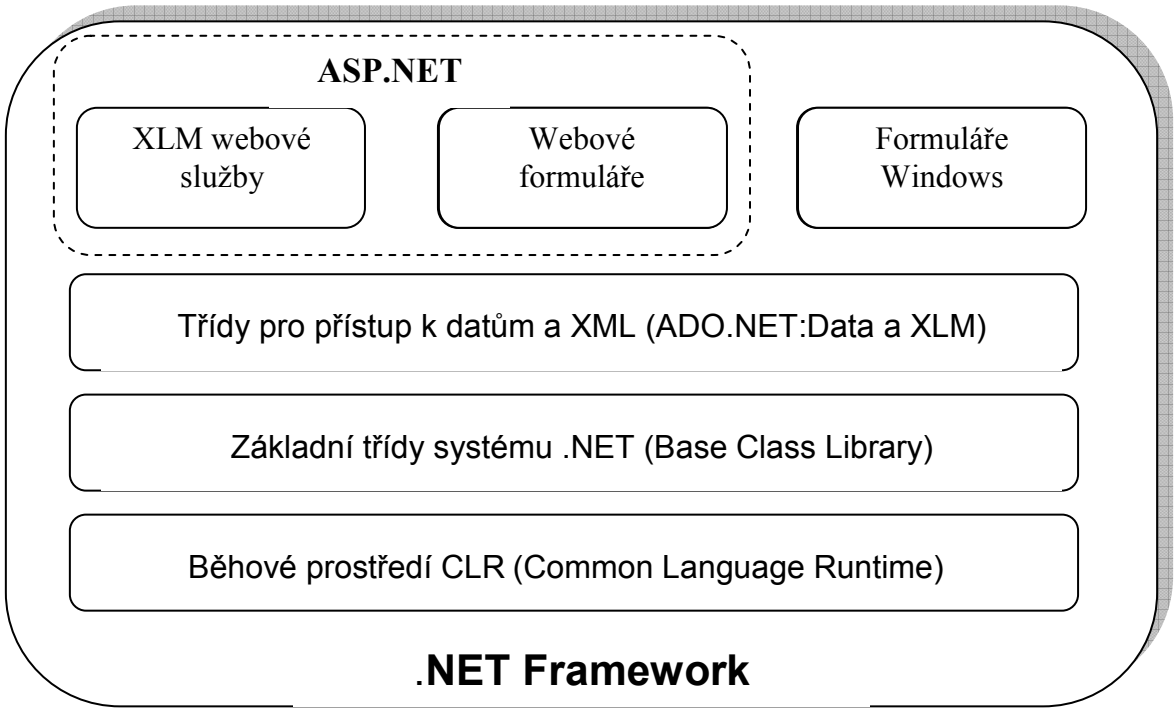
ASP.NET je nová technologie (často se tu používají i termíny jako platforma či prostředí) firmy Microsoft sloužící na tvorbu webových aplikací. Možno říci, že ASP.NET je následovníkem starší ASP technologie, která přinesla, díky jednoduchému modelu generování HTML kódu na straně serveru, doslova revoluci, co se týká tvorby webových aplikací.

Něco podobného, tedy další výrazný krok vpřed v oblasti tvorby aplikací, si firma Microsoft slibuje i od ASP.NET. Ačkoliv je název ASP.NET odvozen od starší technologie pro vývoj webů ASP, obě technologie jsou velmi odlišné.

ASP.NET není žádný nový programovací jazyk, ale technologie, která tvoří nedílnou součást systému .NET Framework [2]. Podle tvrzení Microsoftu vám jeho použití zásadně sníží dobu návrhu i tvorby webových stránek. Vytvoření webové aplikace je při použití .NETu asi 2 až 3 krát rychlejší, než při použití jakéhokoli skriptovacího jazyka. ASP.NET je založen na CLR (Common Language Runtime), který je sdílen všemi aplikacemi postavenými na .NET Frameworku. Programátoři tak mohou realizovat své projekty v jakémkoliv jazyce podporujícím CLR, např. Visual Basic.NET, JScript.NET, C#, Managed C++, ale i mutace Perlu, Pythonu a další. Všechny jazyky pracující na platformě .NET však musí splňovat určité předpoklady dané CLS (Common Language Specification), která obsahuje soubory vlastností a požadavků kladených na tyto jazyky.

Aplikace založené na ASP.NET jsou také rychlejší díky předkompilování do jednoho či několika málo DLL souborů, na rozdíl od ryze skriptovacích jazyků, kde jsou stránky při každém přístupu znovu a znovu parsovány.

.NET framework stojí jako nadstavba nad operačním systémem (obr.2). Obsahuje CLR (Common Language Runtime) zajišťující základní funkcionalitu celého .NET frameworku. Na tomto Common Language Runtimu jsou postaveny všechny další základní knihovny a objekty sloužící například pro přístup k datům (ADO.NET, XML, SQL), k vícevláknovému zpracování aplikací (Threading) k přístupu do sítě a internetu (NET) a k zpracovávání, auditování a monitorování bezpečnosti aplikací (Security).



Obr.2: Architektura .NET Framework

Nad těmito knihovnami stojí už pouze:

a) **Webové služby**

- samotné webové služby
- webové formuláře
- ASP.NET

b) **Uživatelské rozhraní**

- controls (tlačítka, seznamy, posuvníky)
- GDI+ (sloužící pro lepší manipulaci s grafikou, .NET framework totiž podporuje nativně souborové formáty JPG, GIF atd., které si již můžeme sami programově vytvářet)
- služby pro tvorbu klasických desktopových aplikací

Nová technologie ASP.NET tak programátorům ulehčuje přechod od programování klasických aplikací nejen pro Windows, ale i do prostředí webu. Webové stránky tvoří objekty, ovládací prvky (*Controls*), které jsou protějškem ovládacích prvků ve Windows. Při tvorbě webových stránek je tedy možné používat ovládací prvky jako tlačítko (*Button*), nápis (*Label*) a další. Těmto prvkům lze přiřazovat určité vlastnosti, zachytávat na nich události, atd. Tak, jako se ovládací prvky pro Windows samy kreslí do formulářů na obrazovku, webové ovládací prvky produkují HTML kód, který tvoří část výsledné stránky poslané do klientova prohlížeče.

2.2 Instalace, nastavení, konfigurace

2.2.1 PHP

Bohužel PHP je jazyk, který si nevystačí jen s prohlížečem určité verze (třeba jako HTML nebo JavaScript), ale je nutné ho na počítač nainstalovat. Základ tvoří webový server a knihovny. K podpoře PHP je třeba instalovat a konfigurovat server Apache. Instalace PHP se skládá z knihoven PHP (zdrojová data), serveru (Apache) a nakonec třeba i databáze (MySQL). Celá instalace PHP může být provedena ručně, druhou možností instalace PHP je využít nějakého samoinstalačního balíku, které jsou volně ke stažení na internetu. Následující návod se bude týkat instalace a konfigurace potřebných prostředků na platformě Windows.

Dříve než začneme s instalací PHP, je nutné si opatřit něco, čemu se říká *stabilní binární distribuce pro Win32*. To je již zkompilovaný kód PHP, který by měl být funkční na všech verzích 32 bitových Windows, provozovaných na platformě Intel. Je vhodné využívat oficiální distribuce ze serveru *php.net* a především se vyhnout různým speciálně upravovaným knihovnám, které se na internetu hojně vyskytují, např. jejich bezpečnostní riziko je nepoměrně vyšší než u originální distribuce. Jestliže tedy chceme vyvíjet webové aplikace pomocí PHP, je třeba jej stáhnout a nainstalovat, dále je nutný nějaký webový server (nejčastěji Apache), no a případně již zmíněné MySQL:

- **PHP** – stažení PHP balíku např. z domovské adresy www.php.net sekce downloads
- **Apache** – http server Apache z adresy www.apache.org sekce HTTP Server
- **MySQL** – databáze MySQL na adrese http://dev.mysql.com/downloads/

Instalace PHP:

Na disku si předpřipravíme adresář `C:/dev/php`, do tohoto adresáře rozbalíme celý obsah staženého .zip archivu PHP. Následně zkopírujte soubor `C:/dev/php/php.ini-dist` do `C:/windows/php.ini`. Tento soubor je třeba načíst a následně takto upravit:

```
post_max_size = 16M

extension_dir = „c:/dev/php/ext“
upload_max_filesize = 8M
display_errors = On
display_startup_errors = On
extension=php_bz2.dll
extension=php_curl.dll
extension=php_gd2.dll
extension=php_gettext.dll
extension=php_mbstring.dll
```

```
extension=php_mysqli.dll
extension=php_mysql.dll
extension=php_tidy.dll
extension=php_xmlrpc.dll
extension=php_xsl.dll
```

Překopírujeme soubor C:*/dev/php/libmysql.dll* do adresáře C:*/Windows/System32*, aby extension pro mysql pracoval správně. Pokud toto neprovedeme, PHP by při startu nemohlo najít extension *php_mysql.dll*.

Následně provedeme modifikaci systému přes ***Ovládací panely → Systém → Upřesnit → Proměnné prostředí***, a k existující proměnné `PATH` doplňte *`C:/dev/php/`*. Na některých systémech se musíte odhlásit, na Windows XP ne.

Instalace Apache:

Instalaci zahajíme spuštěním instalačního souboru APACHE, jako *NETWORK DOMAIN* a *SERVER NAME*, tedy jméno serveru a doménu zvolíme „**localhost**“. Zvolíme typickou instalaci a pokračujeme dokud instalační program neskončí. Po instalaci si v menu Apache HTTP server zvolíme volbu **stop** a server zastavíme, budeme provádět změnu jeho konfiguračních souborů. Dále si na disku vytvoříme adresář *C:/dev/www* , což bude hlavní adresář sloužící pro ukládání hotových aplikací, všech stránek navazujících na vytvářený web, nazýváme jej též „*root localhostu*“.

Následně přejdeme k modifikaci souboru httpd.conf, nacházejícího se v adresáři C:/Program Files/Apache Group/Apache2conf, otevřeme jej a upravíme takto:

```
//Přidáme řádky
LoadModule php5_module „c:/dev/php/php5apache2.dll“
AddType application/x-httpd-php .php
<Directory „c:/dev/www“>
Options Indexes FollowSymLinks
AllowOverride None
Order allow,deny
Allow from all
</Directory>
//Případně pozměníme a modifikujeme
DocumentRoot „c:/dev/www“
#AddDefaultCharset ISO-8859-1
DirectoryIndex index.html index.html index.php
```

Instalaci MySQL databáze a její konfiguraci již nepopisuji, dále by bylo možné také nakonfigurování a využití nástroje phpMyAdmin. Výše zmíněné nakonfigurování a oddělenou instalaci zmíněných produktů je možné obejít, a nutno podotknout, že pro méně zkušené uživatele je to výhodnější, instalací nějakého balíčku, který veškerá nastavení skrývá již v sobě. Obsahují většinou jak PHP, tak Apache a MySQL pohromadě. Mezi nejznámější takovéto balíčky patří EasyPHP, PHPTriad a další.

Při vývoji stránek umístěných na skutečném serveru, není od věci mít „doma“ stejnou verzi PHP (a MySQL) jako bude na serveru, vyhneme se tak problémům s verzemi, když svůj již hotový projekt umísťujeme na nějaký hosting. Verzi PHP a související informace o konfiguraci zjistíme pomocí následujícího jednoduchého skriptu:

```
<? phpinfo(); ?>
```

2.2.2 ASP.NET

Abysme mohli na svém počítači provozovat .NET aplikace musíme mít nainstalovaný Microsoft **.NET Framework**. Ten lze získat jednoduše ze stránek Microsoftu. K editaci kódu je možno použít Visual Studio.NET, nebo produkty zdarma - ***ASP.NET Web Matrix*** pro .NET Framework 1.1 či ***Visual Web Developer 2005 Express Edition*** pro .NET Framework 2.0. Dále je potřeba mít nainstalován k provozování webových aplikací v ASP.NET samozřejmě nějaký webový server.

Tento problém nemusíme řešit při vývoji, neboť vývojové prostředí, jak *Web Matrix* tak *Web Developer* obsahuje vestavěný webový server tzv. *ASP.NET Development Server*. Pro vývoj je naprosto dostatečný, ale má určitá omezení, která znemožňují jeho použití pro provoz hotového webu. Podporuje totiž pouze lokální připojení (tj. prohlížeč a webový server musí být na stejném počítači), není automaticky spouštěn při startu apod. Pro provozování hotového webu musíme umístit na počítač nejčastěji *Internet Information Server* (IIS). Podporované verze jsou IIS 5.0 (Windows 2000), 5.1 (Windows XP) nebo 6.0 (Windows Server 2003). IIS nabízí vše, co pro provozování webu potřebujeme, a to ve scénáři pro intranet i pro Internet. Opublikování webu na IIS server je velmi snadné. Ze začátku to není nutné, ale postupem času je potřeba mít také nainstalovaný MS SQL server, nebo jednodušší free verzi MS SQL desktop edition.

Aplikace lze provozovat pod operačním systémem Windows 2000 nebo vyšším. Někdy je možné, že narazíme na problém, že tu a tam chybí nějaký Service Pack. Většina takových problémů se dá řešit stáhnutím potřebného nástroje ze stránek Microsoftu, např. dalšími zdroji podpory pro vývojáře http://www.microsoft.com/cze/msdn/podpora/default.mspx na českém MSDN webu. Vhodnou formou pomoci může být pravděpodobně i diskusní skupina *microsoft.public.cs.developer*, přístupná přes:

- **newsreader (NNTP)** - news://news.microsoft.com/microsoft.public.cs.developer
- **webový prohlížeč -** http://support.microsoft.com/newsgroups/newsReader.aspx?lang=cs&cr=CZ&dg=microsoft.public.cs.developer&sloc=cs

Pro činnost webových stránek v ASP.NET je tedy třeba:

- **.NET Framework – potřebné verze ke stáhnutí na adrese** http://www.asp.net/downloads/
- **Webový server – buď lokálně fungující server, který je součástí *Web Matrixu* či *Web Developeru* opět na** http://www.asp.net/downloads/ **, či Microsoft IIS (*Internet Information Serverem)* ze stránek microsoftu** http://www.microsoft.com/downloads/
- **SQL Server 2005 Express Edition – pro případné využití databází** http://www.microsoft.com/sql/default.mspx

2.3 Způsob zápisu kódu, syntaxe

2.3.1 PHP a jeho vsuvky v HTML kódu

Vlastní PHP skript se začleňuje pomocí speciálních, tzv. serverových vsuvek do obyčejného HTML kódu. Skript, který je mezi těmito speciálními znaky vepsán, je prováděn na straně serveru a klientovi je již poslán pouze čistý HTML výstup. Každá instrukce jazyka PHP je ohraničena na začátku <? a na konci ?>. Instrukce mohou být sdružovány do skupin mezi jedním párem <? ?>, když jsou odděleny středníkem. Dokumenty, které mají být interpretovány modulem PHP, musí mít koncovku .php, .php3 a další, případně ty, které máme nastaveny v konfiguračním souboru php.ini (například index.php). Samozřejmě PHP musí umět Váš WWW server, protože to je on, kdo tyto programy provádí.

Způsoby zápisu kódu v PHP:

- `<? echo „Základní způsob vkládání PHP tagů.“; ?>` - Standardní způsob, který se nejčastěji používá. Pro tento způsob je potřeba mít nastavenu v konfiguračním souboru PHP (php.ini) volbu short_open_tag na hodnotu On - standardně je tak nastaveno.
- `<?php echo „Další způsob vkládání PHP do stránek.“; ?>` - Další způsob určený převážně pro generování XML dokumentů.
- `<script language="php"> echo` „Další způsob pro editory, které nepodporují předchozí zápisy.“; </script> - Začlenění PHP skriptů do editorů, které nepodporují předchozí způsoby zápisu.
- `<% echo „ASP tagy.“; %>` - Tento způsob zápisu je povolen pouze tehdy, pokud máme v konfiguračním souboru PHP (php.ini) nastavenu volbu asp_tags na hodnotu On.

Příklad jednoduchého PHP skriptu zobrazujícího text:

Takto je skript zapsán v nějakém souboru s příponou .php

```
<p>
```

Úplně normální text <? echo „a další text generovaný serverem s PHP“ ?>.

```
</p>
```

Výstupem je následující HTML kód vygenerovaný serverem

```
<p>Úplně normální text a další text generovaný serverem
s PHP.</p>
```

Jak je vidno z předchozího přehledu způsobu zápisu kódu, bude zápis pomocí <?php echo ”něco”;?> vyhovovat standardu XHTML a tedy XML validním dokumentem.

```
<?xml version="1.0" encoding="windows-1250" ?>
<!DOCTYPE html PUBLIC „-//W3C//DTD XHTML 1.0 Strict//EN"
„http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" >
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
  lang="en">
<head>
<title>Demo</title>
</head>
<body>
```

```
<?php echo ”něco”;?>
</body>
</html>
```

Takový dokument zcela vyhovuje specifikaci XHTML.Syntaxe PHP se velice podobá syntaxi jazyka C. Jednotlivé příkazy se od sebe oddělují středníkem, složené příkazy se uzavírají mezi složené závorky {}. Proměnné se označují znakem **$** před jejím názvem (např. **$promenna**). Název proměnné musí začínat písmenem nebo podtržítkem, za kterým následuje libovolný počet písmen, podtržítek nebo číslic. Názvy bychom měli také psát bez diakritiky (háčků a čárek), PHP však diakritiku podporuje, a v názvech nesmíme dělat mezery. Není potřeba definovat předem jejich typ - ten se určuje automaticky při přiřazení hodnoty do proměnné během zpracování skriptu serverem.

PHP umožňuje „jednodušší“ zápis kódu, kdy není třeba definovat typ proměnné při deklaraci, její typ bude určen až za běhu. V případě PHP jeho netypovost, nám dává určitou svobodu, avšak mezi velké nevýhody patří jeho chybějící typová bezpečnost. Například při porovnání řetězce s obsahem 10 a proměnné typu int s obsahem 10 bude výsledkem této operace true, což je u typových jazyků nemyslitelné. Výsledek porovnání a zároveň i zajištění prorovnání typů proměnných můžeme ovlivnit použitím operátoru = =.

Tab.2: Příklady escape sekvencí

Znak	**Escape sekvence**
Uvozovky	\“
Znak dolaru	\$
Konec řádku	\n
Návrat na začátek řádku	\r
Tabulátor	\t
Zpětné lomítko	\\

Velmi často potřebujeme do řetězce vložit nějaký znak, který má v PHP speciální význam (uvozovky, znak dolaru), nebo není tisknutelný (konec řádku, tabulátor). K tomu slouží tzv. escape sekvence, které se skládají ze zpětného lomítka a jednoho dalšího znaku (tab.2).

Jazyk PHP je částečně citlivý na velká a malá písmena (case-sensitive) tzn., že rozlišuje velikost písmen v názvech proměnných (**$promenna** je jiná proměnná než **$Promenna**). Ovšem u názvů příkazů a funkcí PHP velikost písmen nerozlišuje. Například **phpinfo()** je tedy to samé jako **PHPInfo(),** stejně tak **echo “něco“** a **eCHo “něco“**.

2.3.2 Serverové ovládací prvky s ASP.NET

Technologie ASP.NET nabízí několik možných forem zápisu kódu. Nejstarším z nich je zápis, který byl používán již u technologie ASP a velmi se podobá zápisu v PHP. Jedná se také o určité vsuvky, speciální znaky, v HTML kódu. K provádění skriptu dochází opět na straně serveru.

Příklad tohoto zápisu:

```
<body>
<% Response.Write("Hello World") %>
</body>
```

Technologie ASP.NET však přichází se zcela jiným a novým přístupem zápisu serverového kódu. Tento nový přístup se opírá o tzv. *serverové ovládací prvky (server controls)*, jež jsou součástí již zmíněného .NET Frameworku. Konkrétně jsou podsoučástí WebForms spadajících pod ASP.NET. Serverové ovládací prvky sdružují veškeré komponenty a vůbec různé objekty, které využíváme na našich stránkách. Nemusí se jednat jen o vizuální komponenty, mezi něž spadají např. Labely, Buttony apod., ale také nevizuální komponenty, které nám zprostředovávají zpřístupnění a zápis do nějaké databáze apod. Tyto objekty vpisujeme do programového kódu jako samostatné tagy, jež mají definované určité vlastnosti a my je můžeme nastavovat. Zvolené komponenty si samy vygenerují potřebný HTML kód, který je dále poslán jako výsledek klientovi. Dokonce se tento výsledný, vygenerovaný HTML kód může lišit v závislosti na použitém prohlížeči, tedy jeho jádře.

Serverové ovládací prvky můžeme rozdělit do těchto skupin:

- HTML Controls, což jsou v podstatě HTML tagy využívající atributu runat="server"
- Web Controls, což jsou předpřipravené prvky, z nichž některé mají svůj obraz v HTML (např. <asp:Label> →) a některé jsou komplexnějšího charakteru (např. <asp:Calendar>)
- User Controls a Custom Controls, což jsou uživatelsky vytvářené ovládací prvky

Příklad jednoduché ASP.NET aplikace zobrazující text:

Takto je aplikace zapsána např. v nějakém souboru s příponou .aspx

```
<body>
<form runat="server">
<asp:Label id="Label1" runat="server"
forecolor="Red">AhojSvěte</asp:Label>
</form>
</body>
```

Výstupem je následující HTML kód vygenerovaný serverem

```
<body>
<span id="Label1" style="color:Blue;">Ahoj Světe</span>
</body>
```

Zdrojový kód obsahuje tag <form runat="server">, jedná se tedy o HTML formulář s novým atributem *runat="server"*, ten ASP.NET napoví, že se má element zpracovat na straně serveru. Dále jsme se zde setkali s elementem <asp:Label>. Ten obsahuje více atributů, tedy

jeho jedinečné id, pomocí něj bychom mohli k tomuto Labelu někde jinde v programu přistupovat, opět parametr *runat="server"*, a nastavení barvy textu na červenou, parametr *forecolor="Red"*. Z HTML výstupu je možné rozpoznat jak ASP.NET generuje použitý label. Obrazem labelu je tedy tag s nastaveným stylem.

Následující ukázka zdrojového kódu je reakcí na událost stisknutí tlačítka (Button):

```
<%@ Page Language="C#" %>
<script runat="server">
  void ukazPozdrav(object sender, EventArgs e)
     {
    pozdrav.Text = „HELLO WORLD!";
     }
</script>
<html>
<head>
<title>Hello World v ASP.NET!</title>
</head>
<body>
  <form runat="server">
  <asp:button id="pozdravit" onclick="ukazPozdrav"
runat="server" text="pozdrav"></asp:button>
    <asp:Label id="pozdrav" runat="server"></asp:Label>
  </form>
</body>
</html>
```

Veškerý skript se píše pod výběr jazyka, do tagu <script> opět musíme napsat, že chceme, aby se vše odehrávalo na straně serveru. Hlavní část HTML kódu neobsahuje žádné serverové vsuvky, pomocí volané funkce měníme objektu *Label* jeho vlastnosti, v našem případě text.

Syntaxe ASP.NET s použitím jazyka C# se v mnoha ohledech podobá syntaxi PHP. Stejně jako v PHP je *case-sensitive*, tedy rozlišuje velká a malá písmena v názvech proměnných. Stejně tak zde platí přibližně stejná pravidla pro názvy proměnných. ASP.NET je však přísně typové. Nelze zde počítat s tím, že by typ proměnné byl dosazen někde při překladu, typ proměnné je třeba nastavit a definovat hned při deklaraci, stejně jak to známe např. z C++ a jiných moderních objektových jazyků. Společný typový systém je důležitou součástí architektury .NET(tab.3).

Tab.3: Srovnání vlastností PHP a ASP.NET

PHP	**ASP.NET**
• **case-sensitive** • **netypové** • **vsuvky v HTML kódu**	• **case-sensitive** • **přísně typové** • **serverové ovládací prvky**

2.4 Objektová orientace jazyků

2.4.1 Zapouzdření

Jak PHP tak ASP.NET (ve všech jazykových mutacích .NET) podporují vytváření objektů a tříd. PHP se však objektovostí jazyka zabývá podrobněji až od verze 5, avšak již ve starodávné verzi 3.0 podporu OOP najdeme. Zapouzdření je jedním z hlavních rysů objektově orientovaných jazyků. Třída je datový typ, který v sobě zapouzdřuje data a funkce - *atributy* a *metody* v názvosloví OOP. Atributy jsou v praxi proměnné primitivních datových typů, například float nebo int, ale také instance jiných tříd - složitější třídy se obvykle skládají z jednoduších tříd. Metody zase zajišťují vlastní funkcionalitu třídy a provádí operace nad jejími atributy. Systém zapouzdření je ještě vylepšen přístupovými modifikátory, které určují, zda jsou či nejsou metody a atributy přístupné z jiných tříd (tab. 4).

Tab. 4: Příklady zapouzdření v PHP a ASP.NET

PHP	**ASP.NET (C#)**

`class MojeTrida {`
`  private $promenna;`
`  //případné další proměnné`
`  function funkce()`
`  {`
`  // kód fce`
`  }`
`}`
	`class MojeTrida {`
	`  private int promenna;`
	`  //případné další proměnné`
	`  public void funkce()`
	`  {`
	`  // kód fce`
	`  }`
	`}`

Při letmém pohledu na zdrojové kódy vytvořených tříd pomocí obou jazyků jsou si v zápisu velmi podobné. Jazyk C# je však opět přísně typovým jazykem a tak zapouzdřené proměnné a fce musejí mít definován svůj typ. Dále pak u zapouzdření rozlišujeme tzv. *modifikátory přístupu* ke členům třídy. Přístupové modifikátory umožňují programátorovi definovat bezpečnostní pravidla pro přístup k členům třídy. Například je vhodné datové složky třídy definovat s modifikátorem private a tím znemožnit jejich přímou modifikaci. Ta se potom provádí nepřímo pomocí veřejných (public) metod dané třídy…

Ve většině OOP jazyků jsou definovány tyto přístupové modifikátory:

- public - neomezený přístup
- protected - přístup je omezen na danou třídu a třídy odděděné
- private - přístup je omezen pouze na obalující typ (danou třídu)

Jazyk C# navíc přidává modifikátory *protected internal* a *internal*. V PHP jsou, ale až od verze 5, výše zmíněné tři hlavní modifikátory přístupu. Modifikátor musí být u PHP před deklarací proměnné uveden. PHP naráží na malý problém se zpětnou kompatibilitou starších skriptů, kdy k označení členské proměnné třídy bylo využíváno klíčové slovo *var*, nyní je automaticky ekvivalentem k modifikátoru *public*. U C# a ostatních jazyků .NET je však implicitním modifikátorem *private*. Pokud u C# tedy žádný modifikátor před členem třídy nedefinujeme je automaticky nastaven na *private*.

2.4.2 Dědičnost

Dědičnost je mechanismus OOP, který umožňuje jednoduchou znovupoužitelnost existujícího kódu, od každé třídy lze oddědit třídu novou. Dědičnost je schopnost definovat určité třídy jako předky a jiné zase jako potomky, potomci dědí po svých rodičích členské proměnné a metody. Nová třída má přístup ke všem členům svých předků pokud nejsou označeny přístupovým modifikátorem *private*.

PHP a ASP.NET se v podpoře dědičnosti prakticky vůbec neliší. Obě dvě technologie podporují pouze jednoduchou dědičnost. Opakem bývá vícenásobná dědičnost jak ji známe např. z C++, kdy lze dědit vícekrát a často existuje speciální třída, která je automaticky předkem všech dalších tříd. U PHP rozlišujeme klíčové slovo *final*, které je ekvivalentem klíčového slova *sealed* v jazyce C#. Od takto označené třídy již nelze dědit.

2.5 Přístup k databázovým systémům

Každá rozsáhlejší webová aplikace či tedy rozsáhlejší web využívá vždy nějakým způsobem možnosti ukládání, načítání a zobrazení dat z databází. Komunikace s databázemi a datovými zdroji je snad jedna z nejdůležitějších prací webových aplikací. Jak ASP.NET tak PHP obsahují pro datové zdroje a databáze bohatou podporu. Avšak každá z technologií přistupuje k problému trochu odlišně. PHP komunikuje s databázemi pomocí volání určitých funkcí obsažených ve speciálních knihovnách. Následující příklad je ukázkou připojení a výpisu dat z databáze mSQL, avšak PHP podporuje mnoho jiných datových zdrojů, především databáze MySQL a dále PostgreSQL, Oracle, Firebird/Interbase atd. Samozřejmě zde najdeme také modul pro ODBC a podporován je i zajímavý projekt SQLite.

```
<?php
/* Connecting, selecting database */
$link = msql_connect(,localhost‘, ,username‘, ,password‘)
    or die(,Could not connect : , . msql_error($link));

msql_select_db(,database‘, $link)
    or die(,Could not select database‘);

/* Issue SQL query */
$query = ,SELECT * FROM my_table‘;
$result = msql_query($query, $link) or die(,Query failed : ,
. msql_error($link));

/* Printing results in HTML /
echo „<table>\n“;
while ($row = msql_fetch_array($result, MSQL_ASSOC)) {
    echo „\t<tr>\n“;
    foreach ($row as $col_value) {
        echo „\t\t<td>$col_value</td>\n“;
    }
    echo „\t</tr>\n“;
}
echo „</table>\n“;
```

```
  / Free result set /
  msql_free_result($result);

  / Close connection */
  msql_close($link);
  ?>
```

Výše zmíněný příklad tedy znázorňuje připojení k databázi mSQL pomocí PHP. K připojení je využito funkcí *msql_connect, msql_select_db* a dalších. Tyto funkce nám zajišťují připojení k databázi na základě nějakého uživatelského jména a hesla, výběr databáze, její části, jak mají být data řazena apod. Každá z nich je ošetřena klíčovým příkazem *or die* např. v případě neúspěchu připojení. K výpisu údajů dochází cyklem *while-do* do tabulky tvořené standardními HTML tagy. Po dokončení práce s databází dojde k jejímu uzavření.
ASP.NET přistupuje k datovým zdrojům pomocí součásti .NET Frameworku ADO.NET. ADO.NET je reprezentována speciálními třídami ve jmenném prostoru System.Data. ASP.NET vylepšuje podporu XML a snaží se oddělit připojení k datům od manipulace s nimi. Následující příklad bude také připojení k mSQL databázi a výpis jejího obsahu avšak pomocí ASP.NET.
Nejdříve vytvoříme soubor s příponou .aspx a do něj vložíme tuto část kódu:

```
...
<form runat="server">
<asp:DataGrid id="MojeMrizka" runat="server" />
</form>
 ...
```

Do souboru jsme tagem *<asp:DataGrid>* umístili serverový ovládací prvek datové mřížky. Tento prvek nám skvěle postačí k tomu abychom do něj vepsali příslušná data. Dále pak doplníme událost *Page_Load* (ta se automaticky spouští při načtení stránky).

```
private void Page_Load(object sender, EventArgs e)
{
DataSet ds = new DataSet();
SqlDataAdapter adapter = new SqlDataAdapter („select * from
database", „server=localhost;database=
my_table;uid=username;pwd=password");
  adapter.Fill(ds);
  MojeMrizka.DataSource = ds;
  MojeMrizka.DataBind();
}
```

Objekt DataSet je objektem ADO.NET. Tento objekt je určitou "databází" a může obsahovat prvky jako jsou např. tabulky. Datových adaptéry vykonávají patřičné SQL příkazy a jsou prostředníky mezi objektem DataSet a databází. Příkaz *adapter.Fill(ds);* provede otevření spojení, získání dat, a uzavření práce s databází. Poslední dva příkazy již provádí výběr dat do mřížky a svázání mřížky s těmito daty.

2.6 Další související záležitosti

2.6.1 Podpora platforem

Internet je heterogenním prostředím s velkým množstvím různých platforem, ať už se to týká operačního systému, či webového serveru nebo kombinací obojího. Největší předností PHP je právě jeho platformová nezávislost, je tedy spustitelné a rozběhnutelné téměř všude. Má širokou podporu jak co se týče operačních systémů (Unix, Linux, Windows, MacOS), tak webových serverů (Apache, IIS a další). Pro PHP je však považováno jako domovské prostředí operační systém Linux v kombinaci s webovým serverem Apache.

Naproti tomu ASP.NET, jako technologie vyvinutá Microsoftem, je „doma“ na platformě Windows + IIS (*Internet Information Server*). Podpora ostatních platforem je zde značně problematická, objevují se však různé aplikační porty na jiná prostředí např. projekt *Mono*. *Mono* je tedy implementace části .NET Frameworku, která je obsahem oficiální specifikace vydané prostřednictvím ECMA, některých dalších částí .NET Frameworku, které nebyly Microsoftem plně zveřejněny a některých dalších knihoven, které bylo třeba vytvořit kvůli portaci .NET Frameworku na jiné platformy nebo knihovny, které rozšiřují funkcionalitu Mona o další možnosti (například knihovny pro spolupráci s Gnome, rozšířená podpora šifrování nebo XML). Platformy pro obě technologie ukazuje tab.6.

Tab.6: Platformy PHP a ASP.NET

PHP	**ASP.NET**
• **Unix, Linux, Windows, MacOS** • **Apache, IIS** • **„platform-independent“**	• **Windows** • **IIS** • **Mono, mód Apache ???**

2.6.2 Vývojové prostředí

Práci při vývoji webových aplikací nám může nejen zpříjemnit, ale velice zrychlit a zefektivnit některé z vývojových prostředí. Aplikace jak v PHP tak pomocí ASP.NET lze však vyvíjet třeba jen s pouhým notepadem a jestliže chceme veškerému dění a kódu detailně rozumět, měl by tento typ vývoje smysl. Kvalitní vývojová prostředí nám však poskytují mnoho užitečných funkcí, které vývoj usnadňují. Zvláště pro vývoj PHP existuje velké množství vytvořených prostředí, avšak mnoho z nich nevyhovuje moderním požadavkům kladeným na IDE (IDE, *Integrated Development Environment*, vývojové prostředí).

Požadavky kladené na IDE:

- co nejlepší podpora při psaní zdrojového kódu – zvýraznění syntaxe, možnost automatického dokončování názvů (systémové knihovny, definované proměnné …)
- integrace nápovědy, informačních zdrojů
- integrace všech potřebných nástrojů přímo v IDE, bez nutnosti hledat a kopírovat řešení odjinud
- podpora ladění projektu (Debugging)
- případně možnost využití nástroje WYSIWYG

Mezi nejlepší IDE pro PHP patří český PSPad, dále o stupínek lepší jEdit, který je také Open Source. Nejkvalitnějším IDE je Zend Studio Professional za zhruba 299 dolarů. Má nejlepší podporu při psaní kódu, hlídá syntaxi a navíc obsahuje nástroj pro ladění a hledání chyb v kódu. Zend studio je také nezávislé na platformě.

Podporované operační systémy a platformy pro ZEND:

- Windows x86 2000, XP, 2003, Vista
- Linux x86
- Linux x86-64
- Mac OS X Power / Intel 10.4

U ASP.NET máme na výběr především ze dvou prostředí poskytovaných zdarma. Těmi jsou *Web Matrix* (určen pro .NET Framework 1.1) a *Visual Web Developer 2005 Express* (určen pro .NET Framework 2.0). Oboje prostředí poskytují WYSIWYG návrh, obsahují pokročilé nástroje pro práci s databázemi, zvýrazňují syntaxi, avšak integrace nápovědy a funkce automatického dokončování je obsažena až ve Web Developeru. Žádné z těchto prostředí nemůže konkurovat novému VISUAL STUDIO 2005. Přichází i nová řada *Visual Studio Express* (zdarma) speciálně navržena pro začátečníky, amatéry a ostatní, kteří se chtějí naučit programovat ve Windows nebo tvořit interaktivní webové aplikace pro Internet.

Firma Microsoft určuje Visual Studio 2005 především pro programování klasických desktopových, serverových, webových (ASP.NET) i mobilních aplikací na platformách Windows a .NET 2.0. Z programovacích jazyků jsou k dispozici Visual C++ (nativní i řízené), Visual C#, Visual Basic a Visual J#. Produktová řada Visual Studia 2005 zahrnuje: VS Standard Edition, VS Professional Edition, VS Tools For Office, VS Team System a minimalistickou edici VS Express (který je na jeden rok zdarma).Plusy a minusy Visual Studia 2005 ukazuje tab. 7.

Tab.7 :Vlastnosti Visual Studia 2005

Plusy	Minusy
+ Správa zdrojových souborů + Řízení práce v týmu + Nové vlastnosti .NET Frameworku + Analytické a testovací nástroje	- Systémové požadavky - Přenos aplikací na .NET Framework 2.0 není vždy spolehlivý

3 Redakční systém

3.1 Úvod

Součástí mé bakalářské práce byla také povinnost navrhnout a realizovat aplikaci středního rozsahu pomocí technologií PHP a ASP.NET. Na základě analýzy demonstrovat základní rozdíly obou platforem při vývoji této aplikace. Rozhodl jsem se pro naprogramování webové aplikace jednoduchého redakčního systému. Tato kapitola pojednává právě o vývoji tohoto redakčního systému od stanovení základních otázek, co by měl tento redakční systém umět a jaké nároky splňovat, přes vlastní návrh aplikace až po její řešení. Ke zvládnutí tohoto úkolu bylo zapotřebí zvládnout základní programovací techniky obou jazyků. Pro úplnost uvádím, že v případě technologie ASP.NET jsem volil jazyk C#. To, že jsou programovací jazyky v mnoha oblastech tykajících se syntaxe, typovosti, přístupu k datovým zdrojům a jiných záležitostech zcela odlišné, shrnují již předchozí kapitoly.

Stejně jako mnoho jiných uživatelů i já jsem začínal s jazykem PHP, a to bez objektového přístupu k naprogramování různých méně či více složitých aplikací. Vždy jsem však s tímto přístupem slavil úspěchy, veškeré aplikace se mi podařilo uvést do plně fungujícího režimu se všemi předtím stanovenými nároky. Netvrdím však, že PHP ve verzi 5 již neobsahuje celkem dosti propracovaný objektový model, ostatně veškeré podrobnosti jsou již zmíněny výše v práci. Také při programování tohoto redakčního systému v PHP jsem volil neobjektový přístup. Avšak nerozhodl jsem se jen na základě toho, že jsem již na tento způsob zvyklý, ale požadavkem bylo opět vytvořit především funkční aplikaci, a to se mi podařilo. Pokud bych dostal stejné zadání od klienta v praxi i on by se většinou nezajímal, kterých programovacích technik jsem při vývoji jeho aplikace využíval, a zda-li využívám nejnovějších standardů a knihoven. U technologie ASP.NET je již situace zcela odlišná. Je koncipována již od začátku na objektovém designu a využívá moderních objektových jazyků. To jak která z technologií přistupuje k databázím, které vestavěné prvky a objekty např. ASP.NET nabízí k výstupu a tedy výpisu informací, celkovou strukturu aplikace, včetně administrace, přihlašovaní autorů a řešení jejich rolí se budu snažit nastínit v následujících kapitolách.

3.2 Instalace potřebných komponent

Ještě než přejdu k nastínění základních stanovených požadavků na aplikaci je potřeba zmínit, kterých systémových platforem, technologií a jaká vývojová prostředí jsem využíval. Vývoj probíhal jak u PHP tak ASP.NET na operačním systému Windows Vista. K rozběhnutí PHP bylo třeba nainstalovat a nakonfigurovat samotné PHP, jako webový server jsem zvolil Apache, no a v neposlední řadě MySQL databázový systém. Dá se tedy říci, že jsem nainstaloval jednu z nejčastějších, ale zároveň i nejstabilnějších a odladěných konfigurací.

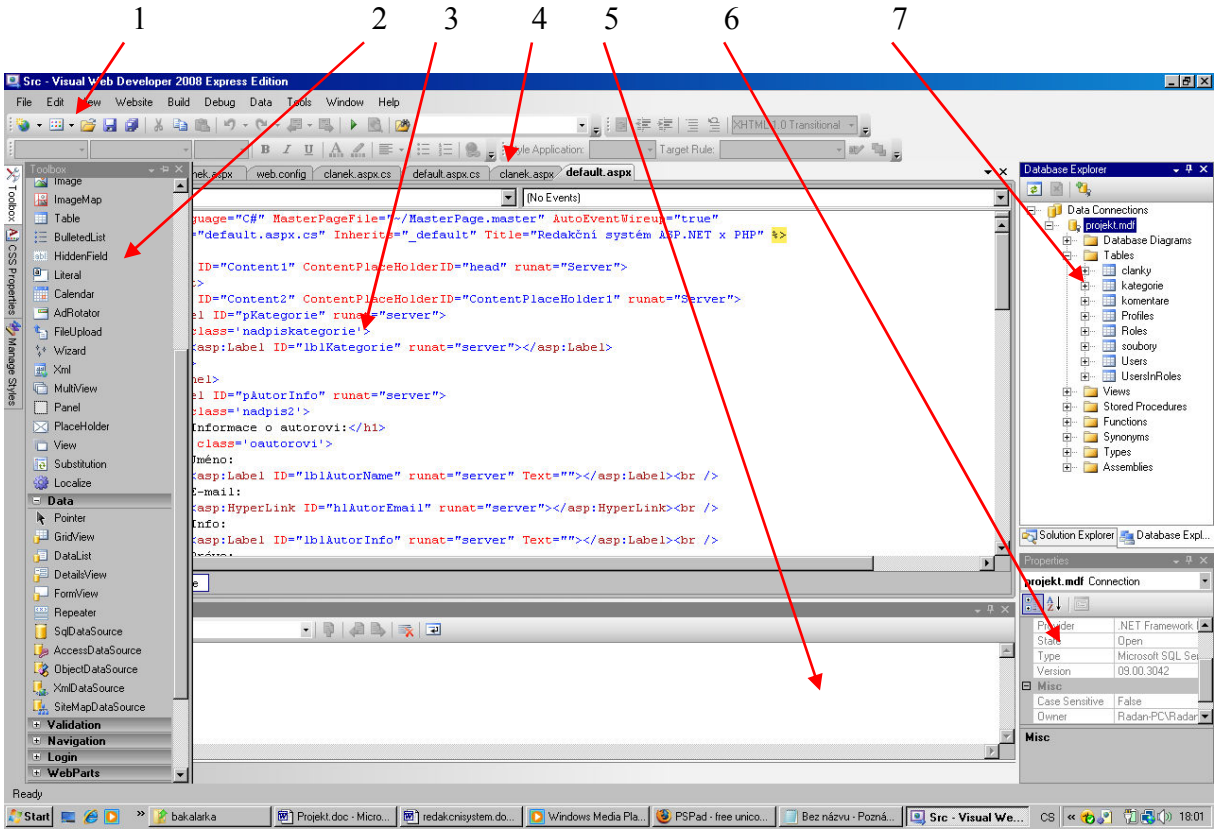
Veškeré podrobnosti ohledně instalace a některých nastavení jsou již probírány v části „instalace, nastavení, konfigurace“. K přístupu do databáze, a veškerým operacím spojeným s vytvořením a úpravou databáze jsem použil nástroj phpMyAdmin kompletně napsaný v jazyce PHP. Umožňuje správu obsahu databáze MySQL prostřednictvím webového rozhraní. Nástroj je schopen vytvářet/rušit databáze, vytvářet/upravovat/rušit tabulky, provádět SQL příkazy a spravovat klíče. Jedná se o jeden z nejpopulárnějších nástrojů pro správu databáze. Program je ke stažení na adrese http://www.phpmyadmin.net/, v předchozí části věnované instalaci a nastavení PHP jsem se o nástroji nezmínil. Dále zde uvedu textový editor PSPad používaný ke psaní samotného zdrojového kódu. PSPad je freewareový editor zdrojových kódů mnoha programovacích, skriptovacích a značkovacích jazyků včetně PHP. Kromě zvýraznění syntaxe obsahuje také spoustu užitečných interních nástrojů. K vývoji pod technologií ASP.NET jsem se rozhodl pro freewareový Visual Web Developer 2005 Express

Edition. Tento nástroj společnosti Microsoft je třeba nainstalovat společně s .NET Frameworkem 2.0 a nejlépe SQL Serverem 2005 Express Edition v následujícím pořadí:

- .NET Framework 2.0
- SQL Server 2005 Express Edition
- Visual Web Developer 2005 Express Edition

Uživatel má však možnost nainstalovat jednotlivé součásti přímo z webu. Stačí si z webu Microsoftu stáhnout program vwdsetup.exe, ten již zajistí stažení potřebných součástí a následně jejich instalaci.

Po spuštění programu máme před sebou rázem celé vývojové prostředí se všemi nástroji potřebnými pro vývoj pohromadě. V rámci jednoho prostředí píšeme jak zdrojový kód, tak vkládáme potřebné komponenty, spravujeme, tvoříme a upravujeme databázi. Po kliknutí na start aplikace proběhne kompilace kódu a spustí se interní server, který je součástí prostředí. Výsledek se nám zobrazí v automaticky otevřeném okně Internet Exploreru na adrese *localhost*. Chvíli trvá než se uživatel naučí s programem pracovat a osvojí si jeho základní rysy. Ovšem jak již bylo řečeno, obsahuje vše pohromadě, na rozdíl od PHP, kde je zdrojový kód psán pomocí nějakého z editorů a přístup k databázi je zajištěn jiným nástrojem, stejně tak webový server je potřeba před testováním spustit zvlášť. Ve Web Developeru můžeme editovat vlastnosti objektů na záložce Properties, máme k dispozici paletu Toolbox, z níž přidáme do stránky jednoduše standardní prvky, html prvky, komponenty pro práci s databází apod. Podpora ladění tzv. „debugging“ je další skvělou vlastností Web Developeru, kdy můžeme aplikaci krokovat a odhalovat tak chyby, a to především u rozsáhlejších projektů, kde nemusí být snadné danou chybu nalézt.



Obr. 3: Základní rozdělení okna programu Web Developer

Rozdělení okna (obr.3):

1. Hlavní nabídka
2. Paleta Toolbox – standardní prvky, HTML prvky, prvky pro práci s daty, prvky pro validaci…
3. Hlavní okno zdrojových kódů
4. Záložky otevřených souborů, tabulek…
5. Okno výstupních informací po kompilaci kódu – např. chybové hlášky…
6. Okno vlastností daných vybraných objektů, změna se automaticky projeví ve zdrojovém kódu.
7. Průzkumník jak databáze, tak kořenové složky webu

Dále bych rád podotknul, že PC s Windows Vista, na kterém vývoj probíhal, sestával z moderních a výkonných hardwarových komponent, konkrétně procesoru Intel Core 2 Duo T8100, 3072 MB RAM paměti. Měl jsem k dispozici také starší PC s AMD Athlon 2800 XP a 256 MB RAM. I na tento starší stroj jsem nainstaloval výše zmíněné součásti ve stejné konfiguraci. Vývoj v PHP nebyl na tomto PC žádným problémem, bylo tedy zcela jedno, zda bych celou aplikaci programoval na jednom či druhém PC, nijak by mne méně výkonné PC časově neomezovalo apod. U ASP.NET a tedy Web Developeru se již na starším PC vyskytl problém. Spuštění aplikace trvalo podstatně delší dobu, docházelo k „zatuhnutí“ i pádu aplikace. Na PC jsem musel mít spuštěnu pouze tuto aplikaci, a výkonnost procesoru hrála určitě roli i při době kompilace a zobrazení výstupu. Není se však čemu divit, Microsoft udává tyto minimální požadavky pro chod aplikace:

- Minimální: 1.6 GHz CPU, 192 MB RAM, 1024x768 display, 5400 RPM hard disk space
- Doporučené: 2.2 GHz or higher CPU, 384 MB or more RAM, 1280x1024 display, 7200 RPM or higher

Pak si pokládám otázku, kterou z technologií zvolit při snaze založit firmu zabývající se vývojem dynamických webových aplikací? Svým vývojovým pracovníkům musím zajistit především nástroje potřebné k vývoji. Tyto nástroje jsou jak hardwarového charakteru tak softwarového. Jak již bylo uvedeno, aplikace v PHP lze vyvíjet na o mnoho méně výkonných strojích a vývojář tak nebude zdržen nějakými časovými prodlevami či dokonce pády aplikací. Pravdou však je, že dnes jde vývoj výpočetní techniky obrovskou rychlostí kupředu, a tak se výkonné stanice dají pořídit za velmi slušné peníze. Stále by však stanice potřebné k vývoji pod PHP přišly na méně financí. Na pořízené stanice musíme nainstalovat a vývojářům zajistit nezbytný software. V PHP lze vyvíjet zcela zdarma, je open source projektem, jednoduše jej stáhneme z webu a nainstalujeme. Navíc je technologií nezávislou na systémové platformě a je tedy možné jej provozovat na bezplatných distribucích Linuxu. ASP.NET je však „doma“ na platformě windows, mód a projekt Mono zmíněný již dříve se ovšem také zdokonalil a je již možné vyvíjet i na systémech Linux. Nejstabilněji se nám bude zatím chovat ASP.NET na platformě Windows a zajistíme si tak bezproblémový chod aplikací. Windows jsou však platformou placenou, a je nutné zaplatit za její licenci. Osobně bych se také snažil poskytnout programátorům jen to nejlepší vývojové prostředí pro ASP.NET a tím je bezesporu Microsoft Visual Studio. I za něj je potřeba zaplatit, ale proč nezaplatit za tak výkonný nástroj. Nic takového u PHP opravdu nenajdeme a dovolím si tvrdit, že při rozsáhlých zakázkách v ASP.NET si na sebe Visual Studio brzy vydělá.

3.3 Požadavky na aplikaci

Nežli se pustíme do návrhu a celkového řešení aplikace redakčního systému, je třeba si stanovit a ujasnit, jaké nároky by měl takový systém splňovat a co by měl umět. To nejpodstatnější se dá shrnout do následujících bodů:

- články – svým čtenářům chceme poskytnout informace v podobě článků, články tedy budou nutností. Dále je třeba, aby vyhovovaly určitým kritériím a obsahovaly údaje jako název, úvodní informace, vlastní obsah, datum vydání apod.
- kategorie – články budeme chtít rozdělit do kategorií dle obsahu článku.
- autoři – možnost třídit články dle jejich autorů.
- komentáře – umožnit čtenářům vyjádřit svůj názor pomocí přidání komentáře k danému článku. Budeme zaznamenávat údaje jako jméno, email, obsah komentáře, datum a čas komentování případně IP adresu.
- administrace – mít možnost nějakým způsobem spravovat veškerý obsah webu, nejlépe přímo přes webové rozhraní.
- ostatní funkce – případné další rozšiřující funkce, vznikající při návrhu a řešení projektu.
- design – dopředu si přibližně rozmyslet strukturu webu, včetně grafické vizuální představy.

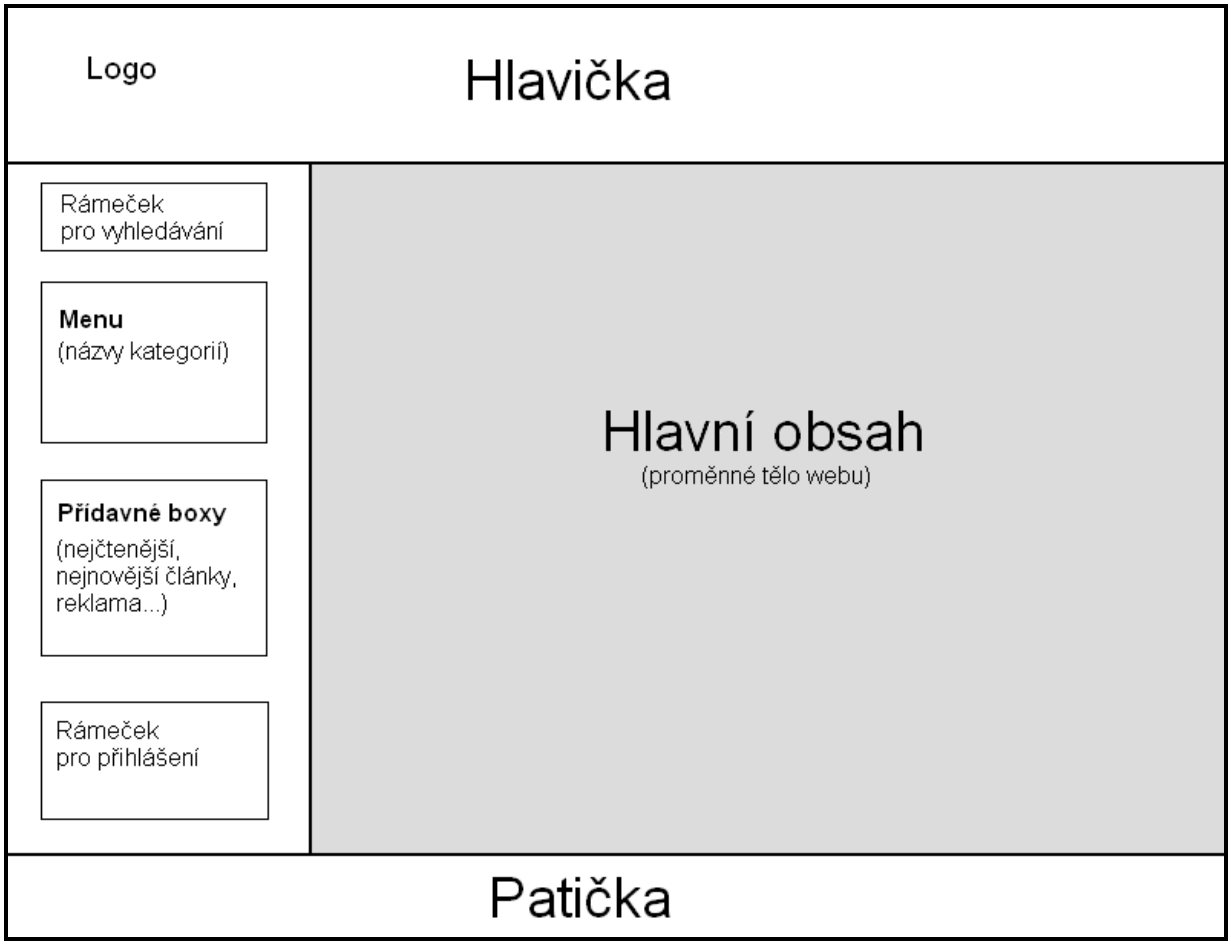
Jakmile jsem si přibližně rozmyslel strukturu webu, vyvstala na povrch otázka „kam veškeré články a jejich obsah ukládat, jak je přidávat na stránky, kterak bude vše souviset s administrací?“. Hned mi bylo jasné, že je potřeba zavrhnout nahrávání souborů stránek s články v HTML na server, tak jak některé redakční systémy dříve i fungovaly. Pokud bychom tímto způsobem přidávali zprávy na server několikrát denně, stalo by se nahrávání stránek na server a jejich úprava noční můrou pro každého správce. Potřebuji dynamický webový projekt s možností administrace, a to i takové administrace, kdy vkládat a upravovat svoje články by mohli i autoři neznalí jazyka HTML. Ideální by tedy byla možnost spravovat obsah našeho webu přímo přes webové rozhraní, aniž bychom potřebovali nějakých jiných nástrojů. Budeme moci poskytnout našim čtenářům nejnovější informace v několika minutách, navíc i tehdy, pokud se zrovna nenacházíme na svém pracovišti a nemáme k dispozici svoje PC. Stačí se například přihlásit z jakékoli internetové kavárny a začít s psaním článku. Bude se jednat o jednoduchý redakční systém CMS (content management system), tedy systém pro správu obsahu webu. Takový systém je definován jako internetová služba, jež je provozována na serveru pomocí technologií jako např. PHP či ASP.NET a databází, za účelem zjednodušení funkcí konkrétního systému, včetně získání informací o návštěvnících daného webu. Databáze bude mít určitou strukturu a obsahovat patřičná data. Redakční systém se tedy hodí zejména tam, kde dochází k časté aktualizaci webového obsahu, kdy se jedná o tzv. blogy, informační servery, e-zine, firemní stránky apod. Dále bude nutné počítat alespoň se základním zabezpečením, stanovit základní pravidla pro vkládání nových informací a stanovit role administrátor a redaktor. Článek poslaný redaktorem by měl projít kontrolou a následným schválením od správce webu. Teprve potom může být definitivně publikován. Jakmile jsem si stanovil alespoň tyto základní požadavky na aplikaci, přešel jsem již k samotnému návrhu a vlastnímu řešení. Těmito záležitostmi se budou zabývat následující kapitoly.

3.4 Vlastní návrh

V této části se budu zabývat návrhy řešení problémů stanovených výše. Nežli jsem se pustil do psaní nějakého zdrojového kódu, tak jsem se snažil problémy promyslet a navrhnout způsob jejich řešení. Nejprve jsem se zamyslel nad strukturou webu, co a především kde se bude na stránce zobrazovat. Poté jsem také navrhl potřebnou databázi se všemi tabulkami a nutnými sloupci a k nim odpovídající datové typy. V ASP.NET by bylo možné stanovit předem i tzv. „Stored Procedures“ neboli „uložené procedury“, tedy předem promyšlené sql dotazy na databázi. To by ovšem vyžadovalo značné zkušenosti a já se spíše přikládám k variantě stanovit dotazy tam kde potřebuji až při vývojovém řešení aplikace.

3.4.1 Struktura webu, uspořádání grafických prvků

Na obrázku 4 máme grafické uspořádání webu s nezbytnými součástmi. Na takto vzniklou šablonu uspořádání je třeba také navrhnout design aplikace, tedy grafický vizuální návrh. Obvykle designer vytvoří maketu webové stránky pomocí profesionálních grafických programů (např. Adobe Photoshop, Paint Shop Pro). Z takto připravené makety by již měla být patrná finální podoba stránky, tedy ještě předtím, než se pustíme do jakéhokoli programování. Jakmile jsme s maketou spokojeni, je třeba naprogramovat HTML kód s případným využitím kaskádových stylů CSS. Výstupem takto napsaného kódu by měla být téměř stejná vizuální podoba stránky zobrazené v prohlížeči jako makety.



Obr 4: Grafické uspořádání prvků na stránce

Samotný HTML kód, který je již aplikačním řešením a stejně tak CSS styly zde uvádět nebudu. Stejný HTML kód včetně CSS jsem použil při řešení aplikace u obou technologií. Nejprve jsem aplikaci vyvíjel v PHP, a poté jsem se zamyslel, zdali mohu daný kód použít i při vývoji v ASP.NET. Ze šablony grafického uspořádání prvků je také hned vidět, že část „hlavní obsah“ (na šabloně podbarvena šedě) bude částí proměnnou. Zde se budou vypisovat, zobrazovat články a ostatní informace. Ostatní části jako „hlavička, sloupec s menu a boxy, patička“ budou při zobrazování jednotlivých stránek neměnné. Bude tedy zapotřebí zajistit sdílení společného vzhledu mezi více stránkami. Nyní trošičku odbočím od návrhu a v podstatě zde nastíním již řešení tohoto problému u obou technologií. Začnu technologií PHP, kde řešení spočívalo v rozdělení struktury stránek do dvou souborů s názvy *hlava.php* a *pata.php*. Tyto soubory byli v každém z modulů, tedy v jednotlivých stránkách obsluhujících jednu z událostí, includovány. Vše lze demonstrovat následujícím kódem:

```
<?php
include "hlava.php";
?>
<?php
//zde se nachází kód obsluhující jednu z událostí
?>
<?php
include "pata.php";
?>
```

Příkaz *include* vloží do právě zpracovávané stránky obsah souboru jehož adresa je za příkazem uvedena v uvozovkách. Efekt bude zcela stejný, jako kdyby byl obsah includovaného souboru umístěn přímo ve stránce. Technologie ASP.NET se k danému problému staví již zcela odlišně. Ve verzi 2.0 zavádí nový prvek, tzv. „vzorové stránky“, které umožňují definovat společné oblasti (např. hlavička, menu, patička) sdílené všemi stránkami.Vzorová stránka dovoluje umístit kód s uspořádáním stránky do jediného souboru, od kterého budou vizuálně dědit všechny ostatní stránky. Tyto stránky umístí svůj vlastní obsah do řídících prvků typu *ContentPlaceHolder*. Vzorová stránka je tzv. *Master Page*, je velmi podobná stránce standardní, v horní části má místo direktivy *@Page* direktivu *@Master* a deklaruje jeden nebo i více řídících prvků typu *ContentPlaceHolder*, do nichž mohou stránky typu .aspx umísťovat svůj obsah. Následující ukázka demonstruje zjednodušený zdrojový kód vzorové stránky (*Master Page*):

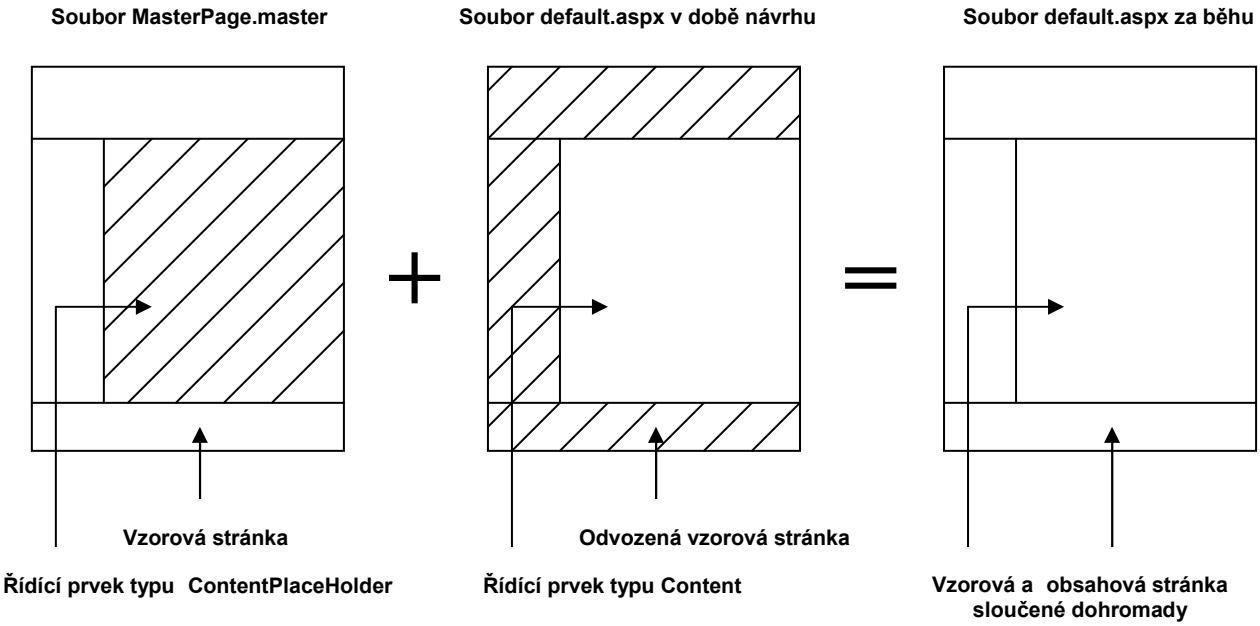
```
<%@ Master Language="C#" AutoEventWireup="true"
CodeFile="MasterPage.master.cs" Inherits="MasterPage" %>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Redakční systém ASP.NET x PHP</title>
</head>

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

```
HTML kód hlavičky
<asp:ContentPlaceHolder ID="ContentPlaceHolder1"
runat="server">
</asp:ContentPlaceHolder>

HTML kód patičky
    </form>
</body>
</html>
```

Pokud bychom chtěli do vzorové stránky vložit obsah souboru *default.aspx* vypadal by její zdrojový kód přibližně takto:

```
<%@ Page Language="C#" MasterPageFile="~/MasterPage.master"
AutoEventWireup="true"
    CodeFile="default.aspx.cs" Inherits="_default"
Title="Redakční systém ASP.NET x PHP" %>

<asp:Content ID="Content2"
ContentPlaceHolderID="ContentPlaceHolder1" runat="Server">
Zde bude obsah stránky
</asp:Content>
```

Pravidlem je, že direktiva *@Page* nastavuje atribut *MasterPageFile* na virtuální cestu ke vzorové stránce, která se má použít. Vlastní obsah je umístěn do řídícího prvku *Content*, jehož atribut *ContentPlaceHolderID* musí být nastaven na ID jednoho z řídících prvků *ContentPlaceHolder* ve vzorové stránce [3]. Vše nám shrnuje obrázek 5.



Obr 5: Grafické reprezentace vzorové stránky

3.4.2 Návrh databázových tabulek

Nežli se pustíme do programování a pomocí dotazů na databázi tahat potřebná data, je nutné celou databázi navrhnout, vytvořit takové tabulky a jejich sloupce, aby korespondovaly s problémy nastíněnými výše. Nejprve si musíme uvědomit jaké tabulky vlastně budeme potřebovat. Chceme vkládat články, tabulka na jednotlivé články tedy bude nutná. Články budou tříděny do kategorií a i ty budou mít svoji vlastní tabulku. Uživatelům bychom rádi umožnili vkládat svoje názory k článkům pomocí komentářů, budeme tedy počítat s tabulkou komentářů. Posledním problémem je vyřešení přihlašování, administrace a stanovení rolí. K tomuto účelu bude sloužit tabulka *autori*. Nyní tedy uvedu seznam tabulek:

Tabulka *kategorie*

Články chceme třídit do kategorií, začneme tedy touto tabulkou. Její struktura bude velice jednoduchá. Bude obsahovat sloupce *ID* a *nazev*. Náš systém bude obsahovat pouze jednoúrovňové kategorie, ty však budou pro jednoduchý redakční web zcela dostačující. Příliš mnoho kategorií by také mohlo mít neblahý vliv na návštěvníky webu, kterým by delší dobu trvalo najít svoji oblast zájmu.

Tab. 8: Sloupce tabulky *kategorie*

Název sloupce	Typ	Null	Popis
ID	int	ne	Jedinečné ID kategorie
nazev	varchar	ne	Název kategorie

Tabulka *clanky*

Tato tabulka by měla především obsahovat název článku, abstrakt a také vlastní obsah článku. Dále je nutné přidat sloupec *datum* zastupující datum vydání. Přidal jsem také sloupce udávající počet přečtení a zdali je článek schválen. Posledními nutnými sloupci byly *id_kategorie* a *id_autor*, umožňující třídění do kategorií a podle autorů. V závorce za datovými typy je uveden typ při vývoji aplikace v ASP.NET.

Tab. 9: Sloupce tabulky *clanky*

Název sloupce	Typ	Null	Popis
ID	int	ne	Jedinečné ID článku
id_autor	int	ne	ID autora
id_kategorie	int	ne	ID kategorie
nadpis	varchar	ne	Nadpis článku
abstract	text	ne	Abstrakt článku
obsah	text	ne	Obsah článku
datum	varchar (datetime)	ne	Datum vydání
pocet_precteni	int	ne	Počet přečtení
schvaleno	char (bit)	ne	Stav schválení článku

Tabulka *komentare*

Jak již bylo řečeno, tabulka slouží k zaznamenávání komentářů. Zajímá nás jméno a mail daného návštěvníka, zaznamenána bude i jeho IP adresa. Důležitými součástmi tabulky jsou též údaje předmět a vlastní text komentáře.

Tab. 10: Sloupce tabulky *komentare*

Název sloupce	Typ	Null	Popis
ID	int	ne	Jedinečné ID komentáře
id_clanek	int	ne	ID článku
IP	varchar	ne	IP adresa návštěvníka
jmeno	varchar	ne	Jméno návštěvníka
mail	varchar	ne	Mail návštěvníka
predmet	varchar	ne	Předmět komentáře
text	text	ne	Obsah komentáře
datum	varchar (datetime)	ne	Datum vložení

Tabulka *autori*

Byla uvedena tabulka *autori*, jako řešení uživatelských účtů obsahující přihlašovací data, údaje o uživatelovy roli a některé informace o jeho osobě. Toto řešení jsem použil u technologie PHP.

Tab. 11: Sloupce tabulky *autori* (Řešení v PHP)

Název sloupce	Typ	Null	Popis
ID	int	ne	Jedinečné ID autora
jmeno	varchar	ne	Jméno autora
mail	varchar	ne	Mail autora
informace	varchar	ne	Informace o autorovi
login	varchar	ne	Login autora
heslo	varchar	ne	Heslo autora
uziv_prava	int	ne	Uživatelská práva

ASP.NET přichází se zcela jiným přístupem k přihlašování a správě uživatelů. ASP.NET 2.0 obsahuje kompletní infastrukturu pro přihlašování (Membership Provider), přiřazování uživatelů do rolí (Role Provider) a správu profilů (Profile Provider). K mým účelům jsem využil jednoduchého *Altairis SimpleSqlMembershipProvider* a vytvořil tabulky uživatelského profilu a jeho rolí (obr.5).

Users

Column Name	Data Type	Allow Nulls
UserId	int	☐
UserName	varchar(100)	☐
PasswordHash	char(86)	☐
PasswordSalt	char(5)	☐
Email	varchar(100)	☐
Comment	text	☑
Enabled	bit	☐
DateCreated	datetime	☐
DateLastLogin	datetime	☑
DateLastActivity	datetime	☑
DateLastPasswordCha...	datetime	☐

Profiles

Column Name	Data Type	Allow Nulls
UserName	varchar(100)	☐
LastUpdate	datetime	☐
Jmeno	varchar(30)	☐
Informace	varchar(70)	☐

UsersInRoles

Column Name	Data Type	Allow Nulls
HashId	int	☐
UserName	varchar(100)	☐
RoleName	varchar(100)	☐

Roles

Column Name	Data Type	Allow Nulls
RoleName	varchar(100)	☐

Obr.5: Struktura tabulek pro autentizaci a autorizaci v ASP.NET

Pokud se naučíme s Providerem pracovat je přihlašování velice jednoduché a zároveň propracované. Přichází s mnoha automatickými prvky, kterých by se dalo využít u větších projektů. Další skvělou novou vlastností verze ASP.NET 2.0 je podpora klientů bez cookies. Pokud klient cookies nepodporuje je ID sezení předáváno v rámci URL adresy. Ještě bych zmínil bezpečnost aplikace v ASP.NET. Později v návrhu tabulky *autori* uvidíme, že heslo je uchováváno jako holý text v databázovém poli. Varianta je sice vhodná pro bezproblémové odeslání hesla zapomětlivému uživateli a není nutno generovat heslo nové. Je to ovšem i metoda nejméně bezpečná bez jakéhokoli šifrování. V mé aplikaci je použita metoda salted hash. Spočívá v tom, že k heslu připojíte náhodnou posloupnost znaků, tzv. salt. Vzniklý řetězec pak teprve hešujete.

3.5 Řešení

Následující část se bude zabývat již implementací aplikace redakčního systému na obou technologiích. Nejprve bych rád nastínil celkovou strukturu aplikační realizace, poté na vybraných příkladech jež obsluhují některou z událostí srovnal přístup technologií k danému problému. Především tedy přístup a získávání dat z databází, způsob výpisu získaných informací a ošetření některých dalších zajímavých funkcí systému.

3.5.1 Struktura aplikační realizace

Po vytvoření databáze se můžeme pustit již k samotné implementaci jednotlivých funkcí systému. Jak jsem již uvedl výše, v PHP je struktura stránky rozdělena v podstatě do 3 částí, tedy hlavičky, vlastního těla a patičky. Dá se tomu rozumět i tak, že každou z těchto částí reprezentuje tedy patřičný soubor. Po kliknutí např. na určitou z kategorií, což představuje vyfiltrovat z databáze články patřící do dané kategorie, budou parsovány 3 soubory. Danou událost sice obsluhuje pouze jediný soubor *kategorie.php*, ovšem ten includuje jak soubor *hlava.php* tak *pata.php* a navíc obsahuje vlastní kód, který řeší danou událost. Veškeré funkce systému budou většinou obsluhovány k tomu přiřazenými soubory a jejich zdrojovými kódy. Systém je navíc rozdělen na část uživatelskou, nacházející se v kořenové složce webu a část administrační se svoji zvláštní složkou. Dá se říct, že administrační část funguje po přihlášení zcela odděleně od části uživatelské.

V ASP.NET je systém rozdělen také na část uživatelskou a část administrační. Fungují zde již zmíněné *Master Pages* a jednotlivé události obsluhují stránky s příponami *aspx*. Někdy ošetřují jednu, ale i více událostí. Je zde však kompletní oddělení zdrojového kódu HTML od kódu programového. Tedy oddělení ASP.NET kódu dá se říct od designu. ASP.NET používá model s tzv. *kódem v pozadí (code-behind)*, kdy je každá stránka rozdělena na 2 soubory. Například stránka *default.aspx* je rozdělena na *default.aspx* s HTML kódem a ovládacími prvky ASP.NET a na *default.aspx.cs* (při použití jazyka C#, popřípadě .vb jestliže využíváme VisualBasic) kde se nachází backendový kód. V souboru *.aspx* se na soubor *.cs* (obsahující kód v pozadí) odkazuji pomocí direktivy *@Page* a jejího atributu *CodeFile*.

Některé speciální adresáře webového projektu v ASP.NET:

- *App_Code* – třídy sdílené pro celou aplikaci
- *App_Data* – adresář používaný pro data, může se jednat o databázi MSSQL (soubory s příponou .mdf), databázi Access, xml soubory apod., složka je zabezpečena a nedostupná zvenčí
- *Bin* – zkompilované assembly (knihovny .dll - např. AJAX, Membership Provider)
- *Controls* - vlastní uživatelské ovládací prvky, vhodné pro opakované kusy kódu, možnost využití pro cacheování, koncovka *ascx*

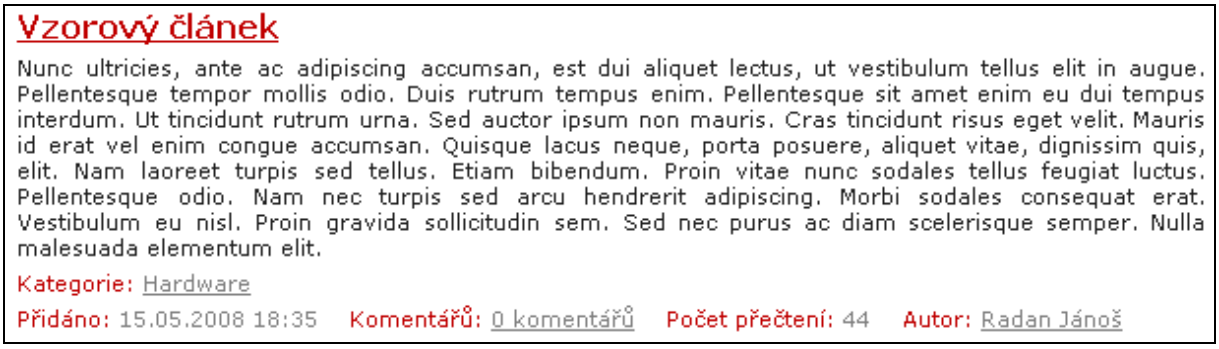
Nyní bych rád shrnul veškeré funkce realizovaného redakčního systému. Některým údalostem se budu v následující kapitole věnovat podrobněji a ukáži řešení pomocí obou technologií.

Funkce redakčního systému:

- Výpis článků s integrovaným stránkováním
- Filtrace dle kategorií a autorů
- Možnost vkládání komentářů k článkům
- Jednoduché vyhledávání v článcích
- Možnost registrace jako redaktor
- Po přihlášení možnost editace profilu, vkládat nové články, editovat, mazat
- Administrátor navíc práva schvalovat články, spravovat uživatele

3.5.2 Obsluha vybraných událostí

V aplikaci redakčního systému bylo jedním z úkolů zajistit výpis seznamu článků. Takový seznam by měl mít určitou strukturu. Dejme tomu, že budeme chtít zobrazovat nadpis článku, abstrakt a další doplňující informace. Takovými informacemi může být například kategorie, do které je článek zařazen, autor článku, datum vydání, počet komentářů apod. Úplnou použitou strukturu znázorňuje obr.6.



Obr.6: Struktura seznamu článků

Abychom mohli vytvořit takovýto seznam vložených článků bude nutné zajistit přístup aplikace k databázi. Na danou databázi učinit jeden či více dotazů, které nám umožní výpis požadovaných údajů. Nejprve je třeba definovat připojení a poté již na databázi směřovat dotazy.

Definice připojení k databázi:

V PHP definuji připojení odděleně v souboru *dbpripoj.php*, tento soubor obsahuje následující kód:

```
<?php
$pripojeni = mysql_connect("localhost", "root", "");
mysql_select_db("projekt",$pripojeni);
mysql_query("SET CHARACTER SET cp1250");
?>
```

Připojuji se k lokálně uložené databázi s uživatelským jménem *root* bez hesla. Konkrétně k databázi se jménem *projekt* a nastavuji kódovaní na *windows-cp1250*.

U ASP.NET se připojení nastavuje a definuje v souboru *web.config* v kořenovém adresáři aplikace:

```
<connectionStrings>
  <add name="ConnectionString" connectionString="Data
Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\projekt.mdf;
Integrated Security=True;User Instance=True"
            providerName="System.Data.SqlClient" />
</connectionStrings>
```

Web Developer nám připojení vytvoří i automaticky, použijeme-li někde v aplikaci přetažením z palety objekt *SqlDataSource*. Jestliže máme nastaveno připojení můžeme se pustit do programového kódu a pomocí dotazů získávat data.

Získání a výpis dat z databáze:

Začnu řešením vybrané události v PHP. O výpis seznamu článku se nám starají tyto soubory:

- *index.php* – zajišťuje výpis článků všech kategorií a autorů
- *kategorie.php* – filtrace seznamu článků dle zvolené kategorie
- *autor.php* – filtrace seznamu článků dle autora

Nyní k důležitým částem zdrojového kódu souboru *index.php*:

```
$vysledek = mysql_query("select
c.id,c.nadpis,c.abstract,c.datum,c.pocet_precteni,k.id,k.nazev,a.id,a.jmeno
from clanky c, kategorie k, autori a where c.id_kategorie = k.id and
c.id_autor = a.id and c.schvaleno = 'ano'",$pripojeni);
$pocetzaznamu = mysql_num_rows($vysledek);
```

Výše zobrazený úsek PHP kódu provede dotaz na databázi a načte z ní požadované informace. Konkrétně tedy z tabulek *kategorie, autori, komentare* za stanovených podmínek. Jednou z podmínek je také schválení článku, bude tedy načítat pouze schválené články. Nakonec počet článků uchová v proměnné *$pocetzaznamu*. Nyní přistoupíme k mnoha datovým informacím uchovávaným v proměnné *$vysledek*.

```
while ($row = mysql_fetch_row($vysledek)) {
    $idclanek = $row[0];
    $nadpis = $row[1];
    $abstract = $row[2];
    $datum = $row[3];
    $pocet_precteni = $row[4];
    $k_id = $row[5];
    $nazev = $row[6];
    $a_id = $row[7];
    $jmeno = $row[8];

$vysledek2 = mysql_query("select id from komentare where id_clanek =
$idclanek",$pripojeni);
$komclanek = mysql_num_rows($vysledek2);

    if ($komclanek >= "5" || $komclanek == "0")
    {
      $pom = "komentářů";
    } elseif ($komclanek == "1") {
      $pom = "komentář";
    } else {
```

```
        $pom = "komentáře";
    }

    $pom2 = "$komclanek $pom";


echo "<a href='clanek.php?id=$idclanek' class='ClankyNadpis'>$nadpis</a>";
echo "<p class='abstract'>$abstract</p>";
echo "<p class='kategorie'>Kategorie: <a
href='kategorie.php?id=$k_id'>$nazev</a></p>";
echo "<p class='pocet'>
Přidáno: <font color='#808080'>$datum</font>   
Komentářů: <a
href='clanek.php?id=$idclanek#komentare'>$pom2</a>   
Počet přečtení: <font
color='#808080'>$pocet_precteni</font>   
Autor: <a href='autor.php?id=$a_id'>$jmeno</a>   
          </p>";
}
```

Pomocí cyklu *while* načtu vybrané záznamy tabulek do číselných polí a uchovám v nově deklarovaných proměnných. Provedu dotaz na databázi komentářů a načtu další potřebné údaje. Ve finále vypíši článek pomocí PHP příkazu *echo*. Cyklus *while* končí až po vypsání všech článků, které odpovídají dotazu. Velmi podobný postup je použit také v souborech *kategorie.php* a *autor.php*, moduly jsou však rozšířeny a pozměněny.

Důležité části zdrojové kódu souboru *kategorie.php*:

Tento modul filtruje články po kliknutí na příslušnou kategorii. Řešení spočívá v předání parametru jedinečného *ID* dané kategorie. Tento parametr je stránce předán z předchozí v rámci adresy URL (např. *http://adresa/stranka.php?id=1*).

```
if (isset($_GET["id"])) {
      $k_id = $_GET["id"];

      $vysledek1 = mysql_query("select
      c.id,c.nadpis,c.abstract,c.datum,c.pocet_precteni,k.id,k.nazev,a.id,a
      .jmeno from clanky c, kategorie k, autori a where c.id_kategorie =
      '$k_id' and c.id_kategorie = k.id and c.id_autor = a.id and
      c.schvaleno = 'ano'",$pripojeni);
      $pocetzaznamu = mysql_num_rows($vysledek1);
}
```

Jestliže adresa URL bude obsahovat potřebný *ID* parametr kategorie, uložíme si jeho hodnotu do proměnné *$k_id*. Tato hodnota je poté použita ke stanovení podmínky v SQL dotazu (vyznačeno červeně), který nám zajistí výpis článku pouze dané kategorie. Stránka *autor.php* funguje na stejném principu jako modul pro výpis kategorií, v URL adrese je jí ovšem předán parametr jedinečné *ID* autora.

Nyní se pustím do řešení výše popsaných událostí pomocí technologie ASP.NET. Všechny operace řeší jediný soubor *default.aspx* a samozřejmě jeho „code-behind“ soubor *default.aspx.cs*. K filtru dle kategorií a autorů také používá parametry *ID* předávané v rámci adresy URL.

```
<asp:Repeater ID="repClanky" runat="server">
        <ItemTemplate>
            <asp:HyperLink NavigateUrl='<%# "~/clanek.aspx?id=" +
DataBinder.Eval(Container.DataItem, "ID")%>'
                runat="server" CssClass="ClankyNadpis"><%#
DataBinder.Eval(Container.DataItem, "Nadpis")%></asp:HyperLink>
            <p class='abstract'>
                <%# DataBinder.Eval(Container.DataItem, "abstract")%></p>
            <p class='kategorie'>
                Kategorie: <a href='<%# "default.aspx?kat=" +
DataBinder.Eval(Container.DataItem, "id_kategorie")%>'><%#
DataBinder.Eval(Container.DataItem, "KategorieNazev")%></a></p>
            <p class='pocet'>
                Přidáno: <font color='#808080'><%#
DataBinder.Eval(Container.DataItem, "datum", "{0:dd.MM.yyyy
HH:mm}")%></font>   
                Komentářů: <a href='<%# "clanek.aspx?id=" +
DataBinder.Eval(Container.DataItem, "id") + "#komentare" %>'><%#
DataBinder.Eval(Container.DataItem, "pocetKomentaru")%>
                    komentářů</a>    Počet
přečtení: <font color='#808080'><%#
DataBinder.Eval(Container.DataItem,
"pocet_precteni")%></font>   
                Autor: <a href='<%# "default.aspx?autor=" +
DataBinder.Eval(Container.DataItem, "id_autor")%>'><%#
DataBinder.Eval(Container.DataItem, "Jmeno")%></a>   
            </p>
        </ItemTemplate>
    </asp:Repeater>
```

Výše uvedený kód je obsahem souboru *default.aspx*. Jedná se o prvek *Repeater*, obsahující šablonu *ItemTemplate*, pro opakující se prvky (zvýrazněno červeně). *Repeater* je instance třídy *System.Web.UI.WebControls.Repeater*. *Repeater* zobrazuje data opakováním specifické šablony na každý záznam (řádek), případně položku v seznamu. Dá se říct, že tato část odpovídá úseku výpisu potřebných údajů pomocí příkazu *echo* v PHP. Ten však byl součástí cyklu, zde cyklus není potřeba. V kódu se také neustále objevuje tento prvek:

```
<%# DataBinder.Eval(Container.DataItem, "nazevSloupce") %>
```

Prvek patří ke třídě *System.Web.UI.DataBinder*, tato třída využívá reflexe ke zjednodušení syntaxe navázání dat. Stěžejní metodou je metoda *Eval*.

Nyní k důležitým součástem stránky *default.aspx.cs* (kód v pozadí):

```
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!Page.IsPostBack)
        {
            LoadData(Helpers.ReadIntFromParam("page") - 1,
                     Helpers.ReadIntFromParam("kat"),
                     Helpers.ReadIntFromParam("autor"),
                     Helpers.ReadStringFromParam("hltext"));
            SetHeaderLabel(Helpers.ReadIntFromParam("kat"),
                           Helpers.ReadIntFromParam("autor"));
        }
    }
```

Nejprve voláme funkci *LoadData*, které jsou předávány vstupní argumenty načtené z parametrů nacházejících se v adrese URL, tak jak jsem zmínil již výše. Využívá se zde třídy *Helpers* obsahující funkci *ReadIntFromParam*.

```
        public static int ReadIntFromParam(string paramName)
        {
            int ret = 0;
            if (HttpContext.Current.Request[paramName] != null)
            {
                try
                {
                    ret =
Convert.ToInt32(HttpContext.Current.Request[paramName]);
                }
                catch
                {
                    ret = 0;
                }
            }
            return ret;
        }
```

Zde jen uvedu, že funkce provede převod řetězcového parametru do datového typu *integer*. Pozor funkce není součástí souboru *default.aspx.cs*, ale třídy *Helpers* se souborem *Helpers.cs* v adresáři *App_Code*.

Funkce *LoadData*, jejíž kód uvedu dále, obsluhuje veškeré události spojené s jejími vstupními parametry. Třídí články dle kategorií, autorů, vypisuje seznam článků s hledaným výrazem a umožňuje stránkování.

```
private void LoadData(int page, int kategorie, int autor, string
hledanyText)
    {
        page = page < 0 ? 0 : page;

        string where = String.Empty;
        if (kategorie > 0)
        {
            where = " AND clanky.id_kategorie = @id_kategorie";
        }

        if (autor > 0)
        {
            where += " AND Users.UserId = @id_autor";
        }

        if (hledanyText.Length > 0)
        {
            where += " AND (clanky.nadpis Like @hledanyText OR
clanky.abstract Like @hledanyText OR clanky.obsah Like @hledanyText)";
        }

        string sqlSelect = "SELECT  clanky.id, clanky.id_kategorie,
kategorie.nazev as KategorieNazev, clanky.nadpis, clanky.abstract, " +
                           " clanky.datum, clanky.pocet_precteni,
komentare.PocetKomentaru, Profiles.Jmeno, Users.UserId as id_autor" +
                           " FROM clanky LEFT OUTER JOIN kategorie ON
clanky.id_kategorie = kategorie.id " +
```

```
                            " LEFT OUTER JOIN (SELECT  clanky.id,
COUNT(komentare.ID) as PocetKomentaru " +
                            " FROM clanky LEFT OUTER JOIN komentare ON
clanky.id = komentare.id_clanek " +
                            " GROUP BY clanky.id) komentare ON clanky.id =
komentare.id " +
                            " LEFT OUTER JOIN Users ON Users.UserId =
clanky.id_autor " +
                            " LEFT OUTER JOIN Profiles ON Users.UserName =
Profiles.UserName " +
                            " WHERE (clanky.schvaleno = 1) " + where + "
ORDER BY clanky.datum DESC";

        DataTable dt = new DataTable();
        using (SqlConnection con = new
SqlConnection(ConfigurationManager.ConnectionStrings["ConnectionString"].Co
nnectionString))
        {
            con.Open();
            using (SqlCommand cmd = new SqlCommand(sqlSelect, con))
            {
                if (kategorie > 0)
                {
                    cmd.Parameters.AddWithValue("id_kategorie", kategorie);
                }
                if (autor > 0)
                {
                    cmd.Parameters.AddWithValue("id_autor", autor);
                }
                if (hledanyText.Length > 0)
                {
                 cmd.Parameters.AddWithValue("hledanyText",
                 "%" + hledanyText + "%");
                }

                using (SqlDataReader dr = cmd.ExecuteReader())
                {
                    dt.Load(dr);
                    dr.Close();
                }
            }
            con.Close();
        }

        PagedDataSource pgitems = new PagedDataSource();
        DataView dv = new DataView(dt);
        pgitems.DataSource = dv;
        pgitems.AllowPaging = true;
        pgitems.PageSize = 7;
        pgitems.CurrentPageIndex = page;
        if (pgitems.PageCount > 1)
        {
            rptPages.Visible = true;
            ArrayList pages = new ArrayList();
            for (int i = 0; i < pgitems.PageCount; i++)
            {
                pages.Add((i + 1).ToString());
            }
            rptPages.DataSource = pages;
            rptPages.DataBind();
        }
```

```
        else
        {
            rptPages.Visible = false;
        }

        if (pgitems.Count > 0)
        {
            repClanky.DataSource = pgitems;
            repClanky.DataBind();

            pZadnyClanek.Visible = false;
        }
        else
        {
            pZadnyClanek.Visible = true;
        }
    }
```

Jestliže adresa URL obsahuje udávané parametry jsou funkci *LoadData* předány jako vstupní argumenty. Funkce nejprve definuje SELECT dotaz na databázi SQL. Celý dotaz je uchován v řetězcové proměnné *string sqlSelect* (v kódu zvýrazněno růžovou barvou). Navíc je celý dotaz dynamicky doplňován o jednu z podmínek dle zadaného vstupního argumentu. Toto doplnění je zajištěno další řetězcovou proměnnou *string where* (zvýrazněno červenou barvou). Připojení k databázi a načtení informací je vázáno na vytvořený *ConnectionString* v souboru *web.config* a v kódu je zvýrazněno oranžovou barvou. Příkaz *SqlCommand(sqlSelect, con)* zajistí připojení k vybrané databázi s daným dotazem. Ještě bych zmínil zdroj a navázáni dat na prvek *Repeater* v souboru *default.aspx* jehož atribut ID je nazván *repClanky*. Tato operace je v kódu zvýrazněna modrou barvou a řeší tedy tyto 2 kroky používané k vázání dat na datové ovládácí prvky ASP.NET:

1. Nastaví se vlastnost *DataSource* ovládacího prvku na instanci třídy odvozené od rozhraní *System.Collections.IEnumerable*.
2. Zavolá se metoda *DataBind*, která spustí vázání dat.

Jak jsem již uvedl, není nutné řešit cyklus pro procházení jednotlivými záznamy, o vše se postará metoda *DataBind*.

Celý redakční systém obsahuje několik dalších modulů řešících jeho funkce. Nebudu zde však probírat všechny jednotlivé moduly. Základní a nejpodstatnější rozdíly při vývoji aplikace jsem se již snažil nastínit. Především byla představena struktura aplikační realizace u obou technologií, na vybraném modulu sestavení připojení k databázi, výpis dat a ošetření některých událostí. K práci je přiložen kompletní zdrojový kód funkční aplikace na základě obou technologií. Jestliže bude čtenáře zajímat programové řešení jednotlivých problému, jak se budou měnit dotazy na databáze při vkládání či úpravě dat, kterak je zajištěna validace dat vstupních formulářů a ostatní události, má možnost do těchto zdrojových kódů nahlédnout.

4 Závěr

V mé práci jsem se snažil prozkoumat vývoj webových aplikací pomocí technologií PHP a ASP.NET. Pokusil jsem se o pochopení základních principů obou technologií a o jejich srovnání. Text poskytuje mnoho oblastí v nichž jsou dané technologie srovnávány, počínaje instalací technologií, způsobem zápisu kódu a syntaxí jazyků. Na názorných ukázkách jsou vystiženy hlavní rozdíly při realizaci aplikací na těchto technologiích, především tedy odlišného zápisu kódu, typovosti jazyka, odlišnostmi generování HTML kódu po proběhnutí aplikace a případnými dalšími rozdíly. Mezi jednu z nejdůležitějších vlastností technologií patří také možnost přístupu k datovým zdrojům. Práce obsahuje příklady přístupu obou technologií a na nich se snaží uvést, jak každá z nich k datovým zdrojům přistupuje, zda využívá funkcí či objektů, speciálních knihoven apod.

Součástí práce byla také implementace funkční aplikace středního rozsahu na obou technologiích. I zde se snažím nastínit rozdíly technologií při již reálném vývoji. Někdy tyto rozdíly korespondují již s částí obecnou, ve které jsou také uvedeny. Přímá aplikační realizace však tyto rozdíly prohlubuje a čtenář již má určitý náhled do praxe. Pro mne samotného bylo naprogramování aplikace redakčního systému jednodušší v PHP. Především tedy proto, že jsem již některé aplikace v PHP programoval. Naopak, co se týče syntaxe jazyka C# a vůbec objektového programování, nesahaly moje znalosti příliš hluboko. Naprogramování aplikace v PHP je tedy pro mne zatím stále rychlejší. Myslím si však, že jakmile dokážu plně využít veškerých možností .NETU a jeho objektů řešených na míru jednotlivým událostem, bude již situace jiná. K PHP je zatím na internetu k dispozici mnohem více hotových řešení, celková podpora a komunita je rozsáhlejší. PHP je stále více oblíbené mezi programátory internetových aplikací. To mohu potvrdit i díky malému průzkumu mezi známými, kteří se programováním webových aplikací zabývají. Samozřejmě mne na druhou stranu nadchlo vývojové prostředí Web Developeru pro technologii ASP.NET, které je postaveno na základě Microsoft Visual Studia. Jak jsem již uvedl, v rámci tohoto prostředí máme tak říkajíc vše pod jednou střechou. Není třeba se stále někam přepínat, což ruší při vývoji aplikace. Ještě bych také jednou vyzdvihnul strukturu aplikační realizace v ASP.NET a jeho oddělení designu od kódu. Aplikace jsou pak velice přehledné, stávají se téměř plnohodnotným objektovým programem. PHP a jeho *spaghetti-code* model, tedy míchání HTML kódu se serverovým skriptem, dělá zdrojové kódy aplikací někdy nepřehledné a špatně udržovatelné. Jsem však především rád, že jsem se díky technologii ASP.NET naučil modernímu objektovému jazyku a svoje znalosti s touto technologií bych chtěl prohlubovat i dále. Nehledě na to, že dobrých ASP.NET programátorů je málo a jejich hodnota bude stále stoupat.

Diskuze na téma PHP versus ASP.NET je tedy dle mého názoru téma věčné. Příznivců každého z táborů je mnoho a tak snad jen čas nám bude moci ukázat, kdo z nich měl úplnou pravdu. Zjednodušeně se dá ovšem říci, že ASP.NET je určeno především pro Windows, je zcela objektově zaměřeno, je tedy moderním objektovým jazykem, který je možno využít k realizaci velkých webových projektů, je možné využívat více jazyků, architektura .NET Frameworku je technologie směřující do budoucna. PHP však nezůstává pozadu, rychle se vyvíjí, již se také stále více od uvedení verze 5 zaměřuje na OOP, stále však není tak vhodné a připravené na realizaci větších projektů. V menších projektech je velice pružné, rychlé, někdy u uživatelů vítězí v jednoduchosti zápisu a má širokou podporu.

A. Seznam použitých zkratek

HTTP – Hyper Text Transfer Protocol

HTML – Hyper Text Markup Language

XHTML - eXtensible Hypertext Markup Language

URL - Uniform Resource Locator

CGI – Common Gateway Interface

WYSIWYG – What You See Is What You Get

CSS – Cascading Style Sheets

WWW – World Wide Web

CMS – Content Management Systems

JSP – Java Server Pages

MySQL – Structured Query Language fy MySQL AB.

MSSQL – Structured Query Language fy Microsoft

PHP – Hypertext Preprocessor

ASP – Active Server Pages

ASP.NET – nástupce technologie ASP (Active Server Pages)

CLR – Common Language Runtime

B. Seznam literatury a použitých zdrojů

Publikace:

[1] Hlavenka, J; Sedlář, R; Holčík, T. *Vytváříme WWW stránky*, Computer Press, 2002, 372 s., ISBN 80-7226-748-5

[2] Písek, S. *ASP.NET začínáme programovat*, Grada, 2003, 228 s. ISBN 80-247-0526-5

[3] Bellinaso, M. *ASP.NET 2.0 Website Programming: Problem – Design – Solution,* Wrox, 2006, 600 s., ISBN 978-0764584640

[4] Lerdorf, R., Tatroe, K., MacIntyre, A. *Programming PHP*, 2nd edition, O'Reilly Media, Inc., 2006, 540 s., ISBN 978-0596006815

[5] Dlouhý, R. PHP v příkladech, ComputerMedia, 2007, 180 s., ISBN 80-86686-83-3

[6] PAYNE, Ch. *Naučte se ASP.NET za 21 dní,* Computer Press, 2002, 786 s., ISBN 80-226-05-5

[7] Kosek, J. *PHP – tvorba interaktivních internetových aplikací*. Grada, 1999. 492 s. ISBN 80-7169-373-1

Webové zdroje:

http://cz.php.net/history/

http://www.php.net

http://www.asp.net

http://www.interval.cz

http://www.programujte.com

http://www.aspnet.cz

C. Obsah přiloženého CD

- \asp_net - obsahuje soubor *redsys_asp.zip*
- \php - obsahuje soubor *redsys_php.zip* a *database.zip*
- \text – obsahuje soubory *bakalar_prace.pdf* a *metadata.pdf*
- *readme.txt* – doplňující pokyny k obsahu CD