Content management system pro Getmore s.r.o.

Content management system for Getmore s.r.o.

Vít Kymla

Bakalářská práce
2007

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2006/2007

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Vít KYMLA**
Studijní program:	**B 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Content managment systém pro Getmore s.r.o.**

Zásady pro vypracování:

1. Vytvořte literární rešerši na zadané téma.
2. Seznamte se s existujícími řešeními content managment systémů (CMS). Charakterizujte je a uveďte jejich výhody a nevýhody.
3. Proveďte návrh vlastního řešení pro správu CMS dle požadavků firmy Getmore, s.r.o.
4. Na základě tohoto návrhu realizujte software, jehož funkčnost ověřte praktickým nasazením. Využijte přitom technologii ASP .NET 2.0.
5. Základní požadavky na aplikaci jsou následující: zobrazení stránek pro běžné návštěvníky, možnost "uzamčení" některých stránek s přístupem pouze některým uživatelům, možnost on-line úpravy obsahu stránek a dále možnost rychlé změny obsahu dynamických částí (novinky, odkazy).

Rozsah práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. Nagel, Ch.: C++ 2005 – programujeme profesionálně. Computer Press, Praha 2007. 2. Evjen, B., Hanselman, S. aj.: ASP.NET 2.0 – programujeme profesionálně. Computer Press, Praha 2007. 3. Sharp, J., Jagger, J.: Visual C++ .NET – krok za krokem. Mobil Media, Brno 2002. 4. http://www.msdn.com – knihovna MSDN

Vedoucí bakalářské práce:	**Ing. Pavel Pokorný, Ph.D.** Ústav aplikované informatiky
Datum zadání bakalářské práce:	**13. února 2007**
Termín odevzdání bakalářské práce:	**24. května 2007**

Ve Zlíně dne 13. února 2007

prof. Ing. Vladimír Vašek, CSc.
děkan

L.S.

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

ABSTRAKT

Práce se v teoretické části zabývá představením technologie ASP.NET 2.0. Přináší zevrubné seznámení s historií, vývojem, se základními principy této technologie a s výhodami jejího využití pro vývoj webových aplikací. Dále se teoretická část zabývá aplikacemi typu CMS. Popisuje možné implementace CMS architektur a shrnuje základní požadavky na CMS. Teoretická část přináší také podrobnější pohled na tři existující řešení CMS. Praktická část se zabývá návrhem CMS pro společnost Getmore na platformě ASP.NET 2.0. Podrobně je rozebrán návrh databázové struktury, rozvržení projektů a způsob integrace do webové aplikace. Praktická část obsahuje také popis implementace základních modulů CMS, které jsou součástí programu vytvořeného v rámci této bakalářské práce.

Klíčová slova: Redakční systém, správa obsahu, ASP.NET 2.0, webové aplikace, databázové aplikace, webové formuláře, serverové ovládací prvky, šablonové prvky, šablony

ABSTRACT

In the theoretic part of this dissertation is introduced the ASP.NET 2.0 technology. It brings basic familiarization with history, development, basic fundamentals of the technology and advantages of using the technology for web application development. The next section of the theoretic part speaks about CMS applications. It describes possible implementations of CMS architectures and it also recapitulates basic requests to CMS systems from customers. The theoretic part also brings more detailed view to three existing CMS solutions. The practical part is concerned with concept of the CMS for Getmore Corporation based on the ASP.NET 2.0 technology. Closely is described concept of database structure, projects layout and way of integration the CMS to another web application. The practical part also contents descriptions of the base CMS modules implementations that are part of program that was created within the frame of this dissertation.

Keywords: Content management system, content administration, ASP.NET 2.0, web applications, database applications, web forms, server web controls, template controls, templates

Poděkování

Předně děkuji vedoucímu této bakalářské práce – Ing. Pavlu Pokornému Ph.D.. Děkuji také vedení společnosti Getmore s.r.o. za umožnění realizace bakalářské práce v rámci pracovního úvazku. Dále děkuji vedoucímu vývoje společnosti Getmore Bc. Janu Vilímkovi za náměty a připomínky.

Motto:

„Změna těší.“

Marcus Tullius Cicero

*„Potřebujeme podstatně **nový** způsob myšlení, jestliže má lidstvo přežít.“*

Albert Einstein

Prohlašuji, že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků, je-li to uvolněno na základě licenční smlouvy, budu uveden jako spoluautor.

Ve Zlíně …….

Podpis diplomanta

OBSAH

ÚVOD 8
I. TEORETICKÁ ČÁST 9
1.1 TECHNOLOGIE MICROSOFT ASP.NET 2.0 10
1.1.1 Historie a vývoj technologie ASP.NET 2.0 10
1.1.2 Budoucnost a směřování vývoje technologie ASP.NET 13
1.1.3 Výhody vývoje aplikací na platformě ASP.NET, cíle ASP.NET 13
1.1.3.1 Východy a novinky ASP.NET 13
1.1.3.2 Hlavní cíle ASP.NET 15
1.1.4 Přehled základních prostředků pro vývoj webových aplikací v ASP.NET 16
1.1.4.1 Objektový model programování 16
1.1.4.2 Princip webových formulářů (WebForms) 17
1.1.4.3 Serverové ovládací prvky (WebControls) 18
1.1.4.4 Framework na tvorbu webových formulářů, master pages 19
1.1.4.5 Práce s daty, ADO.NET 19
1.1.4.6 Správa členství a rolí (membership service a roles service) 20
1.1.4.7 Personalizace a přizpůsobení webového sídla, webparts 21
1.1.4.8 Caching dat na straně serveru 23
1.2 SYSTÉMY CMS 24
1.2.1 Možnosti architektury spojení CMS a webového sídla 24
1.2.1.1 Architektura se společným GUI 25
1.2.1.2 Architektura s odděleným GUI 27
1.2.1.3 Další architektury CMS 29
1.2.2 Požadavky na CMS, základní moduly CMS 29
1.2.2.1 Administrace dostupná on-line 29
1.2.2.2 WYSIWYG editace obsahu 30
1.2.2.3 SEO – optimalizace pro webové vyhledávače 30
1.2.2.4 Fulltextové vyhledávání 30
1.2.2.5 Antispamová ochrana 31
1.2.2.6 Lokalizace 31
1.2.2.7 Základní moduly pro CMS 31
1.2.3 Přehled existujících řešení na českém trhu 33
1.2.3.1 MultiCMS 33
1.2.3.2 Vizus 35
1.2.3.3 Drupal 37
1.2.3.4 Srovnání systémů 38
II. PRAKTICKÁ ČÁST 39
2.1 ZADÁNÍ OD SPOLEČNOSTI GETMORE 40
2.1.1 Požadované použité technologie 40
2.1.2 Požadovaný princip fungování s webovou prezentací 40
2.1.3 Požadovaná architektura aplikace CMS 41
2.1.4 Požadované moduly 41
2.1.4.1 Vrstva oprávnění 41
2.1.4.2 Články 42
2.1.4.3 Document management system 42

2.2 DATABÁZOVÁ STRUKTURA 43
2.2.1 Základní systémová struktura 43
2.2.2 ASP.NET struktura pro membership a roles management 44
2.2.3 Struktura pro oprávnění 45
2.2.3.1 Struktura tabulek pro oprávnění 45
2.2.3.2 Stored procedury pro oprávnění 46
2.2.4 Struktura pro články 47
2.2.4.1 Struktura tabulek 47
2.2.4.2 Stored procedury a triggery pro články 48
2.2.5 Struktura pro DMS – dokumentový systém 48
2.2.5.1 Stored procedury a triggery pro DMS 49
2.3 SOLUTION GETMORE.CMS 50
2.3.1 Projekt Getmore.CMS 51
2.3.2 Projekt Getmore.CMS.DataWrapper 51
2.3.3 Projekt Getmore.CMS.Core 52
2.4 ZPŮSOB NAPOJENÍ NA WWW PREZENTACI 53
2.4.1 Realizace rozhraní 53
2.5 POPIS IMPLEMENTACE MODULŮ 54
2.5.1 Základní třídy 54
2.5.1.1 Třída CMSPage 54
2.5.1.2 Třída CMSModule 55
2.5.2 Datová vrstva 55
2.5.2.1 Třída CMSDataLayer 57
2.5.2.2 Třída ArticlesDataLayer 58
2.5.2.3 Třída ArticlesDataSource 58
2.5.3 Vrstva zabezpečení a opravňování 58
2.5.3.1 Třída PermissionsProvider 59
2.5.3.2 Třída ModulePermissionsHolder 59
2.5.3.3 Třída PermissionsHashTableItem 60
2.5.4 Články 61
2.5.4.1 Třída ArticlesModule 61
2.5.4.2 Třída ArticleDetail 63
2.5.4.3 Třída ArticleEditableDetail 64
2.5.4.4 Poznámka k využití šablon 65
2.5.4.5 Implementace WYSIWYG editoru 66
2.5.5 DMS 66
2.5.5.1 Třída DMSModule 67
2.5.5.2 Třída DMSFoldersPanel 67
2.5.5.3 Třída DMSDocumentsPanel 68
ZÁVĚR 69
ZÁVĚR V ANGLIČTINĚ 70
SEZNAM POUŽITÉ LITERATURY 71
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK 73
SEZNAM OBRÁZKŮ 74
SEZNAM TABULEK 75

ÚVOD

Mezi jeden z trendů moderní doby patří migrace zdrojů informací do prostředí digitálních dat. Klasické knižní publikace se stávají pro některá rychle se měnící odvětví nepraktické. Knihovny jsou plné objemných svazků, které již nikdo nikdy neotevře. Příkladem je počítačová literatura. Ta stárne snad rychleji než cokoliv jiného. Často se stane, že již v době vydání odborného titulu je obsah knihy zastaralý, protože vývoj v oblasti IT předstihl samotného autora i vydavatelství. Stále vznikají požadavky na udržování aktuálních informací ve všech sférách. Některé informace se proto z oblasti tištěné literatury přesouvají do Internetových magazínů, fór a jiných. V tu chvíli je nutno zajistit prostředky pro chod těchto portálů. Významnou roli v této oblasti hrají systémy pro správu dat – tzv. redakční nebo publikační systémy. Žádný pravidelně aktualizovaný portál se bez tohoto systému neobejde a jeho chod je na něm zcela závislý. Redakční systémy však nejsou součástí jen velkých portálů, ale významně se uplatňují také v administraci webových stránek společností nebo institucí.

Existuje nepřeberné množství redakčních systémů. Tato práce si klade za cíl navrhnout a realizovat další systém tohoto typu. Otázka zní: Proč vytvářet další redakční systém a řešit problémy, které jsou již dávno vyřešeny? Předně je nutno vysvětlit, že tento systém bude v některých ohledech výjimečný. Bude realizován na platformě ASP.NET 2.0, která je pro existující redakční systémy spíše netypická. Výjimečný bude také způsob, jakým bude možno systém využívat ve webových prezentacích. Cílem je, aby programátor, který chce tento systém do své webové aplikace zařadit, nemusel napsat jediný řádek programového kódu. Motta uvedená na předchozí stránce předznamenávají, že se práce nechce „zabořit“ do klasických schémat redakčních systémů, ale že se snaží jít novou, doposud nevyšlapanou cestou. Výsledkem má být kompaktní stavebnice, kterou bude mít programátor k dispozici při vývoji vlastního administrovatelného portálu. Touto myšlenkou zapadá redakční systém zcela do celkového konceptu vývoje webových aplikací na platformě ASP.NET.

I. TEORETICKÁ ČÁST

1.1 Technologie Microsoft ASP.NET 2.0

Technologii ASP.NET můžeme v současné době považovat za nejmodernější nástroj pro vývoj aplikací určených do prostředí Internetu. Jde především o aplikace postavené na bázi „klient-server“, kde na klientské straně stojí prohlížeč internetových stránek a na serverové straně webový server. Pro přesnost je nutno uvést, že prostředí ASP.NET můžeme použít také pro vývoj webových služeb, kde klientskou stranou může být libovolná aplikace, která umí vysílat požadavky na webové služby a zpracovávat jejich odpovědi.

1.1.1 Historie a vývoj technologie ASP.NET 2.0

Před vznikem a rozšířením Internetu bylo standardní aplikační vybavení počítačů zastoupeno výhradně tzv. „tlustými aplikacemi“. Za „tlusté aplikace“ považujeme takové aplikace, které jsou pevně nainstalované v konkrétním počítači a v jeden okamžik dokážou sloužit jen jednomu uživateli, který obsluhuje počítač. Rozvoj lokálních sítí pak umožnil vznik tzv. „tlustých klientů“. Takové aplikace již umožňovaly pracovat se vzdálenými aplikacemi a databázemi a fungovaly na bázi klient-server. S jednou „velkou“ serverovou aplikací tak mohlo pracovat více klientských stanic, na kterých však musela být nainstalována klientská aplikace. Toto řešení přinášelo řadu výhod (bezpečnost, stabilita) a i v současné době mají aplikace postavené na tomto modelu nezastupitelné místo (bankovní sektor apod.).

Nástup Internetu v polovině 90. let však připravil prostředí pro vznik webových aplikací. Kromě umožnění rychlé komunikace, sdílení a výměny dat umožnil Internet také vytváření tzv. „tenkých klientů“. Za „tenkého klienta“ považujeme klientskou aplikaci, která funguje výhradně v prostředí webového prohlížeče. Jako „médium“ pro spojení se serverovou stranou (aplikací na webovém serveru) funguje protokol HTTP (resp. HTTPS). Klient vysílá HTTP požadavky a jako odpovědi dostává HTML stránky, které zobrazuje. V současné době umožňují technologie na straně klienta (JavaScript, JScript, ActiveX…) vytvořit velmi sofistikované uživatelské rozhraní, které se značně podobá uživatelskému rozhraní „tlustých klientů“. Velkou výhodou tohoto řešení je fakt, že není nutné na jednotlivé počítače instalovat klientskou aplikaci (resp. je nutno instalovat kompatibilní webový prohlížeč, který umožní běh klientské aplikace).

Pochopitelně bylo nutné také vytvořit technologie pro vytváření server-side aplikací, které by umožňovaly získávat data a generovat výstupy pro „tenké klienty“. Velmi populárními se staly interpretované skriptovací jazyky, které umožnily velmi rychlý a jednoduchý vývoj serverových aplikací. Klientská strana se může dotázat na konkrétní skript (soubor), předat mu parametry a čekat na odpověď ve formátu HTML. Společnost Microsoft vytvořila koncepci „Active server pages“ (ASP), která umožnila psát skripty v jazyce VBScript nebo JScript. Pomocí této technologie je možno vytvářet i velmi komplikované a obsáhlé webové aplikace, se kterými je možno se setkat na veřejném Internetu i v soukromých intranetových sítích. Poslední verze Active Server Pages 3.0 byla uvolněna na konci roku 1998.

Řešení pomocí serverových interpretovaných skriptů má však podstatné nedostatky. Především je aplikace doslova roztříštěna do velkého množství samostatných souborů, které nejsou nijak systémově svázány do jednoho produktu. Pro vytváření rozsáhlé webové aplikace je pak při vývoji nutno dodržovat přísná pravidla na úrovni postupů, aby se vývoj stal i nadále udržitelným a umožňoval další rozšiřování aplikace. Další podstatnou nevýhodou je interpretace skriptů. To znamená, že zdrojový kód aplikace není nikdy naráz kompilován. U nekompilovaného kódu nemůžeme nikdy zajistit syntaktickou bezchybnost a chybí také kontrola referencí na procedury a funkce. V konečném důsledku si nemůžeme být při nasazení aplikace nikdy jisti, že všechny části spolu dokážou spolupracovat, dokud celou aplikaci kompletně neotestujeme. Další nevýhodou, která souvisí s předchozím problémem, je skutečnost, že celá aplikace je na serveru vystavena ve formě zdrojových kódů. To způsobuje hned několik nedostatků. Primárně kód aplikace není nijak chráněn a může být odcizen třetí stranou, což může způsobit škodu na „know-how“ výrobce nebo může zapříčinit snížení zabezpečení celé aplikace, protože znalost zdrojového kódu může být vodítkem pro případné narušitele. Dalším problémem u zdrojových kódů skriptů je jejich verzování. Bez pomocných nástrojů není možno nijak jednoduše zajistit jednorázové stanovení verze všech skriptů aplikace, což může vést k výrazným problémům při nasazování nových verzí, zajištění zpětné kompatibility apod.

Odpovědí firmy Microsoft na vyřešení výše popsaných nedostatků byla technologie ASP.NET 1.0, která se stala nástupcem technologie Active Server Pages 3.0. Představena byla v létě roku 2000 na konferenci profesionálních vývojářů (Professional Develo-

pers Conference) pod názvem ASP+ společně s finální verzí .NET Framework 1.0. Později došlo k jejímu přejmenování na ASP.NET. Přípona „.NET“ signalizovala zapojení této technologie do celkového konceptu firmy Microsoft v té době. Koncept „.NET“ směřoval vývoj aplikací právě do oblasti Internetu, webových služeb a tenkých klientů[1]. [3, str. 34] Nejpodstatnější myšlenkou této nové technologie bylo umožnění vývoje aplikací podobným způsobem, jak tomu byli programátoři zvyklí u klasických tlustých aplikací. Navíc k vývoji mohli použít jednu platformu, jeden programovací jazyk a v konečném důsledku i jedno vývojové prostředí (IDE). Podstatnou změnou bylo převedení způsobu vytváření webových aplikací z procedurálního na objektově orientovaný způsob. Tato smělá myšlenka umožňovala oddělit vytváření funkčních objektů a uživatelského rozhraní od psaní HTML kódu. Toto však znamenalo ztrátu kontroly nad generovaným HTML kódem a mnoho programátorů nebylo ochotno na tento nový model přistoupit. Přesto se však technologie v praxi velmi osvědčila a dosáhla velkého rozšíření hlavně na profesionální úrovni. Podstatnou výhodou technologie ASP.NET resp. .NET Framework bylo obrovské množství hotových knihoven, které mohli programátoři přímo využívat. Odpadla fáze programování vlastních komponent a knihoven pro zajištění základních funkcí. Objektový způsob navíc umožnil jednoduchou úpravu existujících objektů pouhým zděděním a přidáním nových funkcionalit. Technologie ASP.NET 1.0 se dále vyvíjela a došlo k uvolnění několika aktualizací a service-packů.

Protože se nastolená cesta vývoje webových aplikací ukázala jako správná, uvolnila společnost Microsoft v listopadu roku 2005 novou verzi technologie ASP.NET. Nová verze 2.0 přímo souvisela s vydáním nové verze rámcového prostředí .NET Framework 2.0. Nejednalo se však o pouhé vylepšení stávající technologie, ale o další revoluční změny srovnatelné s přechodem od ASP na ASP.NET. Nová verze odstranila nedostatky a doplnila nástroje, které v původní verzi vývojářům chyběly. Došlo k zautomatizování některých postupů a pochopitelně k rozšíření základních knihoven .NET Framework o nové komponenty.

[1] Koncept .NET se pochopitelně netýkal pouze vývoje webových aplikací, ale i klasických tlustých aplikací a tlustých klientů. „.NET“ spíše signalizoval kompatibilitu s prostředím Internetu.

ASP.NET 2.0 je také poslední oficiálně vydanou verzí prostředí ASP.NET. Pokud se dále v textu budu zmiňovat o ASP.NET, pak se bude jednat o ASP.NET 2.0.

1.1.2 Budoucnost a směřování vývoje technologie ASP.NET

Vývojáři firmy Microsoft (vedoucí projektu je Scott Guthrie) pracují na nové verzi technologie ASP.NET. Projekt je veden pod označením „Atlas“ a je zaměřen na „klientský framework“, který umožní vytváření webových aplikací, které komunikují se serverovou stranou prostřednictvím technologie AJAX. To povede ke snížení počtu obnovení (reload) celé HTML stránky v prohlížeči. Dle dostupných údajů bude celý klientský framework postaven 100% na JavaScriptu a bude umožňovat vytváření velmi bohatého uživatelského rozhraní. Technologie bude spojovat výhody DHTML, JavaScriptu a XMLHTTP. [4]

V přípravě je také nová verze vývojového prostředí Microsoft Visual Studio, která nese kódové označení „Orcas“ a v současné době je k dispozici BETA verze 1. S nástupem operačního systému Windows Vista byl uvolněn .NET Framework 3.0. Tato technologie však nepřináší žádné revoluční změny. Virtuální stroj pro běh .NET aplikací (CLR) a základní knihovna tříd (BCL) zůstávají stejné jako ve verzi 2.0. Novinkou jsou knihovny pro vývoj WinFX (Avalon) [1] aplikací, které však nijak nesouvisí s technologií ASP.NET.

1.1.3 Výhody vývoje aplikací na platformě ASP.NET, cíle ASP.NET

1.1.3.1 Východy a novinky ASP.NET

Výhody technologie ASP.NET jsem zmínil již v předcházející části, abych demonstroval, jakým způsobem byl Microsoft inspirován při vývoji této technologie. Následující seznam stručně shrnuje podstatné výhody ASP.NET, tak jak je propaguje společnost Microsoft.

- **Objektový model programování** – Je možno používat všechny výhody, které přináší objektové programování – dědičnost, přetěžování metod, překrývání metod, polymorfismus… Objektový model přináší obecnější využití již napsaného kódu, což snižuje čas na psaní zdrojového kódu.

[1] Aplikace WinFX jsou speciálně určené pro systém Windows Vista

- **Common language runtime (CLR) –** Webovou aplikaci je možno psát v libovolném jazyce podporovaném platformou .NET (Visual C#, Visual Basic, Visual C++, Visual J#…) a jazyky je možno kombinovat v rámci jedné aplikace. Toto vlastnost má však daleko větší význam, než se na první pohled zdá. Vývojář může například napsat jedinou knihovnu s vlastními komponentami a tu pak využívat v klasické Windows aplikaci, v aplikaci pro PDA zařízení nebo ve webové aplikaci aniž by se musel obávat problému s kompatibilitou, propojením technologií nebo verzováním.
- **Kompilovaný kód** - Veškerý zdrojový kód je kompilován do sestavení s pevně danou verzí. Velké množství chyb je odhaleno ještě před spuštěním aplikace.
- **Obrovské množství hotových řešení** – Platforma .NET Framework nabízí velmi rozsáhlé knihovny s hotovými komponentami, které je možno přímo použít v aplikaci. Komponenty navíc obsahují rutiny pro spolupráci s vývojovým prostředím Visual Studio, a tak je možno řešit složité programátorské problémy bez nutnosti napsat jediný řádek zdrojového kódu (tzv. deklarativní způsob).
- **Možnost vývoje všech součástí pomocí jediného IDE** – Pro vývoj na platformě .NET nabízí Microsoft vývojové prostředí Visual Studio 2005. Jedná se o velmi mocný nástroj na vytváření aplikací na této platformě, který umožňuje využití všech prostředků, které .NET Framework nabízí. Kromě standardních nástrojů jako je debugger, vizuální editor apod. může vývojář využívat dokonalou predikaci zdrojového kódu (tzv. **IntelliSense**), dále tzv. „**útržky**“ (předpřipravené části zdrojového kódu), XML **Custom tools** (uživatelské programy, které je možno spouštět nad XML soubory umožňující například automatické generování souborů apod.), **regiony** (direktivy Visual Studia pro ohraničování částí zdrojového kódu a možnost jejich minimalizování do jediného řádku), **XML komentáře** zdrojového kódu (slouží pro zobrazování v IntelliSense nebo pro generování programátorské dokumentace), **vývojový webový server** pro ladění aplikací a spousta dalších užitečných utilit a nástrojů.
- **Více než 50 nových uživatelských ovládacích prvků** – Od předchozí verze došlo k výraznému rozšíření počtu uživatelských ovládacích prvků, které je možno využívat k tvorbě webových aplikací. Jde například o `LoginBox`, `TreeView`, `GridView`, `Content`, `ContentPlaceHolder` a jiné.
- **Nová technologie WebParts** – WebParts jsou jednotlivé části webové stránky, jejichž vzhled a jiná nastavení může měnit přímo uživatel webové aplikace. Uživatel

může WebParts přesouvat, minimalizovat, zavírat nebo nastavovat jejich vzájemné provázání. Tato technologie přímo souvisí se současnými požadavky na uživatelské přizpůsobení obsahu a vzhledu webové aplikace.

- **Nový framework pro vytváření stránek** – ASP.NET 2.0 umožňuje implementovat vizuální dědičnost stránek pomocí technologie MasterPages (hlavních stránek). Dále jsou k dispozici propracované systémy témat a skinů, lokalizace a také nový způsob kompilace webové aplikace.

1.1.3.2 Hlavní cíle ASP.NET

Vývojový tým ASP.NET si stanovil jasné cíle, kterých má technologie dosáhnout. Následující část stručně shrnuje základní cíle ASP.NET:

- **Zvýšení produktivity vývojáře** – Cílem je, aby se vývojář mohl zaměřit více na řešení konkrétních problémů na úrovni vyvíjené aplikace a nemusel vymýšlet řešení problémů, jejichž schémata se opakují při vývoji každé aplikace. Příkladem je stránkování zobrazovaných dat ve webovém prohlížeči. Vývojář na platformě skriptovacího jazyka ASP musí nejprve vymyslet, jak tento problém vyřešit, pak musí napsat obslužné procedury a teprve potom může tuto funkcionalitu použít. Vývojář na platformě ASP.NET může pomocí několika kliknutí myší povolit stránkování na komponentě `GridView` a dosáhne stejného efektu, aniž by napsal jediný řádek zdrojového kódu. Výše uvedený příklad je sice velmi jednoduchý a samoúčelný, ale demonstruje jak rychle a jednoduše je možno vytvářet webové aplikace na platformě .NET. Technologie nabízí také spoustu hotových vývojových infrastruktur jako například správu členství a rolí, zosobnění, navigace po webovém sídle, monitorování chodu apod.
- **Správa webových aplikací a konfigurace** – Microsoft implementoval do .NET Framework 2.0 nová API rozhraní, která mohou vývojáři používat při řízení, správě a konfiguraci svých webových aplikací. Microsoft nabízí i hotová řešení pro konfiguraci webových aplikací – například zásuvný modul pro Microsoft Management Console (MMC), který je dostupný například přes konfiguraci webové aplikace ve správě IIS. Dále Microsoft nabízí „Web Site administration Tool“, což je utilita pro konfiguraci webové služby s webovým rozhraním. [3]

- **Nejrychlejší server webových aplikací na světě** – Tento smělý cíl byl naplněn mnoha výkonnostními vylepšeními. Například se jedná o vylepšenou možnost cachování dat z SQL serveru. Platforma ASP.NET 2.0 přináší nově možnost zrušení platnosti SQL mezipaměti. Tato technologie umožňuje svázání závislosti exspirace paměti cache přímo s SQL serverem[1]. Dále se jedná o možnost provozovat ASP.NET aplikace na 64bitových procesorech. [3]
- **Zvýšení bezpečnosti** – Technologie ASP.NET zavádí integrovanou správu členství a rolí. Celý mechanizmus je možno jednoduše napojit na datový zdroj s uživatelskými účty (SQL databáze, XML, MS Access, textový soubor) a na základě ověření uživatele pak přidělovat oprávnění k určitým částem webového obsahu. K dispozici jsou dokonce hotové ovládací prvky pro přihlašování nebo pro změnu profilu přihlášeného uživatele. [3]

1.1.4 Přehled základních prostředků pro vývoj webových aplikací v ASP.NET

Tato část stručně popisuje základní koncepci vývoje aplikací pomocí technologie ASP.NET. Tato koncepce je značně odlišná od „současné klasické“ koncepce tvorby webových stránek a webových aplikací. Obecně platí zásada, že vývojář musí bez výjimky přistoupit na pravidla této nové technologie. Jedině v tom případě bude tato technologie užitečná a zajistí splnění výše popsaných cílů. Jakékoliv nestandardní postupy na sebe navážou nutnost vytváření dalších nestandardních postupů, což v konečném důsledku může degradovat ASP.NET aplikaci na tvar, který nepřináší žádná pozitiva.

1.1.4.1 Objektový model programování

Vývojář musí plně chápat objektový model programování. Jakékoliv pokusy o implementaci procedurálního způsobu programování vedou k narušení struktury aplikace. Veškeré jazyky pro platformu .NET jsou plně objektové. Při vytváření nových komponent by se měl vývojář snažit o maximální využití dědičnosti a polymorfismu. Zjednodušeně se dá říci, že je nutno se na webovou stránku začít dívat jako na uspořádanou množinu objektů, které „žijí vlastním životem“ a úkolem vývojáře je pouze vytvářet vazby mezi nimi

[1] Tato funkcionalita je dostupná pouze Microsoft SQL server 7.0 a vyšší

nebo je rozšiřovat o nové funkcionality. Pohled na webovou stránku jako na algoritmus, který má vygenerovat výsledné HTML, musíme na platformě .NET zavrhnout ještě před započetím programování.

1.1.4.2 Princip webových formulářů (WebForms)

Webové stránky technologie ASP.NET jsou implementovány pomocí tzv. webových formulářů. Webový formulář je soubor s příponou „.aspx“. Tento soubor obsahuje kód, který se podobá jazyku HTML. Na následujícím obrázku je příklad zdrojového kódu jednoho webového formuláře:

```
<%@ Page Language="C#" AutoEventWireup="true"  CodeFile="Default.aspx.cs" Inherits="_Default" %>

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

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
        <title>Untitled Page</title>
</head>
<body>
        <form id="form1" runat="server">
                <asp:Button ID="Button1" runat="server" Text="Button" />
        </form>
</body>
</html>
```

Obrázek 1 – Zdrojový kód webového formuláře (WebForm)

Na první pohled se zdá, že webový formulář není nic jiného než klasická webová stránka. Jsou zde však zásadní odlišnosti. Na webový formulář je možno vkládat tzv. serverové ovládací prvky. V příkladu vidíme jediný ovládací prvek , a sice „Button“. Po vložení prvku do stránky se tento stává automaticky součástí serverového objektového modelu a k prvku je možno přistupovat podobně jako jsme zvyklí například z DOM modelu při skriptování na straně klienta. Velkou část práce můžeme vykonat pouhým naskládáním prvků do stránky a nastavením příslušných atributů. Tyto operace můžeme provádět také pomocí vizuálních nástrojů Microsoft Visual Studia 2005.

K webovému formuláři může (a nemusí) náležet tzv. kód na pozadí (code behind). Kód na pozadí je soubor s příponou „.cs“[1], což indikuje kód v jazyce C#. Kód na pozadí

[1] Code behind nemusí být pochopitelně psaný v jazyce C#, ale v libovolném jiném jazyce podporovaném v prostředí .NET

obsahuje programové rutiny, obsloužení událostí apod. Tento kód je kompilován do DLL knihoven. Pomocí kódu na pozadí můžeme přistupovat k jednotlivým objektům webového formuláře a měnit jejich vlastnosti (například napojovat datové zdroje, přidávat sloupce do tabulek, provádět výpočty apod.).

Při dotazu na webový formulář (ASPX soubor) je navrácen vygenerovaný HTML kód. Jednotlivé serverové ovládací prvky se „vyrenderují“ do HTML. Je nutné si uvědomit, že veškerý generovaný HTML kód má na starosti prostředí ASP.NET. Vývojář by nikdy neměl použít standardní výstup do objektu Response tak, jak je to standardní u skriptovacích jazyků. Následující ASP kód ukazuje postup, který je v ASP.NET zcela nevhodný (ne však nerealizovatelný):

```
<%
Response.Write "<table bgcolor='blue'>"
Response.Write "<tr><td style='color: red'>Obsah nějaké buňky</td></tr>"
Response.Write "</table>"
%>
```

V ukázce se HTML kód výsledné stránky generuje přímo do objektu HTTP response (odpověď přes HTTP protokol). V prostředí ASP.NET bychom spíše použili serverový ovládací prvek „TABLE“, kterému bychom nastavili potřebné vlastnosti.

1.1.4.3 Serverové ovládací prvky (WebControls)

O WebControls jsem se již zmínil v předchozí části. Všechny WebControls jsou součástí jmenného prostoru `System.Web.UI.WebControls` architektury .NET Framework. WebControls jsou popsány třídami, které jsou zkompilované v knihovně System.Web.UI.WebControls.dll. Použití WebControls v našem webovém formuláři v podstatě znamená vytvoření instancí těchto tříd a jejich zapojení do objektového modelu webového formuláře. Vnitřní funkcionalita WebControls je zapouzdřená v objektech (black box) a programátor může pracovat jen s veřejnými položkami objektů nebo s rozhraními.

Programátor si může vytvářet třídy s vlastními ovládacími prvky. Podmínkou je zdědit třídu `System.Web.UI.WebControls.WebControl` nebo libovolnou jinou třídu, která tuto třídu dědí. Tento způsob vytváření ovládacích prvků spočívá v psaní konkrétního kódu například v jazyce C#. Programátor má plnou kontrolu nad generovaným HTML

kódem. Dále je možno podobným způsobem vytvářet tzv. „složené ovládací prvky“ nebo „šablonové ovládací prvky“.

Další možností jak vytvářet vlastní ovládací prvky spočívá ve vytváření tzv. „uživatelských ovládacích prvků“. Tento způsob je přívětivější a časově méně náročný, protože programátor pouze „naskládá“ již existující ovládací prvky na nový „formulář“. Tento „formulář“ pak tvoří „balíček“, který je možno umisťovat do webových formulářů jako standardní ovládací prvky. Nevýhodou je, že prvky nemůžeme uložit do DLL knihovny a používat je jako reference v jiných projektech[1].

1.1.4.4 Framework na tvorbu webových formulářů, master pages

ASP.NET 2.0 přinesla řadu vylepšení pro vytváření webových formulářů. Přibyla především podpora vizuální dědičnosti stránek. Vizuální dědičnost je implementována pomocí tzv. hlavních stránek (master pages). Master page je stránka, která určuje základní vzhled webové aplikace a jedná se o šablonu, kterou můžeme implementovat do dalších stránek. Vnořování master pages je možné do libovolné úrovně. Master page spočívá v tom, že na ni můžeme umístit tzv. bílá místa (ContentPlaceHolders). Obsah těchto bílých míst můžeme definovat ve stránce, která dědí danou master page. Pokud obsah bílých míst nedefinujeme, je zobrazen výchozí obsah specifikovaný v master page. Master page je také vhodným místem pro umístění globálních objektů a definic, které chceme používat ve všech stránkách, které danou master page dědí.

1.1.4.5 Práce s daty, ADO.NET

ADO.NET je nová verze datové vrstvy, která už v ASP.NET 1.0 nahradila původní koncepci ADO používanou například v ASP aplikacích. ADO i ADO.NET nabízí stejnou škálu základních objektů[2]. ADO.NET navíc přináší řadu vylepšení a také velké množství zcela nových objektů, které jsou přímo přizpůsobeny pro platformu .NET.

---

[1] Je možné přeložit daný formulář a pak v dočasném adresáři prostředí .NET nalézt příslušnou DLL knihovnu uživatelského ovládacího prvku. Ke zdrojovému kódu se však nedostaneme.

[2] Bohužel však základní objekty vrstvy ADO a ADO.NET nejsou navzájem kompatibilní. Například komponenta psaná na platformě .NET nemůže sdílet stejný objekt DbConnection jako používá ASP aplikace, která využívá danou komponentu (jako COM objekt).

ADO.NET umožňuje napojit aplikaci na téměř libovolný datový zdroj (databázi). Nejvíce preferovaný datový zdroj je pochopitelně Microsoft SQL server, ale není problém využít Oracle server, jakýkoliv ODBC nebo OLEDB datový zdroj nebo také Windows SQL CE. Novinkou v ADO.NET jsou tzv. „ovládací prvky datových zdrojů“. Jedná se o klasické serverové ovládací prvky (tj. dědí nejprve z `DataSourceControl` a poté z `Control`). Tyto prvky je pak možno použít jako zdroje dat pro jiné ovládací prvky bez nutnosti napsat jediný řádek programového kódu. Většinu operací pro vytváření a napojování datových zdrojů je také možno realizovat pomocí průvodců. Programátorovi stačí definovat SQL dotazy pro tzv. „CRUD“ operace. Ovládací prvek pak sám dokáže využít přiřazeného ovládacího prvku datových zdrojů a vyvolává potřebné SQL dotazy na SQL server.

1.1.4.6 Správa členství a rolí (membership service a roles service)

Tyto dvě nové služby umožňují spravovat uživatele webového sídla a přiřazovat jim role. Uživatelům nebo rolím je pak možno udělovat oprávnění v rámci aplikace. Můžeme tak odlišit, jaký obsah uvidí registrovaní a anonymní uživatelé, můžeme omezit přístup k určitým stránkám jen pro určité uživatele nebo role, omezit funkcionalitu v kódu apod. Jako datový sklad těchto informací slouží primárně databáze na MS SQL serveru, ale je možno si napsat své vlastní poskytovatele (providery) na libovolný datový zdroj. Přidání podpory správy členství a rolí do SQL databáze spočívá v přidání několika tabulek a stored procedur. Tuto operaci je možno provádět pomocí příkazu aspnet_regsql.exe s patřičnými parametry.

Pro potřeby těchto nových služeb existuje podpora i mezi ovládacími prvky. Jedná se například o prvky `Login`, `LoginView`, `LoginStatus`, `LoginName`, `CreateUserWizard` apod. Tyto prvky se vizuálně tváří nejčastěji jako formuláře, pomocí kterých se může uživatel zaregistrovat do aplikace, přihlásit, měnit svoje údaje, resetovat hesla apod. Všechny prvky umožňují velké množství nastavení a je možno je přizpůsobit dle vlastních nároků a to nejen vzhledově, ale hlavně funkčně a obsahově. Veškerá logika ověřování vůči databázi[1], kontrola vyplněných údajů (validace), správa session objektů

[1] Standardně je implementováno zabezpečené zasílání a ověřování hesla v HASH podobě

dané relace apod. je zapouzdřena přímo v prostředí ASP.NET a není nutné ji programově implementovat. Programátor má při tvorbě aplikace dostupný objekt typu `MembershipUser`, ze kterého zjistí všechna potřebná data (jméno, datum vytvoření, datum poslední aktivity, datum poslední změny hesla, příslušnost k rolím apod.). Pokud tento objekt není naplněn, jedná se o anonymního uživatele, který neprošel procesem ověření[1].

Veškeré operace, které nám nabízí „Login" ovládací prvky je možno pochopitelně provádět také pomocí API funkcí[2]. Třídy a implementace pro správu členství jsou dostupné v oboru názvů `System.Web.Security.Membership`, pro správu rolí v oboru názvů `System.Web.Security.Roles`. Aby bylo možné administrovat uživatele, přiřazovat je k rolím a provádět jiné potřebné operace, není nutné si vytvářet vlastní administrativní moduly ve webovém sídle. K dispozici je hotová webová aplikace „ASP.NET Web Site Administration Tool", která je standardně k dispozici na každém IIS webovém serveru s instalovanou podporou pro .NET Framework 2.0. K aplikaci je možno přistupovat přes webové rozhraní po zadání adresy ve tvaru http://server/nazev_aplikace/Webadmin.axd.

1.1.4.7 Personalizace a přizpůsobení webového sídla, webparts

Ve světě Internetu je kladen velký důraz na uživatelské přizpůsobení webových sídel – uživatel si může sám zvolit, které části stránky uvidí a které ne, může si zvolit vzhled webové prezentace, přizpůsobit si nastavení prvků na stránce apod. Všechny tyto funkcionality jsou přímo zabudované do technologie ASP.NET 2.0. Tyto implementace jsou soustředěny do oboru portálových rámců (portal frameworks) a webových částí (webparts), které pak pro ukládání personalizačních údajů používají službu „personalizace".

Služba personalizace slouží k velmi efektivní správě informací, které je nutno uchovávat specificky pro každého návštěvníka webového sídla. Dříve byla tato funkcionalita implementována pomocí cookies, session nebo application objektů. Cookies slouží především pro ukládání trvalých informací o daném klientovi, které je možno využít na-

[1] I pro neověřené uživatele je možno stanovit oprávnění.

[2] V praxi se jedná o opačný proces – ovládací prvky používají dané API funkce

příklad při příští návštěvě webového sídla, naopak session a application objekty slouží pro uchování dočasných informací pro danou relaci resp. životnost aplikace na serveru. Tyto objekty mají velkou slabost v netypovosti ukládaných dat – po uložení dat do session se z nich stává objekt, který nazpět získáme pouze přetypováním. Navíc zde může nastat stav označovaný jako „session hell“, kdy se používání session vymkne vývojovému týmu kontrole a objekt obsahuje nesystematická roztroušená a duplicitní data. Objekty cookies mají nedostatky v tom, že nemusí být například v klientském prohlížeči povoleny nebo mohou být jednoduše uživatelem změněny. Navíc se uživatel může pohybovat mezi více počítači, kde cookies objekty naprosto nedostačují. Služba personalizace řeší všechny tyto problémy. Programátor může velmi jednoduše pomocí souboru web.config nakonfigurovat přísně typový personalizační profil, včetně možnosti vytvářet skupiny personalizačních dat, nastavení výchozích hodnot, nastavení položek jen pro čtení apod. K tomuto profilu je možno velmi jednoduše programově přistupovat (dokonce je možno využít i IntelliSense). Personalizační profily nejsou uloženy v session objektech ani v cookie objektech, ale v datovém skladu na straně serveru.

Portálové rámce a webové části přímo využívají službu personalizace k uložení nastavení a vzhledu webového sídla specificky pro každého registrovaného uživatele. Základem je rozdělení webové stránky na zóny. Do těchto zón je pak možno vkládat jakékoliv ovládací prvky, které se automaticky stávají webovými částmi (webparts). Je možno vytvářet také speciální ovládací prvky dědící ze třídy `System.Web.UI.WebControls.WebParts.WebPart`, kterým je možno implementovat speciální vlastnosti pro umožnění změny nastavení konkrétního ovládacího prvku. Uživateli je pak umožněno, aby si jednotlivé webové části zobrazoval, zavíral, přesouval (pomocí drag&drop) mezi jednotlivými zónami, měnil jejich nastavení a vytvářel mezi nimi vazby. Typickými příklady webových částí jsou kalendář, informace o počasí, seznam novinek apod. Vazba mezi webovými částmi může být například to, že webová část „novinky“ zobrazuje novinky jen z regionu, který je nastaven ve webové části „informace o počasí“. Všechna nastavení provedená uživatelem jsou ukládána do datového skladu na straně serveru a jsou tedy zachována i po odhlášení uživatele.

1.1.4.8 Caching dat na straně serveru

Caching má za úkol uchovávat data na straně serverové aplikace mezi zobrazením jednotlivých stránek bez nutnosti je pokaždé vytvářet nebo načítat z datového zdroje. Načítání dat z datového zdroje může být časově velmi náročné a navíc objekty pro spojení do databáze jsou nákladné na systémové prostředky. Systém cachování má za úkol zrychlit odezvu aplikace a také šetřit systémové zdroje na straně serveru tím, že sníží počet náročných interakcí s datovým zdrojem nebo tím, že uchová již vygenerované HTML stránky.

Problémem u kešovaných dat je jejich aktuálnost vůči datovému zdroji. Je proto nutné používat kešování s uvážením, zda zastaralá data nemohou způsobit problémy uživateli aplikace. Často je nutno zajistit, aby uživatel viděl vždy aktuální data. ASP.NET 2.0 přináší nové vlastnosti, které umožňují vázat závislost platnosti a vypršení paměti cache na stav datových zdrojů, ze kterých data pochází.

Podstatnou novinkou v nové verzi je přidání třídy `SqlCacheDependency`, která má na starost závislost platnosti paměti cache na datech na SQL serveru. Princip je takový, že pokud se změní data na SQL serveru, pak dojde automaticky k vypršení paměti cache. Je třeba si uvědomit, že je zde rozdíl mezi MS SQL serverem 2005 a předchozími verzemi. Ve starších verzích SQL serverů se v databázi udržuje tabulka, která nese informace o tom, kdy došlo ke změně ve sledovaných tabulkách v databázi. Tato tabulka je plněna triggery na sledovaných tabulkách. Aplikace ASP.NET se pak v malých časových úsecích (výchozí je 0,5 s) dotazuje na tuto tabulku. Jestliže došlo ve sledované tabulce ke změně, označí závislou cache paměť za neplatnou. Toto zjišťování probíhá v samostatném vlákně ASP.NET aplikace. U SQL serveru 2005 je tato funkcionalita zajištěna opačným směrem. SQL server je přímo vázán na aplikaci a odesílá jí události o změnách v datech, která danou aplikace „zajímají“ resp. o takových změnách, k jejichž sledování se daná aplikace zapsala. Tato hlášení ze strany serveru jsou vázána na objekt `SqlCommand` v ASP.NET aplikaci, pomocí kterého získáváme data z SQL serveru. Z tohoto objektu dokáže SQL server rozpoznat, o jakých změnách dat má aplikaci informovat. Jsou zde však značná omezení ve vystavění SQL dotazu daného příkazu. Nemůžeme používat například znak „*“ pro získání všech sloupců z tabulky, není možno používat agregační funkce, LEFT OUTER JOIN operace, odkazy na serverové proměnné, velké datové typy jako ntext, text,

image, dočasné tabulky, výpočtové sloupce, dotazy na VIEW a podobně. Veškerá omezení jsou podrobně popsána v MSDN library[1]. Pokud pro závislost použijeme dotaz, který nesplňuje výše uvedené požadavky, dojde po vykonání SQL dotazu a uložení dat do paměti cache k okamžitému vypršení paměti cache a při příštím načtení stránky je nutno data opět načíst z databáze.

1.2 Systémy CMS

Zkratka CMS pochází z anglického „Content Management System“ - přeloženo do češtiny – systém pro správu obsahu. U nás jsou tyto systémy známy spíše jako „redakční“ nebo „publikační“ systémy. Základním úkolem CMS je umožnit běžným uživatelům udržovat aktuální obsah webového nebo intranetového sídla bez nutnosti zasahovat do zdrojového kódu aplikace. [14]

Na CMS systémy jsou kladeny ze strany uživatelů vysoké nároky a mnoho velkých společností si nechává programovat CMS „na míru“ svým potřebám. Velký důraz je kladen také na zabezpečení přístupu k aplikaci a na celkovou bezpečnost dat. Pokud se podobné aplikace nasazují například do bank nebo finančních institucí, tak většinou prochází přísným bezpečnostním auditem na všech vrstvách. Pro vytváření CMS je tedy nutno velmi dobře zvážit použité technologie.

1.2.1 Možnosti architektury spojení CMS a webového sídla

Základní otázkou, kterou je nutno před zahájením vývoje CMS vyřešit, je základní architektura celého systému a jeho vazba na webové sídlo. CMS můžeme chápat jako sadu nástrojů, které nám umožňují vytvářet a ukládat obsah webového sídla – to znamená, že nám umožňují provádět administrativní úkony. Otázkou však zůstává, která část aplikace by se měla starat o zobrazování dat koncovému uživateli. Pokud bychom tuto funkcionalitu přenechali plně webovému sídlu (tj. ne CMS), pak se nevyhneme stavu, kdy bude nutné veškeré zobrazovací moduly programovat separátně pro každé webové sídlo, ve kterém budeme chtít CMS použít. Navíc je nutné si uvědomit, že i administrativní část

[1] Například na adrese http://msdn2.microsoft.com/en-us/library/ms181122.aspx

potřebuje zobrazení již existujících dat v systému. Určitě by nebylo vhodné programovat zobrazení dat zvlášť ve webové aplikaci a zvlášť v CMS.

1.2.1.1 Architektura se společným GUI

Pod označením GUI je chápáno grafické uživatelské rozhraní. Tato architektura spočívá v tom, že zobrazování i administrace dat probíhá pomocí jediného uživatelského rozhraní konkrétního modulu aplikace. Modul pak musí umět na základě ověření uživatele zobrazovat patřičné ovládací prvky. Pokud se na modul dívá anonymní návštěvník webového sídla, pak se modul „přepne“ do zobrazení „jen pro čtení“. Pokud se dívá uživatel ověřený jako administrátor, pak modul umožní ve stejném GUI administrátorské operace.

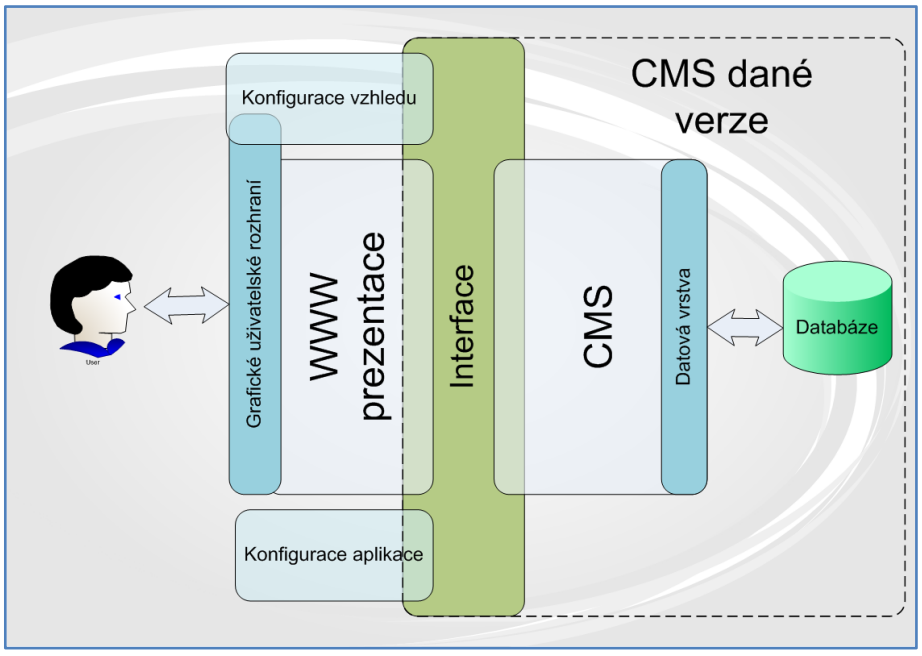
Mohl by vzniknout dojem, že pokud chceme vytvářet náš CMS podle této architektury, tak je nutno vyvíjet webové sídlo s CMS jako jednu aplikaci (jedno uživatelské rozhraní). A skutečně. Spousta webových sídel (především menších) je vytvořena tímto mechanizmem. Bohužel je zde zásadní problém – v podstatě se nejedná o CMS, ale pouze o administrovatelné webové stránky. Celá logika CMS je smíchána v jednom zdrojovém kódu s webovou prezentací. CMS systém nelze nikdy odejmout nebo jej nahradit novou verzí. Povýšení verze CMS znamená současně povýšení verze aplikace a naopak. Také není možno CMS nabízet jako samostatný produkt.

Z předchozího odstavce by mohl vzniknout dojem, že tato architektura je zcela nesprávná. Nesprávné jsou však spíše prostředky, které se většinou k dosažení této architektury používají. Pokud bychom chtěli postupovat správně je nutno dodržet tyto předpoklady:

- **Separování kódu CMS –** Veškerý kód, který se stará o zobrazování, editaci apod. je nutno umístit do separátních souborů, které jsou navíc odděleny zcela mimo složku s webovou prezentací. Ideální je stav, kdy se CMS kompletně nachází v jedné složce, kterou můžeme kdykoliv přepsat novou verzí, aniž bychom se museli obávat, že přijdeme o údaje specifické k dané webové prezentaci.
- **Oddělení vzhledu od dat –** Moduly CMS nesmí za žádných okolností generovat graficky formátovaný výstup. Grafické formátování je totiž specifické pro každou webovou prezentaci, ve které je daný CMS použit. CMS nám může maximálně nabídnout různé typy zobrazení a rozložení administrátorských prvků. Definice barev,

grafických objektů apod. je nutno ponechat výhradně na webové prezentaci. Nejčastějším „médiem“ grafického formátování jsou tzv. „kaskádové styly“ (CSS). CMS pak vygeneruje pouze „čisté HTML“ a nabídne webové prezentaci sadu tříd CSS stylů, které je možno definovat a tím změnit vzhled modulů. V současné době přináší technologie CSS naprosto nepřekonatelné možnosti formátování HTML dokumentů a stále více se dostává do popředí zájmu vývojářů. Stále však přetrvává problém nejednotné interpretace CSS stylů v různých webových prohlížečích.

- **Separování konfigurace a nastavení –** CMS jako celek může nabízet velkou množinu nejrůznějších nastavení. Programátor by se měl snažit vyseparovat co nejvíce možných nastavení „ven“ z kódu. Jakékoliv napevno zadané hodnoty a konstanty v kódu se později mohou stát překážkou při rozšiřování funkcionalit nebo při nasazení k jiným zákazníkům, kteří vyžadují jiné chování. Nikdy by neměl nastat případ, že se vývoj v takových případech rozdělí a pro každého zákazníka je vedena jiná větev vývoje s mírně odlišnou funkcionalitou. Kód musí být vždy naprosto stejný u všech zákazníků a CMS se liší pouze konfiguračními soubory. Konfigurační soubory CMS je vhodné umístit do jiné složky, než je složka, ve které jsou umístěny zdrojové kódy CMS, aby nemohlo dojít k nechtěnému přepsání konfigurace.



Obrázek 2 – Schéma architektury CMS se společným GUI

Obrázek 2 schematicky zobrazuje interakci webové prezentace a CMS systému v této architektuře. Ze schématu je vidět, že CMS systém je chápan jako „balík“, který v sobě zapouzdřuje zdrojové kódy systému[1], vrstvu pro přístup do databáze, samotnou databázi a hlavně rozhraní (interface) pro webovou aplikaci. Toto rozhraní je nutno chápat spíše abstraktně, ale není záhodno podceňovat jeho existenci. Ve skutečnosti se nemusí jednat o žádný kód, ale o pomyslná pravidla, která musí dodržovat daná webová aplikace, aby mohla s CMS správně pracovat. Interface můžeme také chápat jako souhrn veřejných (public) funkcionalit, které CMS nabízí webové prezentaci. Rozhraní by mělo na programové úrovni být zachováno stále stejné – změna vnitřní logiky CMS nemá vliv na chování webové aplikace. Pokud se CMS vyvíjí a nová verze nabídne nové funkcionality, je pochopitelně nutné změnit rozhraní mezi aplikací a CMS. Nová verze rozhraní by však vždy měla zachovávat zpětnou kompatibilitu k původnímu rozhraní. Jinak řečeno – nová verze CMS by měla nabídnout nové funkcionality nebo možnost řešit staré funkcionality jiným způsobem. Původní funkcionality by však měly být zachovány. V diagramu jsou dále vidět objekty „konfigurace CMS“ a „konfigurace grafického vzhledu“. I tyto objekty mají vazbu na interface mezi CMS a webovou aplikací – interface nám pouze stanovuje, jaké konfigurační možnosti poskytuje daná verze CMS. Při nasazení nové verze nesmíme opomenout přidat definice nových konfiguračních možností do konfiguračních souborů.

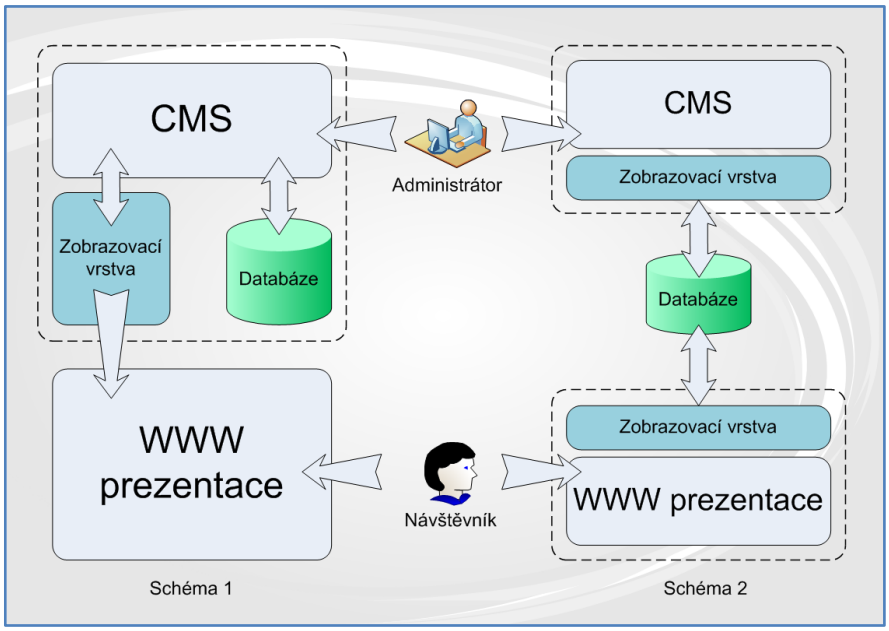
1.2.1.2 Architektura s odděleným GUI

Tato architektura vyžaduje existenci dvou aplikací. První aplikací je samotná webová prezentace a druhou je redakční systém. Webová prezentace slouží výhradně k zobrazování dat. Editace obsahu probíhá v oddělené aplikaci. Je zde nutné opět vyřešit spojovací element mezi webovou prezentací a redakčním systémem. I když je webová prezentace grafickým uživatelským rozhraním zcela oddělena od CMS, stále musí mít možnost dostat se k datům z databáze a zobrazovat je. Zobrazovací vrstva musí být znovuvyužitelná v různých webových aplikacích, a proto musí být součástí redakčního systému. Pokud je to možné, je velmi vhodné, aby stejnou zobrazovací vrstvu využíval redakční systém i webová aplikace. Je nutno si uvědomit, že tato architektura nám umožňuje vytvořit CMS

[1] Nebo binární kódy v DLL knihovnách – to je specifické pro použitou technologii.

na odlišné platformě, než je webová aplikace. Redakční systém může být implementován například jako klasická tlustá aplikace. V takovém případě je nutno realizovat dvě zobrazovací vrstvy – jedna pro potřeby CMS a druhá pro potřeby webové prezentace. Realizace CMS jako tlusté aplikace není příliš častá. Zásadní nevýhodou je nepřístupnost administrace odkudkoliv z Internetu. Administrátor musí mít aplikaci nainstalovanou na počítači, na kterém momentálně pracuje. Navíc přibývají problémy s publikováním nových verzí – je nutno vytvářet instalační balíčky a šířit je mezi zákazníky. Za výhody tohoto řešení můžeme považovat například vyšší zabezpečení (administrace není dostupná z veřejného Internetu) nebo možnost využít pokročilejšího uživatelského rozhraní. Většinou však převáží požadavek na dostupnost administrace přímo z Internetu a CMS je pak kompletně realizován na platformě WWW.

Následující diagram schematicky ukazuje implementaci webové prezentace a CMS podle této architektury:



Obrázek 3 – Schémata architektur CMS s odděleným GUI

Na prvním schématu je vidět architektura, kdy je CMS i webová aplikace vytvořena na stejné platformě a může tedy sdílet jedinou zobrazovací vrstvu. Zobrazovací vrstva může být skupina skriptů nebo například DLL knihovna s komponentami. Na druhém schématu je zobrazena realizace CMS na odlišné platformě, než je platforma webové aplikace. Je vidět úplné oddělení redakčního systému od koncové spravované aplikace. Patrná je nutnost dvou zobrazovacích vrstev.

1.2.1.3 Další architektury CMS

Kromě výše uvedených architektur se můžeme setkat ještě s dalšími přístupy k implementaci CMS. Existují například tlusté aplikace, které umožňují editovat data, ukládat je do databáze a poté vygenerovat kompletní HTML prezentaci, kterou je možno umístit na webový server. Výsledkem je čistě statický HTML kód. Výhodou je značná rychlost odezvy koncovému návštěvníkovi webové prezentace, protože odpadají přístupy do databáze nebo generování HTML „on fly“. Dále jsou takové webové prezentace velmi dobře optimalizované pro internetové vyhledávací roboty (SEO). Zřejmou nevýhodou je statická podoba prezentace – není možno dynamicky měnit data například pomocí vyhledávání nebo řazení. Je zřejmé, že tato architektura má své místo spíše na poli webových prezentací, kde se data aktualizují v řádech dní. Příkladem tohoto řešení může být například WebGenerator ProSite[1].

1.2.2 Požadavky na CMS, základní moduly CMS

Tato kapitola shrnuje základní požadavky na CMS, tedy to co by měl CMS umět. Zabývá se také popisem základních modulů, které by měly být součástí každého redakčního systému.

1.2.2.1 Administrace dostupná on-line

Tento požadavek je zcela zásadní. Tlusté aplikace v tomto směru velmi zaostávají. Veškeré operace včetně importů a exportů dat by měly být proveditelné přes webový prohlížeč. Zde je důležité si uvědomit, že některé administrativní moduly mohou být velmi náročně na kompatibilitu prohlížečů, protože mohou obsahovat velké množství klientského skriptovacího kódu. Je vhodné stanovit požadavky na prohlížeč pro veřejné prohlížení (bez administrativních prvků) a pro administraci. Logicky pro administraci budou požadavky více konkrétní na specifický prohlížeč.

[1] Stránky výrobce - http://www.webgenerator.nl

1.2.2.2 WYSIWYG editace obsahu

„What You See Is What You Get“ – „Co vidíš, to dostaneš“. Cílem je, aby běžný uživatel měl možnost formátovat obsah webových stránek stejným způsobem jako v klasických aplikacích pro sazbu textu (např. MS Word). Pro tento účel existuje spousta řešení. Nejjednodušší aplikace jsou psány v JavaScriptu. Existují ale i nejrůznější JavaApplety nebo ActiveX komponenty.

1.2.2.3 SEO – optimalizace pro webové vyhledávače

SEO dnes hraje velmi významnou roli, zvláště ve spojení s internetovým obchodováním. Webové vyhledávače, jako například Google, neustále „skenují“ obsah Internetu a ukládají informace o nalezených webových stránkách do vlastních databází. Pokud uživatel použije daný webový vyhledávač, nabízí mu systém odpověď podle určitých pravidel. Základem je, aby webový vyhledávač měl ve své databázi uložené správné informace. Aby mohl webový vyhledávač správně načíst data, musí být webové stránky pro vyhledávač „přívětivé“. Toho lze docílit například naprosto validním HTML kódem nebo přepisováním URL. Přepisování URL spočívá v tom, že webová aplikace dokáže překládat nepřívětivá URL, jako například URL s parametry pro webové skripty, na čistá URL obsahující pouze čitelný text. Cílem je, aby se v URL vyskytla klíčová slova pro obsah, který se na daném URL nachází. Pokud se v URL nachází klíčové slovo, pak je to webovými vyhledávači bráno jako významný argument a stránka se dostává na přední místa ve vyhledané množině dat. Velký vliv na webové vyhledávače má také optimalizace kódu pro XHTML.

1.2.2.4 Fulltextové vyhledávání

Tento požadavek je také zcela zásadní. Vyhledávání má význam nejen na velkých webových portálech, ale také na standardní firemní webové prezentaci. Vždy je potřeba si uvědomit, že web bude tak úspěšný, jak rychle na něm budou uživatele schopni nalézt požadovaná data. Implementace fulltextového vyhledávání není zcela triviální záležitost a často je výhodnější použít nějakou existující komponentu. Je například možno využít portálu Google, který umožňuje hledat data jen v určité doméně.

1.2.2.5 Antispamová ochrana

Redakční systém většinou nabízí velké množství formulářů, pomocí kterých je možno odesílat data na server. Tyto formuláře se často stávají cílem útoků „webových pavouků“, kteří se do nich pokouší zadávat nejrůznější odkazy na komerční stránky apod. Často dochází k tomu, že se ve fórech ukládají „pavučiny“ a administrátoři je musí ručně odmazávat. Tento problém je možno řešit pomocí kontrolních číselných kódů nebo jednoduchých otázek, na které nemůžou webový roboti odpovídat algoritmickým vyhodnocením. Dalším podstatným problémem je ochrana uživatelských e-mailových adres. Pokud jsou adresy generovány jako běžné texty a ještě k tomu jako hypertextové odkazy, pak se stávají snadným cílem webových pavouků a jsou ukládány do databází pro zasílání webové reklamy. Některé systémy řeší tento problém generováním obrázků s e-mailovou adresou, kterou zobrazují místo klasického odkazu. Po kliknutí na takový obrázek se přes identifikátor zjistí e-mailová adresa z databáze a v postbacku je teprve vrácena návštěvníkovi webové prezentace.

1.2.2.6 Lokalizace

Tento požadavek vyplývá z obecného požadavku na jazykové přizpůsobení webových prezentací. Pokud je obsah dynamicky spravován, měl by CMS umožňovat zadávat obsahové elementy ve více jazycích.

1.2.2.7 Základní moduly pro CMS

Základním modulem každého CMS jsou „**články**“. Články většinou tvoří celý obsah webu. Články je možné řadit do skupin nebo struktur a tak je zobrazovat na různých stránkách webové prezentace. Často je požadované řazení do stromové hierarchické struktury, která odpovídá struktuře webového sídla. U článků je požadována možnost vzájemného propojení tak, aby si uživatel mohl zobrazit související články. Častým požadavkem je také udržování historie verzí jednotlivých článků. Ke článkům většinou také patří „diskuze“ nebo „hodnocení“.

„**Diskuze**“ je modul, který umožňuje napojení na různé jiné elementy CMS a umožňuje uživatelskou diskuzi k daným objektům. Zde je obvykle požadováno stromové zobrazení příspěvků – uživatelé mohou založit nový strom nebo přidat odpověď do existujícího stromu.

„**Správa souborů**“ nebo také „document management system – DMS“ by měl umožňovat vytvořit virtuální adresářovou strukturu s možností přidávat, přesouvat nebo odebírat složky, přidávat soubory pomoci „uploadu“ a umožňovat vystavení souborů ke stažení v libovolných částech webu. Existují dvě rozdílné implementace ukládání souborů na serveru. První možností je ukládat soubory do „zrcadlené“ struktury v souborovém systému webového serveru. Druhou možností je ukládat soubory přímo do databáze jako binární data. První možnost má výhodu v rychlosti (ukládání a načítání dat do/z DB je extrémně náročné), druhá možnost má velkou výhodu v koncentraci všech dat v jediném médiu – databázi, což je velmi důležitá vlastnost pro zálohování webového sídla.

„Správa uživatelů a udělování oprávnění“ je modul, který je obzvláště významný, pokud se jedná o webové sídlo typu „portál“. Nejčastěji jsou uživatelům přiřazeny role, které „hrají“ ve webovém sídle. Role pak uživateli udává šíři jeho oprávnění. Příkladem rolí může být „ADMINISTRATOR“, „ANONYM“, „NEWS EDITOR“ apod. Součástí tohoto modulu je také možnost registrace nových uživatelů (registrační průvodce), přihlašování uživatelů, restartování zapomenutých hesel apod.

Dalším důležitým modulem je „**galerie webové grafiky**“. Galerie má za úkol spravovat veškerý grafický obsah na webovém sídle a tento pak přiřazovat k různým elementům redakčního systému. Základní funkcionalitou je také vytváření skupin fotografií (např. fotografie z akcí apod.). Pro tento účel by měl CMS poskytovat možnost načtení celé složky fotografií najednou a možnost automatického vytvoření miniatur. Často může být realizována jako zvláštní položka DMS, protože grafické objekty jsou také soubory.

Mezi „**moduly pro sběr dat od návštěvníků**“ patří například hodnocení článků, ankety nebo „formuláře“. Formuláře jsou modul, který umožňuje administrátorovi navrhnout jednoduchý formulář, který může umístit na webovou stránku. Návštěvníci jej pak mohou vyplnit a odeslat na server. Administrátor má možnost sledovat data z odeslaných formulářů, nebo jsou data automaticky odesílána na mail pověřené osobě.

Dále většinou zákazníci požadují modul na „**správu produktů**“ nabízených danou společností. Pod tímto modulem se rozumí administrace skupin produktů, ceníků (import z jiných systémů), detailů produktů apod. Tento požadavek pak většinou vede k nutnosti vytvořit modul „**elektronický obchod**“. Elektronické obchody jsou však často realizovány jako samostatné aplikace, na které se základní firemní prezentace jen odkazuje. Může

však docházet k duplicitě dat mezi redakčním systémem a elektronickým obchodem. V takovém případě je nutno data mezi systémy replikovat.

1.2.3 Přehled existujících řešení na českém trhu

Redakční systémy jsou velmi populární a existuje velké množství hotových řešení. Tato řešení je možno hledat nejen mezi komerčními, ale také mezi volně šiřitelnými produkty. Nejčastěji nabízejí redakční systémy společnosti, které se zabývají tvorbou HTML prezentací. K prezentaci pak zákazníkovi přidají licenci na CMS. V kapitole 1.2.1 (Možnosti architektury spojení CMS a webového sídla) jsem popsal, jak je možno CMS integrovat do webové prezentace. Z uvedeného textu vyplývá, že se nejedná vždy o jednoduchou operaci. Proto jsou na trhu spíše výše popsaná řešení, která zákazníkovi nabídnou webovou prezentaci včetně CMS.

Tato kapitola představuje vybraná existující řešení CMS na českém trhu a pokouší se srovnat jejich výhody a nevýhody[1]. Výběr byl volen především tak, aby bylo představeno co nejvíce funkcionalit, které může CMS nabídnout. Informace získané z tohoto porovnání mají být výchozím inspiračním zdrojem pro funkcionality Getmore CMS realizovaného v rámci této bakalářské práce. U každého řešení uvádím zevrubný pohled na standardní moduly a také vyčleňuji moduly, které jsou z mého osobního pohledu něčím specifické nebo zajímavé.

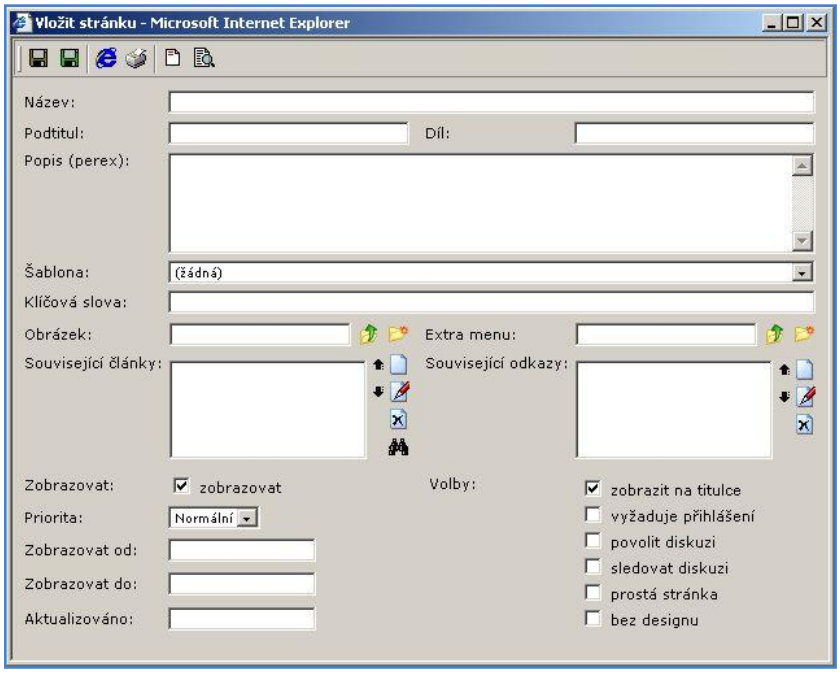
1.2.3.1 MultiCMS

Jedná se o plně komerční CMS systém s množstvím standardních modulů, který na první pohled poskytuje vše, co bychom od CMS požadovali. Tento systém patří mezi systémy, které je nutno zakoupit společně s webovou prezentací – nebo přesněji – webová prezentace musí být upravena výrobcem tak, aby mohla pracovat s CMS.

[1] Občas je nutno vycházet pouze z propagačních údajů uváděných výrobcem. U komerčních řešení nejsou dostupné zdrojové kódy a často ani demo aplikace.

Technologie:

MultiCMS je kompletně naprogramován v jazyce PHP 4.0. Jako datový sklad používá databázi MySQL nebo MS SQL. Výrobce také uvádí nezávislost na platformě webového serveru – tj. je možno použít Windows nebo Linux (a další OS založené na UNIXU). Pro plnou funkčnost je požadován internetový prohlížeč Internet Explorer 5.5 (doporučený je Internet Explorer 6.0) nebo prohlížeč Mozilla 1.4 a vyšší nebo jiné prohlížeče založené na jádře Gecko (Netscape Navigator od verze 6.0). Prohlížeč musí mít zapnut skriptovací jazyk JavaScript. Systém též využívá cookies, které však nejsou podmínkou pro funkčnost. Systém i implementace se drží standardu HTML 4.01. Bohužel výše uvedené požadavky na prohlížeče nejsou zvlášť specifikované pro administrativní a prohlížecí režim.



Obrázek 4 – Ukázka uživatelského rozhraní redakčního systému MultiCMS

Standardní moduly:

- **Publikační systém** – Každá stránka může mít zadáno období zobrazení a další vlastnosti. Díky šablonám, speciálním značkám a vkládání předpřipravených struktur lze publikovat stránky bez nutnosti skládání designu stránek.
- **Elektronický obchod** – Standardní elektronický obchod s možností definovat různé typy produktů a možností úpravy jejich parametrů.
- **Správa obrázků, galerie** – Možnost správy databáze obrázků s integrovaným editorem obrázků přes webové rozhraní.

- **Správa souborů** – Klasický dokumentový systém s možností zařazovat dokumenty do struktury, přesouvat je, publikovat ke stažení apod.
- **Diskuze a příspěvky** – obecný modul, který je možno navázat na jakoukoliv jinou položku v systému. Je možno tak přiřadit diskuzi ke článku apod. Navíc je možno nastavit podporu grafických „smajlíků", cenzuru vulgárních slov nebo cenzuru odkazů na konkurenční stránky apod.
- **Další moduly** - Systém dále obsahuje funkce pro export a import dat, zobrazení pro tisk, zobrazení pro WAP nebo PDA zařízení. Dále je k dispozici automatická detekce jazyka, zasílání mailů apod.

Výjimečné (neobvyklé) moduly:

- **Rotace položek** – Náhodné zobrazování položek uživateli. Systém dokáže hlídat, jaké položky z kolekce již uživatel viděl a nabízí mu pokaždé jiné.
- **Datamining** – Systém sbírá statistická data, která posléze mají vysokou míru predikce a velkou schopnost vypovídat o chování návštěvníka či zákazníka v internetovém obchodu.

Cena produktu, webové stránky výrobce:

Systém je nabízen v pěti různých edicích. Cena základní edice je 6 000,- Kč. Nejrozšířenější edice je k dispozici za 90 000,- Kč. Veškeré podrobnosti je možno získat na webových stránkách výrobce http://www.multicms.net. [16]

1.2.3.2 Vizus

Vizus je dalším příkladem komerčního CMS, který je k dispozici na českém trhu. Na webových stránkách výrobce je k dispozici plně funkční demo. Na první pohled mě administrace systému zaujala svou přehledností a přívětivým designem. WYSIWIG editor HTML obsahu je také velmi zdařilý a umožňuje velké množství pokročilých funkcí.

Technologie:

Vizus je naprogramován v PHP a jako datový sklad používá databázi MySQL. Pro administrativní část je vyžadován prohlížeč alespoň Internet Explorer 5.5 se zapnutou podporou technologie ActiveX. Pro veřejné webové části je možno použít libovolný pro-

hlížeč. Výsledné generované HTML je dle údajů výrobce plně validní nebo odpovídá standardu XHTML.

Standardní moduly:

- **Stránky** - Správa struktury webových stránek a jejich obsahu s možností náhledu před zveřejněním.
- **Novinky** – Systém rozlišuje několik typů novinek, archiv starých novinek…
- **Články** - Rozdělení do tematických rubrik, anotace, obrázky a přílohy, řízené schvalování a zveřejnění…
- **Fotogalerie** - Rozdělení do tematických kategorií, hromadný import…
- **Odkazy** – Tematicky rozdělené odkazy do internetu, včetně popisu nebo náhledu cílové stránky…
- **Dokumenty** – Tematicky roztříděné dokumenty, popis obsahu, fulltextové vyhledávání v obsahu…
- **Ankety** – Řízené zveřejňování anket podle data od – do, grafické vyhodnocení…
- **Katalog produktů** – Elektronická verze katalogu, rozdělení do kategorií, náhledy, možnost objednání…
- **Diskuse** – jednoduchá diskuse pod článkem nebo velké diskusní fórum, možnost schvalování příspěvků…
- **Správa uživatelů a oprávnění** – Systém umožňuje vytvářet uživatele a zařazovat je do skupin. Dále je možno specifikovat práva pro zobrazení a editaci. Práva jsou však aplikována jen v rámci administrace. Výsledná webová prezentace nemá možnost registrace nebo přihlášení uživatelů.
- **Další moduly** –Zasílání novinek na e-mailové schránky, systém pro správu bannerů, statistika stránek, mapa webu, RSS apod.

Výjimečné moduly:

- **Kalendář akcí** – Přehled budoucích akcí na časové ose s možností filtrování nebo rozdělováním do kategorií.
- **Formuláře** – Tvorba vlastních formulářů pro sběr informací od návštěvníků.

Cena produktu, webové stránky výrobce, demo aplikace:

Výrobce uvádí cenu od 15 000,- Kč do 30 000,- Kč bez vytvoření grafiky. Veškeré podrobnosti je možno získat na webových stránkách výrobce http://www.vizus.cz. Veřejná demo aplikace se nachází na http://cms.vizus.cz. [17]

1.2.3.3 Drupal

Poslední systém, na který se podíváme podrobněji, je Drupal. Drupal je „Open-Source" šířený pod licencí GNU/GPL. Je tedy možno jej volně upravovat a šířit. Drupal je dílem holandského studenta Driese Buytaerta a vznikl v roce 2000. Původně sloužil jako systém na sdílení informací mezi studenty koleje. O vývoj Drupalu se stará několik hlavních vývojářů a více než 400 přispěvovatelů, kteří poskytli své patche do jádra. Největší předností Drupalu jsou ale vývojáři modulů - těch jsou stovky a vytvářené moduly jsou volně ke stažení.I přes svou jednoduchost má Drupal velmi mnoho významných referencí. Například se jedná o oficiální stránky britské hudební stanice MTV (http://www.mtv.co.uk), stránky českého časopisu Computer (http://computer.zive.cz) nebo stránky o vývoji Linuxového jádra GNU (http://kerneltrap.org).

Technologie:

I tento redakční systém je naprogramovaný v jazyce PHP. Od verze 4.7.5 podporuje PHP 5.2.0. Jako datový sklad používá databáze MySQL nebo PostgreSQL. Na straně serveru je požadován webový server Apache nebo Microsoft IIS. Na rozdíl od předchozích redakčních systémů nabízí tento systém i registraci a přihlašování uživatelů. Projekty postavené na tomto řešení jsou spíše dynamické webové portály sloužící pro výměnu informací mezi členy dané komunity. Proto se také výrazně liší od předchozích dvou produktů.

Vybrané moduly z demo aplikací:

- **Články –** Poměrně nedokonalý modul. Umožňuje přidávat články do jednoúrovňové struktury kategorií. K editaci článku není možno využít žádný WYSIWYG editor.
- **Fórum** – Klasické fórum se stromovou strukturou témat. Je možno vkládat příspěvky do fóra nebo je editovat. Velmi mě zaujala možnost zadat při vytváření příspěvku vlastní URL, pod kterým bude článek mapován. Tato funkcionalita pak může mít zásadní vliv pro SEO.

- **Fotogalerie** – Modul umožňující vkládat nové obrázky a zařazovat je do skupin.
- **MessageBox** – Součást modulu pro správu uživatelů. Umožňuje zasílání zpráv mezi uživateli jednoho portálu. Po přihlášení vidí uživatel seznam nových zpráv.
- **Personal blog** – Možnost vedení dnes populárních blogů. Modul opět nedisponuje zvláštními funkcionalitami.

Cena produktu, webové stránky výrobce, demo aplikace:

Jak již bylo řečeno, produkt je zdarma pod licencí GNU/GPL. Oficiální stránky produktu jsou http://drupal.org. Česká verze stránek je na adrese http://drupal.org. [13]

1.2.3.4 Srovnání systémů

Následující tabulka obsahuje sumární pohled na výše popsané systémy. Některé hodnoty ke srovnání nebyly k dispozici. V takových případech uvádím „neuvedeno“. Dále jsem redakční systémy podrobil krátkému testu na validitu generovaného HTML kódu. K tomuto účelu jsem použil validátor HTML kódu na adrese http://validator.w3.org. Validátor vyhledá v dokumentu specifikaci HTML (popř. XHTML nebo jinou), kterou se stránka snaží dodržovat (podle hlavičky DOCTYPE). Poté provede kontrolu HTML kódu přesně podle specifikací pro danou technologii. Výsledkem je sada chyb, upozornění a informací. U každého CMS jsem náhodně vybral 3 významné reference, které jsou generovány pomocí daného redakčního systému, a podrobil je testu validace. V tabulce uvádím počet nalezených chyb v HTML kódu.

Tabulka 1 – Porovnání CMS

Sledovaný údaj	MultiCMS	Vizus	Drupal
Cena	6000 Kč - 90000 Kč	15000 Kč – 30000 Kč	Zdarma
Programovací jazyk	PHP 4.0	PHP	PHP 5.0
Datový sklad	MySQL, MS SQL	MySQL	MySQL, PostgresSQL
WYSIWYG editor	ANO	ANO	NE
Dostupné demo	NE	ANO	ANO
Splňovaná norma HTML	HTML 4.01	validní HTML, XHTML	neuvedeno
Validace náhodné reference 1	21 chyb	OK	30 chyb
Validace náhodné reference 2	7 chyb	23 chyb	2 chyby
Validace náhodné reference 3	20 chyb	3 chyby	2 chyby
Požadavky na prohlížeč	IE 5.5	neomezeno	neuvedeno
Požadavky na prohlížeč - administrace	IE 5.5	IE 5.5	neuvedeno
Požadavky na server	Windows, Linux, Unix	Unix, Apache	Apache, IIS
Webové stránky	http://www.multicms.net	http://www.vizus.cz	http://drupal.org

II. PRAKTICKÁ ČÁST

2.1 Zadání od společnosti Getmore

Společnost Getmore v současné době nedisponuje nástrojem pro správu a aktualizace své webové prezentace. Cílem je vytvořit systém, který bude sloužit primárně pro potřeby Getmore, sekundárně pro potřeby dalších zákazníků společnosti Getmore. Je potřeba navrhnout základní architekturu systému a implementovat základní moduly. Navržená architektura musí být dostatečně robustní, aby bylo možno produkt dále rozšiřovat.

2.1.1 Požadované použité technologie

Cílem je dosáhnout maximálního využití technologií Microsoft ASP.NET 2.0 na všech úrovních a přistoupení na všechny postupy vývoje webových aplikací dle Microsoft. Výsledné řešení by pak mělo obsahovat minimum částí třetích stran a minimum vlastních implementací, které by překrývaly existující technologie a postupy definované v rámci ASP.NET 2.0. Jako datový sklad bude sloužit Microsoft SQL server 2005 nebo Microsoft SQL server 2000.

2.1.2 Požadovaný princip fungování s webovou prezentací

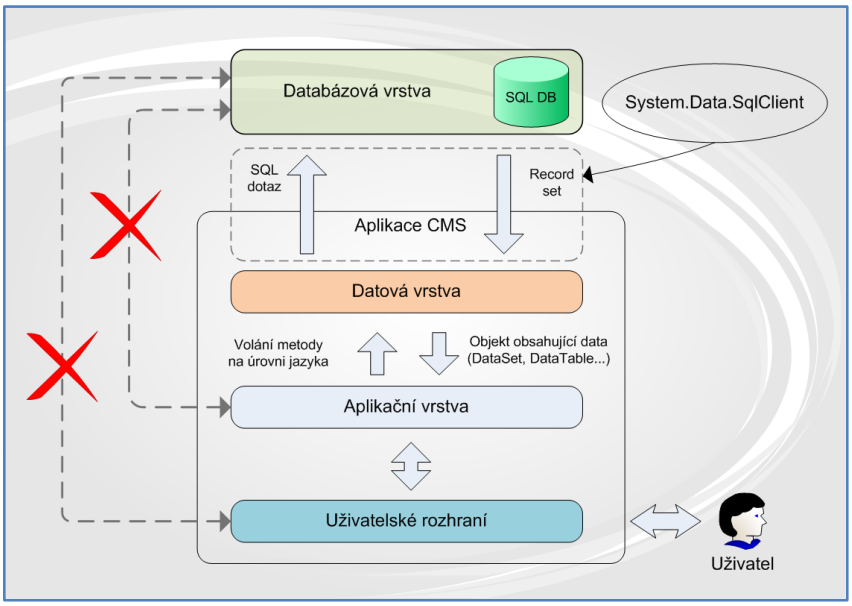
CMS bude součástí webové prezentace, ale jen na úrovni uživatelského rozhraní. To znamená, že administrace bude probíhat přes uživatelské rozhraní webové prezentace. Po ověření uživatele dojde k zobrazení editačních prvků a k umožnění administrace webové prezentace (samozřejmě pouze na základě user role management a přidělených oprávnění).

Na straně kódu by však měl být CMS striktně oddělen od webové prezentace. Vždy by mělo platit, že CMS je možno kdykoliv odebrat nebo jej nahradit novou verzí.

Verze CMS je dána programovým kódem CMS a rozhraním (interface) s webovou prezentací. Webová prezentace musí splňovat požadavky na interface, aby ji bylo možno provozovat s CMS dané verze. Interface tvoří propojení mezi CMS a webovou prezentací. Interface bude realizováno spíše abstraktně a jeho platnost bude hlídat překladač Visual Studia 2005 (2.4.1 Realizace rozhraní).

2.1.3 Požadovaná architektura aplikace CMS

Při vývoji CMS je nutno dodržet přísné členění aplikace na vrstvy. Především je vyžadováno úplné oddělení datové vrstvy od aplikační vrstvy. Datová vrstva bude separována do zvláštního projektu. Následující diagram ukazuje požadovanou strukturu aplikace CMS.



Obrázek 5 – Schéma vrstvené architektury CMS

Diagram také ukazuje, že pro komunikaci datové vrstvy s databází bude využito objektů z oboru názvů `System.Data.SqlClient`. Veškeré datové operace musí být realizované na databázové úrovni formou „stored“ procedur – tj. veškeré objekty typu `SqlCommand` musí mít vlastnosti CommandType jako „StoredProcedure“. V kódu datové vrstvy se nesmí vyskytovat přímé textové SQL příkazy.

2.1.4 Požadované moduly

V rámci této bakalářské práce je nutno především navrhnout kompletní architekturu, která bude dále rozšiřitelná modulárním způsobem. V rámci práce by měly být realizovány následující moduly.

2.1.4.1 Vrstva oprávnění

Základní požadavek je na vytvoření jádra CMS – tj. správa uživatelů a uživatelských rolí. Tato funkcionalita bude plně převzata z ASP.NET a bude rozšířena jen o možnost přidělovat oprávnění uživatelům a rolím pomocí aplikace. K tomuto účelu bude později slou-

žit systémový modul „**Oprávnění**“. Logickou součástí tohoto modulu bude vrstva v aplikaci CMS, která bude sloužit pro ověřování oprávnění. Jednotlivé moduly CMS budou moci k této vrstvě přistupovat, a tak budou umožňovat zobrazení podle aktuálního nastavení oprávnění. Jednotlivé moduly by vždy měly k udělování oprávnění používat tuto vrstvu a neměly by svévolně zasahovat do objektů v oborech názvů `System.Web.Security.Membership` a `System.Web.Security.Roles`.

Modul pro udělování oprávnění musí umožnit přidávat nová možná oprávnění a tato oprávnění přiřazovat k roli nebo k uživateli. Každé nově přidávané oprávnění se bude vázat na konkrétní modul a na specifickou akci v modulu. Příkladem oprávnění je například „NEWS_VIEW“ – takové oprávnění znamená právo vidět modul novinky. Oprávnění pak může být přiřazeno například k uživateli ANONYM nebo roli ANONYMOUS USERS. Moduly pro přihlašování nebo registraci nových uživatelů budou plně převzaty z technologie ASP.NET a nebudou nijak obaleny v CMS. Bude tak ponechána naprostá volnost při realizaci těchto komponent při implementaci CMS do webové prezentace.

2.1.4.2 Články

Tento modul by měl odpovídat popisu v kapitole 1.2.2.7 (Základní moduly pro CMS). Modul bude realizován jako serverový ovládací prvek a bude umožňovat uživatelské nastavení vzhledu pomocí šablon (template control). Modul dále umožní zobrazovat seznam článků s možností stránkování, zobrazovat detail vybraného článku, přidávat nové články nebo editovat a mazat stávající články. Ke každému článku bude udržována historie revizí (verzí). Články budou mít mezi sebou vazby, aby mohly být ke každému článku zobrazeny odkazy na související články. Modul články bude obecný a bude se moci v prezentaci vyskytovat vícekrát. Každý modul typu „články“ bude mít svůj řetězcový kód – tzv. ModuleCode. Modul bude zobrazovat jen články přiřazené k danému modulu (napojení bude realizováno pomocí ModuleCode). Články uvnitř modulu nebudou dále nijak segmentovány do struktury apod. Základní oprávnění pro tento modul bude právo „VIEW“ – vidět modul a právo „EDIT“ – upravovat data v modulu.

2.1.4.3 Document management system

Tento modul se bude odlišovat od standardního pojetí modulů pro správu souborů. Bude fungovat jako FRONT-END aplikace pro Microsoft SharePoint portal. Nejjedno-

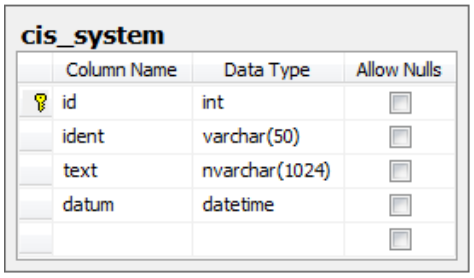
dušším pojetím bude umisťování odkazů na soubory v systému SharePoint. Bylo by vhodné vysledovat, jaké další možnosti nabízí Microsoft SharePoint pro komunikaci s jinými aplikacemi a popřípadě tyto možnosti využít. Požadavek na napojení na SharePoint je podstatný z důvodu již existující databáze souborů společnosti Getmore, která využívá právě tento portál. Později bude možno doplnit DMS o standardní funkcionalitu, která bude umožňovat fungování modulu i bez napojení na SharePoint.

2.2 Databázová struktura

Tato kapitola popisuje strukturu databáze, kterou bude využívat aplikace CMS. Dle požadavků společnosti Getmore je databáze realizována na platformě Microsoft SQL server 2005. Názvové konvence SQL objektů vychází z obecných programátorských postupů společnosti Getmore.

2.2.1 Základní systémová struktura

Základní systémová struktura je tvořena jedinou číselníkovou tabulkou **cis_system**. Tato tabulka udržuje různá systémová nastavení metodou klíč – hodnota. Tato struktura je implementována do všech databází společnosti Getmore. Jejím účelem je mimo jiné udržovat verzi a historii verzí databáze. Tabulka je nezbytná také pro využití utilit společnosti Getmore na provádění upgrade struktury databáze. Na následujícím obrázku je struktura tabulky cis_system.

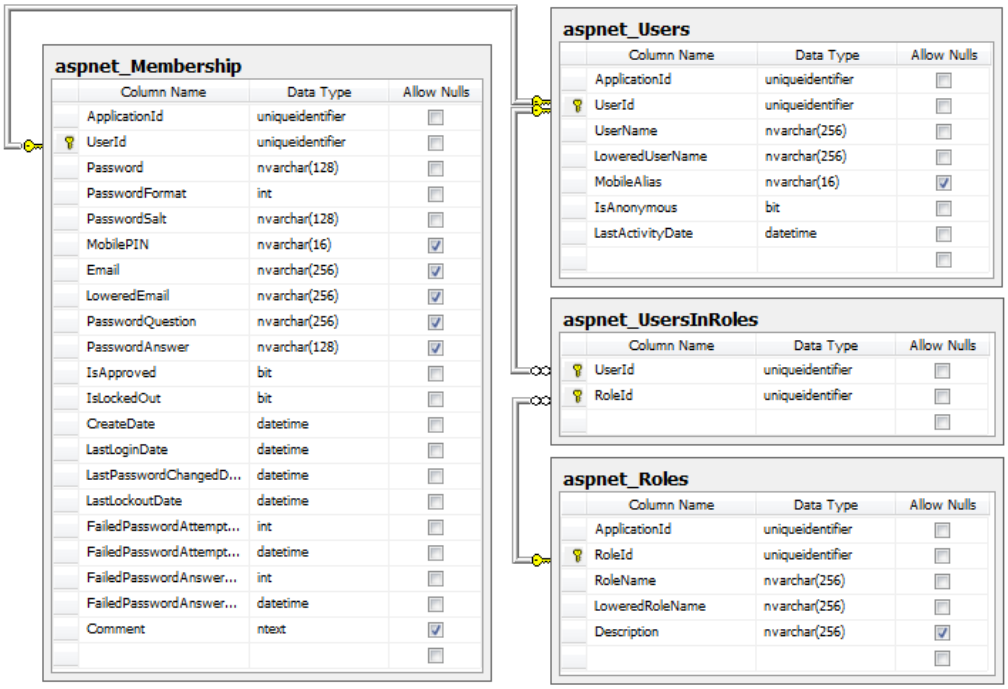


Obrázek 6 – Struktura tabulky cis_system

Ze struktury tabulky (Obrázek 6) je zřejmé, že obsahuje typově nezabezpečená data. Podle klíče „ident“ je možné získat hodnotu „text“, která je typu NVARCHAR(1024). Pro manipulaci s tabulkou cis_system je proto vytvořena množina CRUD procedur a SQL funkcí, které umožňují pracovat s tabulkou typově bezpečně.

2.2.2 ASP.NET struktura pro membership a roles management

Aby mohla aplikace CMS používat standardní službu pro ověřování uživatelů a správu rolí, je nutno do databáze implementovat struktury pro uložení potřebných informací. Tato struktura je pevně dána společnosti Microsoft a skládá se z poměrně velkého množství tabulek a stored procedur. V zásadě existují dvě možnosti implementace. První z nich je umístění těchto struktur do zcela samostatné databáze. Struktury totiž umožňují spravovat v jedné databázi data pro více webových aplikací. Na tuto databázi se pak může odkazovat „datová“ databáze dané aplikace přes specifické application ID. Druhou možností je nainstalovat podporu pro tyto služby přímo do konkrétní databáze aplikace. Já jsem zvolil druhou možnost, protože budu služby rozšiřovat o implementace pokročilejšího oprávnění, které vyžadují odkazy na objekty typu uživatel nebo role. Pokud by se data nacházela v jiné databázi, nebylo by možno zajistit dokonalou integritu dat bez nutnosti realizace replikačních procedur. Na následujícím obrázku (Obrázek 7 – Databázová struktura pro membership a user roles) je výřez struktury, která slouží pro uložení informací o uživatelích a rolích. Tabulka aspnet_users uchovává základní informace o uživatelích. Detail záznamů z této tabulky je uložen v tabulce aspnet_membership. Tabulka aspnet_roles obsahuje seznam rolí definovaných v rámci aplikace. Tabulka aspnet_usersInRoles zařazuje uživatele do jednotlivých rolí. Vazby jsou patrné z obrázku.



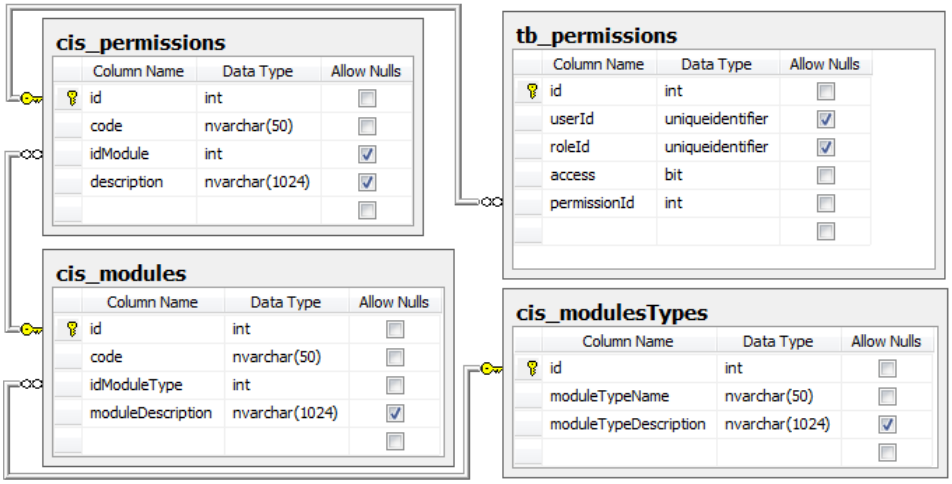
Obrázek 7 – Databázová struktura pro membership a user roles

2.2.3 Struktura pro oprávnění

2.2.3.1 Struktura tabulek pro oprávnění

Tato struktura rozšiřuje základní funkcionalitu správy uživatelů a rolí. Základní struktura totiž neumožňuje stanovit přístupová práva uživatelům a rolím dynamicky. Přístup pro jednotlivé role by tak musel být napevno zadán v kódu modulů CMS. Pokud by chtěl administrátor webové prezentace přidat některé roli více oprávnění, znamenalo by to vytvořit novou verzi CMS s jiným rozložením oprávnění pro dané role. Dalším řešením, které by nevyžadovalo rozšíření existujících struktur, by bylo vytvoření velkého množství dílčích rolí jako například „ARTICLES MANAGER“, „ARTICLES VIEWER“, „DMS MANAGER“ apod. Každý modul by pak nabízel sadu několika rolí, které by bylo možno přiřadit nebo odebrat uživatelům. Problém by však nastal v případě, že daný modul může mít v systému více instancí. Typickým příkladem je modul články. Nebylo by pak možné odděleně stanovit oprávnění pro jednotlivé instance modulu články. Z výše uvedených důvodů jsem se rozhodl strukturu databáze rozšířit tak, aby umožňovala uchovávat informace o přidělených oprávnění k daným rolím a uživatelům.

Struktura je zobrazena na následujícím obrázku (Obrázek 8 – Databázová struktura pro oprávnění).



Obrázek 8 – Databázová struktura pro oprávnění

Hlavní datovou tabulkou je tabulka **tb_permissions**. Slouží pro uchování oprávnění pro daného uživatele nebo roli. Nad tabulkou je zaveden CHECK CONSTRAINT pro zaručení, že je naplněn vždy právě jeden ze sloupců userId a roleId. Sloupce userId a roleId od-

kazují do tabulek aspnet_users a aspnet_roles. Sloupec access udává, zdali je dané oprávnění uděleno nebo odebráno. Sloupec permissionId ukazuje do číselníku možných oprávnění, které je možno přidělit nebo odebrat. Tímto číselníkem je tabulka **cis_permissions**. Tabulka udává kód (code) oprávnění (mělo by se jednat o jednoznačný identifikátor pro danou hodnotu ve sloupci idModule, což je zajištěno pomocí indexu na těchto sloupcích idModule a Code). Sloupec idModule indikuje, k jakému modulu dané oprávnění patří, což také udává logickou skupinu oprávnění. Seznam instancí modulů obsažených v dané webové prezentaci je uložený v tabulce **cis_modules**. Každá instance je definována kódem, který je jedinečný v rámci celé tabulky (to je zajištěno indexem na tomto sloupci). Tento kód také slouží pro napojení daného modulu v systému na databázovou strukturu. Každému modulu je přiřazen základní typ přes sloupec idModuleType, který odkazuje do tabulky **cis_modulesTypes**. Základním typem můžou být například „články“. Instancí může být například modul s kódem NEWS. Obsah tabulky cis_modules je tedy specifický pro danou webovou prezentaci využívající CMS, naopak obsah tabulky cis_modulesTypes je specifický pro danou verzi CMS.

2.2.3.2 Stored procedury pro oprávnění

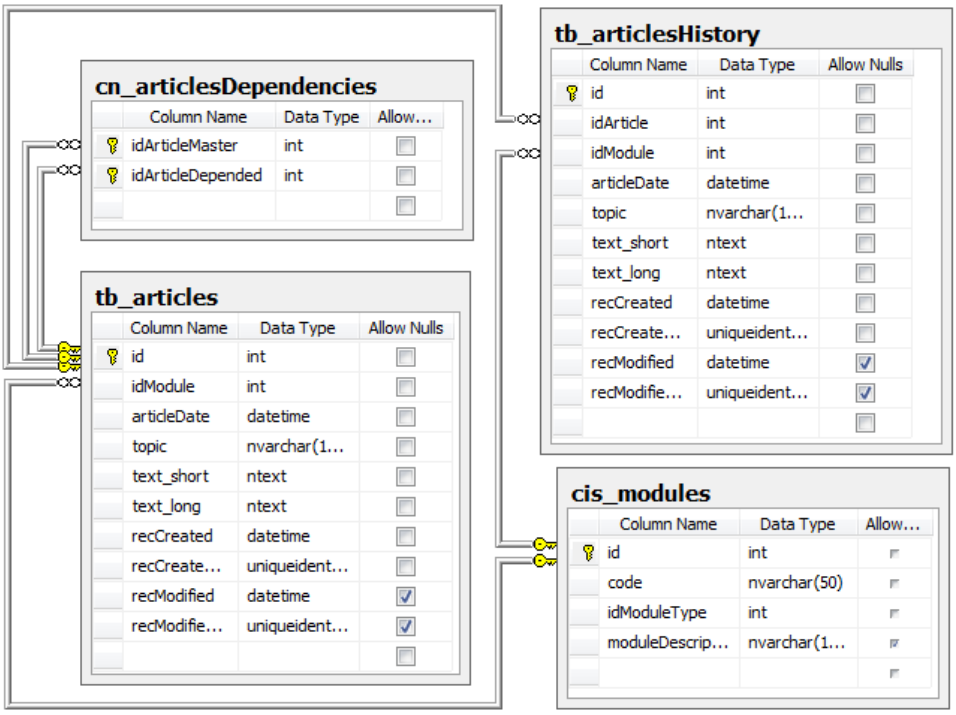
Pro manipulaci s oprávněními je nutno definovat procedury pro klasické CRUD operace. Pojmenování procedur opět vychází z názvových konvencí společnosti Getmore.

- **cms_CRUD_tb_permissions_RET –** Procedura slouží pro získání nastavení všech oprávnění v systému.
- **cms_CRUD_tb_permissions_CREATE_UPD –** Procedura slouží pro uložení nového nastavení oprávnění nebo pro úpravu stávajícího nastavení oprávnění. Úpravou je myšlena především změna sloupce access. Parametry odpovídají datovým položkám tabulky tb_permissions. Pro změnu nastavení oprávnění je nutno zadat nepovinný parametr @PermissionId.
- **cms_CRUD_tb_permissions_DEL –** Procedura slouží pro odebrání existujícího nastavení oprávnění. Parametrem je ID nastavení oprávnění z tabulky tb_permissions, které chceme odebrat.

2.2.4 Struktura pro články

2.2.4.1 Struktura tabulek

Tato struktura udržuje informace o článcích a jejich historii. Struktura je zatím v základní podobě a po implementaci dalších modulů bude rozšířena. Jedním z požadavků bylo také uchovávání historie verzí jednotlivých článků a také ukládání závislostí mezi články. Struktura pro uložení článků je zachycena na následujícím obrázku.



Obrázek 9 – Databázová struktura pro články

Základní tabulkou pro uložení článků je tabulka **tb_articles**. Tato tabulka obsahuje nejnovější verze všech článků v systému. Pomocí sloupce idModule jsou články připojeny ke konkrétnímu modulu. Zde je mírný problém, protože články je takto možné napojit i na jiné instance než na instance typu modulu „články“. V důsledku se nejedná o problém, který by narušoval integritu dat, ale jedná se potom o tzv. mrtvé záznamy. Bylo by vhodné ošetřit tento stav také patřičným triggerem nad tabulkou tb_articles, který by neumožnil v tabulce udělat takovou změnu, která by vedla na propojení na instanci jiného typu než typu „články“. Pak by bylo nutné také ošetřovat triggerem tabulku cis_modules, který by neumožnil změnu typu modulu na řádku, na který jsou napojeny články.

Tabulka **tb_articlesHistory** udržuje historii jednotlivých článků. Každý záznam z této tabulky se pomocí slupce idArticle váže na aktuální verzi článku. Pomocí CRUD procedur pro články a triggerů je zajištěno, že při každé změně v tabulce tb_articles je uložena nová verze do tabulky tb_articlesHistory.

Tabulka **cn_articlesDependencies** slouží jako spojovací tabulka a zajišťuje závislosti mezi články. Jednomu článku může příslušet libovolný počet závislých článků. Sloupec idArticleMaster určuje konkrétní článek a sloupec idArticleDepended určuje závislý článek.

2.2.4.2 Stored procedury a triggery pro články

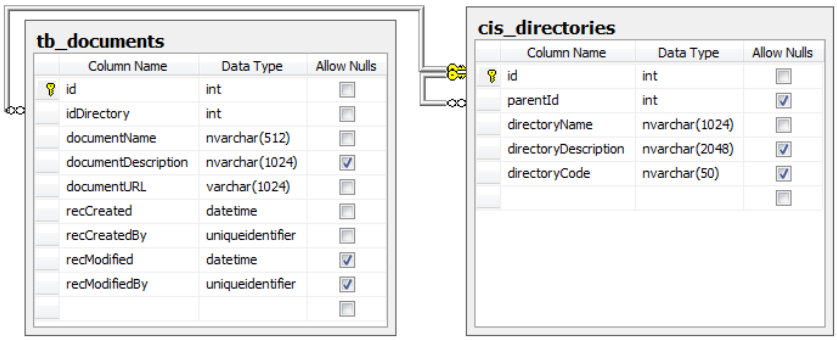
Pro články jsou nutné klasické CRUD procedury nad tabulkou tb_articles. Dále je nutný trigger, který zajišťuje plnění tabulky tb_articlesHistory s historií verzí článků.

- **cms_CRUD_tb_articles_RET –** Procedura pro získávání článků z tabulky tb_articles. Parametrem je kód modulu, ke kterému chceme dočíst články.
- **cms_CRUD_tb_articles_CREATE_UPDATE –** Procedura pro vytvoření nového nebo změnu existujícího článku. Parametry odpovídají sloupcům v tabulce tb_articles. Pro změnu článku je nutno naplnit nepovinný parametr @ArticleId.
- **cms_CRUD_tb_articles_DEL –** Procedura pro úplně odstranění článku ze systému. Současně s článkem dojde ke smazání všech záznamů, které se na článek vážou cizím klíčem. Dojde tak i ke smazání historie verzí článku a informací o vazbách na jiné články.
- **cms_TR_tb_articles_IU –** Trigger nad tabulkou tb_articles, který se vykonává pro akce INSERT a UPDATE. Jeho úkolem je kopírovat vkládaná nebo měněná data do tabulky tb_articlesHistory. Záznám o historii se vytváří i v případě vložení nového článku.

2.2.5 Struktura pro DMS – dokumentový systém

Struktura pro DMS je také jen v základní verzi. Navržená struktura slouží pouze pro uložení dokumentů v databázi a pro jejich administraci. Později bude nutno strukturu rozšířit o další tabulky, které budou udržovat přehled o napojení dokumentů na jiné objekty

v systému. Konkrétně se bude jednat například o přidávání dokumentů ke stažení k jednotlivým článkům.



Obrázek 10 – Struktura pro dokumentový systém

Tabulka **tb_documents** ukládá informace o jednotlivých souborech zavedených do systému. Jednotlivé vlastnosti jsou zřejmé z názvů sloupců. Vazba idDirectory ukazuje na adresář, ve kterém je dokument virtuálně zařazen. Struktura adresářů je dána tabulkou **cis_directories**. Jedná se o stromovou tabulku s vazbou na sebe sama. Sloupec parentId udává nadřazenou úroveň daného záznamu. Pokud sloupec parentId nabývá hodnoty NULL, pak se jedná o kořenový adresář stromu. Z návrhu je zřejmé, že struktura neobsahuje binární data souborů. Soubory jsou ve struktuře uloženy jen virtuálně a jedná se v podstatě o odkazy na dokumenty umístěné kdekoliv v Internetu.

2.2.5.1 Stored procedury a triggery pro DMS

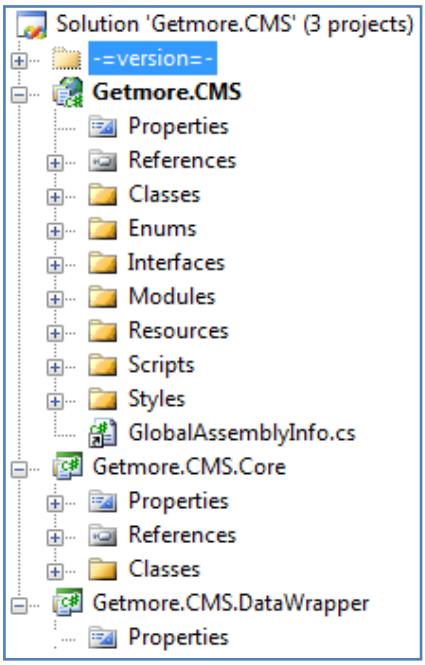
Pro DMS je nutno vytvořit dvě sady CRUD procedur. Jedna sada slouží pro manipulaci se stromovou strukturou adresářů. Jsou požadovány základní akce – načítání struktury, přidávání adresářů, editace adresářů a mazání adresářů (přesouvání adresářů je realizování úpravou sloupce parentId). Druhá sada slouží pro obdobné operace nad tabulkou s dokumenty.

- **cms_CRUD_cis_directories_RET** - Procedura slouží pro získávání stromové struktury. Na základě získaných dat je třeba sestavit vhodný datový zdroj pro serverový ovládací prvek `Treeview`, který umožní zobrazení struktury.
- **cms_CRUD_cis_directories_CREATE_UPD –** Procedura pro přidání nebo změnu jedné položky stromové struktury. Pro změnu je nutno naplnit nepovinný parametr @DirectoryId.

- **cms_CRUD_cis_directories_DEL –** Procedura pro smazání jedné položky stromové struktury. Bude obsahovat volitelné vstupní parametry, které umožní mazat i neprázdné složky struktury nebo složky, které obsahují dokumenty. V takovém případě je nutno smazat i veškeré podřízené složky resp. dokumenty.
- **cms_CRUD_tb_documents_RET –** Procedura umožní získat seznam všech dokumentů v tabulce nebo seznam dokumentů z určitého adresáře. To bude zajišťovat nepovinný parametr @DirectoryId.
- **cms_CDRUD_tb_documents_CREATE_UPD –** Procedura, která zajistí vložení nového nebo úpravu existujícího dokumentu. Procedura také umožní přesouvat dokumenty mezi složkami, protože tato operace obnáší pouze změnu hodnoty idDirectory v tabulce tb_documents.
- **cms_CRUD_tb_documents_DEL –** Procedura pro smazání dokumentu ze systému.

2.3 Solution Getmore.CMS

Celý systém bude realizován v rámci jednoho solution Visual Studia 2005, které bude obsahovat tři projekty (viz níže). Realizace v rámci více projektů má zajistit oddělení základních vrstev aplikace. Rozvrstvení aplikace bylo popsáno již výše v textu (kapitola 2.1.3). Následující kapitoly stručně charakterizují význam jednotlivých projektů v solution.



Obrázek 11 – Struktura projektů a složek v solution Getmore.CMS

Všechny projekty v solution jsou vnitřně členěny do logických celků tak, aby se podobné funkcionality nacházely na „podobném“ místě. Zejména jsou odděleny zdrojové kódy implementující třídy od kódu s rozhraními (interface) a od kódu s enumerátory. Toto logické rozčlenění je realizováno složkami „Classes“ (pro třídy), „Interfaces“ (pro rozhraní) a „Enums“ pro enumerátory. Každá z těchto složek je vnitřně členěna dle označení oborů názvů (namespaces), do kterých dané zdrojové kódy náleží. Dalším logickým celkem je složka „Modules“, která obsahuje public třídy veřejných modulů. Tyto moduly tvoří veřejné rozhraní knihovny s webovou aplikací, a proto jsou odděleny zvlášť. Logicky by mělo platit, že jen třídy v této složce jsou označeny jako public. Ostatní třídy v projektu by měly mít označení internal.

2.3.1 Projekt Getmore.CMS

Jedná se o základní projekt celého systému. Výstupem tohoto projektu je zejména DLL knihovna, která obsahuje veškeré moduly CMS, které je možno rozmisťovat do webové prezentace. Projekt dále obsahuje některé pomocné ovládací prvky, které jsou využity ve veřejných modulech. Tyto ovládací prvky jsou však skryty (jsou soukromé – private), protože jejich funkcionalita je specifická a jejich samostatné využití ve webové aplikaci je bezvýznamné. Projekt také obsahuje zdroje (resources) jako například lokalizační zdroje, obrázky, šablony kaskádových stylů, soubory s javascripty apod. Resources jsou také výstupem tohoto projektu. Po zkompilování se tedy ve výstupním adresáři kompilace (složka bin) objeví jednak knihovna Getmore.CMS.DLL a jednak složky obsahující obrázky, javascripty, css soubory apod.[1]

2.3.2 Projekt Getmore.CMS.DataWrapper

Tento projekt je realizací datové vrstvy. Jeho základním významem je poskytovat celému systému přístup do datových zdrojů. Definuje objekty pro přístup k datům (označované v projektu jako DataLayers) a dále objekty pro práci s daty. Objekty pro práci s daty většinou dědí některý existující datový zdroj (nejčastěji `SqlDataSource`) a vnitřně

[1] Pokud by se nejednalo o webovou aplikaci, bylo by možno umístit speciální zdroje (jako obrázky) přímo do DLL knihovny jako tzv. „embedded resources“. V tomto případě je však nutno vyčlenit obrázky do zvláštních souborů, protože webová aplikace pracuje s URL obrázků a nikoliv s binárními daty.

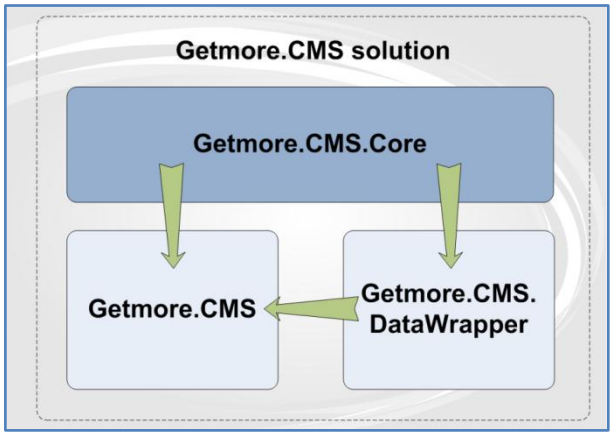
jej rozšiřují o implementace specifické pro účel datového zdroje. Takto je například vytvořen datový zdroj pro modul články (`ArticlesDataSource`). Tento zdroj vnitřně definuje SQL dotazy pro CRUD operace nad tabulkou tb_articles. Pomocí specifické datové vrstvy (`ArticlesDataLayer`) je tento objekt poskytován do modulů CMS.

Tento projekt produkuje jedinou DLL knihovnu. Tuto knihovnu pak konzumuje projekt Getmore.CMS jako referenci v rámci solution. Obecně platí, že třídy z projektu Getmore.CMS nesmí přímo pracovat s objekty oboru názvu `System.Data` a `System.Data.SqlClient`. Vždy by měly pracovat s objekty poskytovanými datovou vrstvou. Tohoto záměru jsem chtěl docílit odebráním referencí na sestavení `System.Data` a `System.Data.SqlClient` z projektu Getmore.CMS a ponecháním těchto referencí jen v projektu Getmore.CMS.DataWrapper. Toto odebrání však není možné, protože tímto by projekt Getmore.CMS ztratil informace o implementacích základních objektů datových elementů (`DataRow`, `DataView`…), což by znemožnilo použití např. zdroje `ArticlesDataSource`, který dědí `SqlDataSource`, protože ten využívá tyto základní elementy. Pokud bychom z Getmore.CMS odebrali reference na standardní datová sestavení .NET a ponechali jen referenci na Getmore.CMS.DataWrapper, zahlásí překladač chybu a vyžaduje připojení daných sestavení.

Tento projekt má výhradní napojení na databázovou vrstvu, která je popisována výše v textu (2.2 Databázová struktura). Projekt je také specifický v tom, že neobsahuje jediný klasický SQL příkaz. Veškerá komunikace je realizována přes SQL stored procedury.

2.3.3 Projekt Getmore.CMS.Core

Tento projekt představuje pomyslné jádro celého solution Getmore.CMS. Jádro stojí na nejvyšší vrstvě solution a obsahuje obecné definice, které mohou být využity ve všech projektech solution. To znamená, že všechny projekty solution obsahují referenci na tento projekt. Následující diagram ukazuje rozložení projektů v solution Getmore.CMS.



Obrázek 12 – Struktura solution Getmore.CMS

2.4 Způsob napojení na WWW prezentaci

Kapitola popisuje postup při napojení výstupu z projektu Getmore.CMS do libovolné ASP.NET 2.0 aplikace.

2.4.1 Realizace rozhraní

V požadavcích byla realizace logického rozhraní mezi webovou aplikací a mezi CMS. Toto rozhraní je vyřešeno velmi jednoduše a přitom velmi dokonale a zajistí většinu požadovaných kontrol. Tím, že se podařilo veškeré programové části vměstnat do DLL knihovny, je veškerá kontrola kompatibility aplikace a CMS zajištěna prostým přeložením webové aplikace. V případě, že došlo ke změně implementace v CMS, která znemožní jeho použití v aplikaci používající předchozí verzi CMS, nebude možné webovou aplikaci přeložit a publikovat, dokud v ní neproběhnou potřebné změny tak, aby byla schopna pracovat s novou verzí CMS.

Co se týče konkrétní realizace napojení CMS do webové aplikace, je nutno vykonat následující kroky. V projektu webové aplikace je nutno vytvořit složku CMS, do které se nakopíruje daná verze CMS. Verze CMS je dána obsahem složky BIN, kterou vygeneruje projekt Getmore.CMS ze solution Getmore.CMS. Dále je nutno projektu s webovou aplikací přidat souborovou referenci na knihovnu Getmore.CMS.DLL ze složky CMS. Dalším krokem je vložit odkaz na tuto referenci pomocí direktivy „Register“ do všech stránek webové aplikace, které mají využívat moduly CMS. Posledním krokem je změna bázové třídy všech stránek, na které chceme umisťovat CMS moduly, ze `System.Web.UI.Page`

na `Getmore.CMS.Web.UI.CMSPage`. Následující obrázek ukazuje použití direktivy „Register“ v ASPX souboru:

```
<%@ Page Language="C#" MasterPageFile="~/MainLayout.Master" Codebehind="DMS.aspx.cs" %>
<%@ Register Assembly="Getmore.CMS" Namespace="Getmore.CMS.Modules.DMS" TagPrefix="gm" %>

<asp:Content ContentPlaceHolderID="MainPageContentHolder" ID="MainPageContent" runat="server">
        <gm:DMSModule runat="server" ID="MainDMS" TopParentFolderCode="ROOT">
        </gm:DMSModule>
</asp:Content>
```

Obrázek 13 – Ukázka použití direktivy „Register“ v ASPX souboru

Tento postup v sobě skrývá jednu velkou výhodu. Veškeré stránky, které nevyužívají moduly CMS, zůstávají naprosto beze změny a neobsahují žádné zbytečné implementace, které by zpomalovaly chod webové aplikace. Z obrázku je také vidět, jak jednoduše je možno do aplikace vložit kompletně fungující dokumentový systém bez nutnosti psát jakýkoliv další kód.

2.5 Popis implementace modulů

Tato kapitola popisuje implementaci základních modulů. Stanovený rozsah této bakalářské práce bohužel neumožňuje popsat implementaci všech modulů a popsat ji do takových podrobností, jak by bylo vhodné. Popis implementace je veden tak, aby bylo zřejmé, jak celkový koncept systému funguje. Při přípravě dalších modulů se bude postupovat velmi podobně.

2.5.1 Základní třídy

2.5.1.1 Třída CMSPage

Jedná se o abstraktní třídu, kterou musí zdědit každá stránka webové aplikace, na které chceme používat moduly CMS. Tato třída je součástí projektu Getmore.CMS a náleží do oboru názvů `Getmore.CMS.Web.UI`. Tato třída dědí ze základní třídy `System.Web.UI.Page` a rozšiřuje ji mimo jiné o vlastnosti pro implementaci datové vrstvy nebo o implementaci vrstvy pro zabezpečení a opravňování. Popis veřejných položek je uveden dále v textu vždy u konkrétní implementace.

2.5.1.2 Třída CMSModule

Tato třída je také abstraktní a patří do stejného projektu a oboru názvů jako třída CMSPage. Všechny moduly nabízené v rámci CMS dědí tuto třídu. Třída sama o sobě dědí třídu `System.Web.UI.WebControls.WebControl`, což zajišťuje, že každý modul CMS je současně serverovým ovládacím prvkem. Třída udržuje tyto důležité veřejné položky:

- **ModuleCode –** Kódové označení modulu v systému. Tato hodnota je důležitá pro nastavování oprávnění pro daný modul. Jednoznačně identifikuje daný modul vůči vrstvě zabezpečení resp. vůči databázi. Tato položka je i pro zápis a její definice při vkládání modulu do webové aplikace je vyžadována.
- **Page –** Tato položka překrývá definici položky `Page` zděděnou z nadřazené úrovně. Položka se pokouší přetypovat hodnotu `base.Page` na datový typ `CMSPage`. Pokud položka `base.Page` není typem `CMSPage` je vyhozena specifická výjimka. Tato kontrola zajistí, že CMS moduly jsou umístěny jen na stránkách dědících ze třídy `CMSPage`.
- **PermissionsHolder –** Jedná se o položku typu `ModulePermissionsHolder`. Tato položka „drží“ oprávnění pro daný CMS modul. Podrobný popis této položky je uveden v kapitole 2.5.3 Vrstva zabezpečení a opravňování.

2.5.2 Datová vrstva

Jak již bylo popsáno výše, je datová vrstva realizována výhradně v projektu Getmore.CMS.DataWrapper. Základní myšlenkou je umožnit webové stránce jeden přístupový bod do databáze, který mohou využívat všechny moduly umístěné na dané stránce. Z logiky webové aplikace vyplývá, že tento přístup k datovému zdroji musí být umístěn na každé stránce CMS zvlášť. Není možno umístit objekty pro přístup k datovému zdroji na vyšší úroveň než na úroveň stránek. Snad jediným řešením by bylo uchovávat objekty pro přístup k datům v cache nebo application objektech. Toto řešení by pak umožnilo vytvoření například jediného globálního spojení do databáze, které by mohly využívat všechny moduly CMS. Došlo by k výraznému ušetření systémových prostředků, které by bylo jinak nutno vynakládat na soustavné vytváření nových a nových spojení pro každý modul, resp. pro každou operaci vůči databázi. Ironický nádech předchozí věty předznamenal, že toto řešení je naprosto nesprávné. Jediné globální spojení do databáze

by mohlo způsobit závažné ztráty výkonu při běhu aplikace. Prvním důvodem je to, že i webové aplikace mohou být webovým serverem zpracovávány ve více vláknech. Vícevláknové provádění je například možno povolit nad fondem aplikací webového serveru IIS 7.0 a má významný vliv na výkon aplikace, zvláště pokud se jedná o často navštěvovanou stránku. Jednotlivé http dotazy se mohou vykonávat asynchronně, aniž by se navzájem blokovaly. Teď se dostáváme k jádru problému. Pokud by aplikace využívala globální spojení do databáze umístěné například v application objektu, pak by vícevláknové zpracování ztratilo kompletně na svém významu, protože veškerá vlákna by se musela dělit o jediné spojení do databáze. Nejhorším případem by byl stav tzv. „deadlocku“, kdy by bylo nutno po detekci „sebeuzamčení“ ukončit relaci jednoho vlákna, což by způsobilo pád aplikace jednomu z klientů. A je zřejmé, že tento způsob implementace přístupu do databáze by byl velmi náchylný na vznik těchto „deadlocků“. Globální spojení do databáze navíc kompletně ztrácí na významu, protože ASP.NET aplikace mohou využívat tzv. „connection pooling“. [9]

Technologie „connection pooling“ slouží právě k tomu účelu, aby se pro každý nový dotaz na datový zdroj nemusel vytvářet nový objekt spojení. Aplikace si udržuje fond použitých spojení a při požadavku na databázové spojení může poskytnout již vytvořené a právě nevyužívané spojení do databáze. V případě, že není žádné spojení volné, je možno do fondu přidat další spojení. Tuto operaci je však možno provést jen za předpokladu, že nebylo dosaženo maximálního počtu možných spojení ve fondu. V případě, že je dosaženo maximálního počtu spojení, je požadavek na spojení zařazen do fronty a čeká na uvolnění některého spojení ve fondu nebo na vypršení „timeoutu“ daného požadavku na spojení. Connection pooling je možno aktivovat nebo deaktivovat. Dále je možno nastavit maximální počet spojení ve fondu a také minimální počet spojení ve fondu. Minimální počet určuje, kolik spojení se implicitně vytvoří při založení fondu spojení. Tato hodnota je ve výchozím nastavení rovna 0. [8]

Po vyloučení možnosti udržovat globální spojení do databáze jsem vypracoval návrh třídy `CMSPage` (viz 2.5.1.1 Třída CMSPage). Datová vrstva je implementována readonly položkou `DataLayer`, která je typu `CMSDataLayer`. Privátní instance této položky je vytvářena až při prvním dotazu na datovou vrstvu dané stránky, což zabezpečí, že instance není nikdy vytvářena zbytečně.

2.5.2.1 Třída CMSDataLayer

Tato třída patří do oboru názvu `Getmore.CMS.DataWrapper`. Jejím posláním je poskytovat přístup k datovým vrstvám pro jednotlivé moduly CMS. Pro vytvoření instance této třídy je nutno zadat connection string do datového zdroje a odkaz na objekt cache konzumující stránky. Odkaz na objekt cache je nutný pro případné cachování některých dat v systémových modulech, které nejsou reprezentovány jako WebControls a nemají tedy přímý přístup do objektu cache. Například se jedná o modul centrálního opravňování. Tato třída poskytuje následující veřejné (public) metody:

- **CreateNewSQLConnection()** – Metoda vrací objekt typu `SqlConnection` do datového zdroje specifikovaného připojovacím řetězcem při vytváření instance třídy.
- **CreateNewSQLCommand()** – Metoda vrací objekt typu `SqlCommand`, který je vázán na nové spojení do databáze.
- **GetDBName()** – Metoda vrátí název aktuální použité databáze. Tato hodnota je získána z připojovacího řetězce, který si třída udržuje jako privátní položku. V implementaci je s výhodou použito objektu typu `SqlConnectionStringBuilder`.

Třída dále obsahuje veřejné položky, které reprezentují jednotlivé elementy datové vrstvy. Datová vrstva je totiž pro větší přehlednost rozčleněna na celky, které odpovídají modulům CMS. Instance těchto pod-vrstev jsou vytvářeny až při prvním požadavku na konkrétní pod-vrstvu. Tento princip opět zajišťuje, že se zbytečně nevytváří instance pod-vrstev, které nejsou na dané CMS stránce potřeba. V současné době třída disponuje těmito pod-vrstvami:

- **Permissions** – Tato položka je datového typu `PermissionsDataLayer` a je přístupovým bodem do pod-vrstvy pro ověřování.
- **Articles** – Tato položka je datového typu `ArticlesDataLayer` a je přístupovým bodem do pod-vrstvy pro moduly typu články.
- **DMS** – Tato položka je datového typu `DMSDataLayer` a je přístupovým bodem do pod-vrstvy pro moduly typu DMS.

Soubor pod-vrstev bude dále rozšiřován s přibývajícími moduly CMS. Pro definici nové pod-vrstvy je nutno dědit ze základní bázové abstraktní vrstvy `DataLayer`. Tato

bázová třída slouží pro držení instance na rodičovskou CMS datovou vrstvu – tedy instanci typu `CMSDataLayer`. Tato instance je reprezentována položkou `MainDataLayer`.

2.5.2.2 Třída ArticlesDataLayer

Tato třída popisuje datovou pod-vrstvu pro moduly typu „články“. Uvádím ji jako příklad, protože všechny ostatní třídy popisující datové pod-vrstvy jsou velmi podobné.

Třída definuje konstruktor s jediným parametrem, který udává instanci nadřazené vrstvy a explicitně volá konstruktor bázové třídy `DataLayer`, kterému tento parametr předává.

Třída disponuje jedinou veřejnou metodou **GetArticlesDataSource(…)**. Tato metoda vrací objekt typu `ArticlesDataSource` (viz 2.5.2.3 Třída ArticlesDataSource). Parametrem metody je řetězec s kódem modulu typu „články“, ke kterému chceme získat datový zdroj. Metoda pouze vytvoří novou instanci třídy `ArticlesDataSource` a vrátí ji jako výstupní hodnotu.

2.5.2.3 Třída ArticlesDataSource

Třída popisuje datový zdroj pro modul typu články. Její konstruktor vyžaduje zadání kódu modulu typu „články“ a také připojovacího řetězce pro vytváření spojení. Tato třída dědí základní .NET datový zdroj `SqlDataSource`. V konstruktoru dochází k naplnění základních vlastností `SqlDataSource` jako `SelectCommand`, `InsertCommand`, `UpdateCommand` a `DeletCommand` a jiných vlastností, které slouží pro zajištění CRUD operací nad daným datovým zdrojem. Jsou definovány názvy SQL stored procedur a jejich parametry. Třída přidává jedinou veřejnou metodu SelectData(), která vrací objekt typu `DataView` obsahující všechna data poskytovaná tímto datovým zdrojem. Tato metoda je s výhodou použita při implementaci seznamu článků, kde nebylo možno využít standardních vlastností prvku `SqlDataSource`. O veškeré další CRUD operace v daném modulu se stará výhradně tento datový zdroj.

2.5.3 Vrstva zabezpečení a opravňování

Vrstva zabezpečení a opravňování je rozšířením základních ASP.NET přístupů k users and roles managementu. Důvody pro toto rozšíření jsou popsány v kapitole 2.2.2. Tato kapitola popisuje samotnou implementaci této vrstvy v CMS.

Většina implementací je v rámci oboru názvu `Getmore.CMS.Permissions`. Vstupním bodem do vrstvy pro ověřování je položka třídy `CMSPage` (viz. 2.5.1.1 Třída CMSPage) `Permissions`. Tato položka je datového typu `PermissionsProvider`.

2.5.3.1 Třída PermissionsProvider

Třída popisuje poskytovatele zabezpečení a opravňování pro celou aplikaci a její instance je držena každou stránkou dědící třídu `CMSPage`. Základní funkcí třídy je vytvářet objekty typu `ModulePermissionsProvider` pro různé CMS moduly.

Pro vytvoření instance této třídy je nutno zadat instanci stránky typu `CMSPage`, na které se bude nová instance nacházet. Tento odkaz na rodičovskou stránku je důležitý kvůli přístupu do datové vrstvy dané stránky. `PermissionsProvider` totiž musí mít možnost načítat oprávnění z databáze. V konstruktoru třídy dále dojde k načtení všech oprávnění z databáze nebo z paměti cache. Pokud data v paměti cache neexistují, jsou vyžádána z databáze a uložena do paměti cache. Cache paměť má nastavenou závislost na tabulku tb_permissions – tedy k vypršení paměti cache dojde pouze v případě, že došlo ke změně v tabulce tb_permissions. Uložení oprávnění do paměti cache má své opodstatnění. Dat s oprávněními je poměrně hodně a nejsou často měněna. Navíc se může na stránce nacházet více modulů, které vyžadují kontrolu oprávnění, což znamená, že data jsou potřeba velmi často. Implementace pomocí cache by se měla pozitivně odrazit na celkovém výkonu aplikace. Díky závislosti (`SqlCacheDependency`) je navíc zajištěno, že data jsou vždy platná a aktuální.

Třída v současné době poskytuje jedinou veřejnou metodu:

- **GetModulePermissions(…)** – Metoda vrátí „balíček oprávnění“ pro specifikovaný modul. Modul je určen vstupním parametrem ModuleCode. Balíček pro daný modul je dán filtrací z datového zdroje oprávnění, který je držen jako privátní položka typu `DataSet` v třídě `PermissionsProvider`. Datový typ tohoto balíčku je popsán dále.

2.5.3.2 Třída ModulePermissionsHolder

Jedna instance této třídy je držena každým CMS modulem a umožňuje uvnitř modulu provádět dotazy na oprávnění přihlášeného uživatele. Uvnitř třídy je objekt typu

`PermissionsHashTable`, ve kterém jsou specifickým způsobem uložena všechna oprávnění pro všechny uživatele a role vztahující se k danému modulu. Vazba mezi modulem a sadou oprávnění je realizována pomocí již několikrát zmíněné vlastnosti ModuleCode. Ve zmíněné hash tabulce jsou uloženy objekty typu `PermissionsHashTableItem`. Tato třída poskytuje veřejné metody na ověřování libovolných oprávnění pro daný modul. Jsou připraveny metody, které umožňují velmi jednoduše získávat základní oprávnění. Jedná se o tyto veřejné metody:

- CanAdminister()
- CanView()
- CanEdit()
- CanAdd()
- CanDelete()

Tyto bezparametrické metody okamžitě vrátí hodnotu typu `bool` indikující zda aktuálně přihlášený uživatel může v modulu provádět danou akci. Výše uvedené metody slouží pro základní oprávnění, která jsou stejná v rámci všech modulů. Tyto metody nedělají nic jiného, než že volají další veřejnou metodu této třídy, která slouží pro ověření libovolného oprávnění identifikovaného kódem oprávnění v rámci modulu:

- **CheckModulePermission(…)** – Ověří libovolné oprávnění v rámci daného modulu. Parametrem je kód oprávnění definovaného v rámci modulu. Toto oprávnění musí být definováno v databázi. Pokud není, je vyhozena specifická výjimka. Tato metoda sama o sobě nevyhodnocuje oprávnění, ale volá metodu třídy `PermissionsHashTable` – **CheckPermission**(…).

Každý CMS modul pak může prostřednictvím položky `PermissionsHolder` volat výše popsané metody a podle jejich výsledku např. zobrazovat nebo skrývat administrátorské prvky, nebo přesměrovávat na chybovou stránku v případě, že se uživatel pokouší provádět akce, na které (již) nemá oprávnění.

2.5.3.3 Třída PermissionsHashTableItem

Jak již bylo zmíněné výše v textu, jedná se o jednu položku z kolekce typu `PermissionsHashTable`. Každá tato položka se vztahuje k jednomu oprávnění v rámci modulu a drží seznam uživatelů a rolí, kterým je toto oprávnění uděleno a kterým bylo

explicitně odebráno. Vlastní vyhodnocení výsledného oprávnění pro daného uživatele je realizováno veřejnou metodou **GetResultPermissionForCurrentUser**(). Tato bezparametrická metoda si získá instanci aktuálně přihlášeného uživatele a zjistí jeho zařazení do rolí. Poté na základě obsahu konkrétní instance vyhodnotí výsledné oprávnění pro uživatele a vrátí hodnotu typu `bool` indikující, zda uživatel má nebo nemá dané oprávnění. Výsledné oprávnění je dáno výrazem: (Alespoň jedné roli uživatele je oprávnění uděleno A ZÁROVEŇ není oprávnění explicitně odebráno uživateli) NEBO je oprávnění explicitně uděleno uživateli.

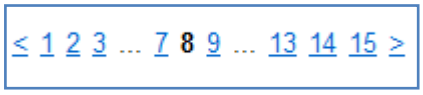
2.5.4 Články

Tato kapitola zkráceně popisuje implementaci modulu „články“. Pro zjištění detailní funkcionality je asi lépe se podívat do zdrojových kódů, nebo do nápovědy vygenerované z XML komentářů u tříd a metod. Nejsou zde například popisovány některé business objekty, enumerátory apod.

2.5.4.1 Třída ArticlesModule

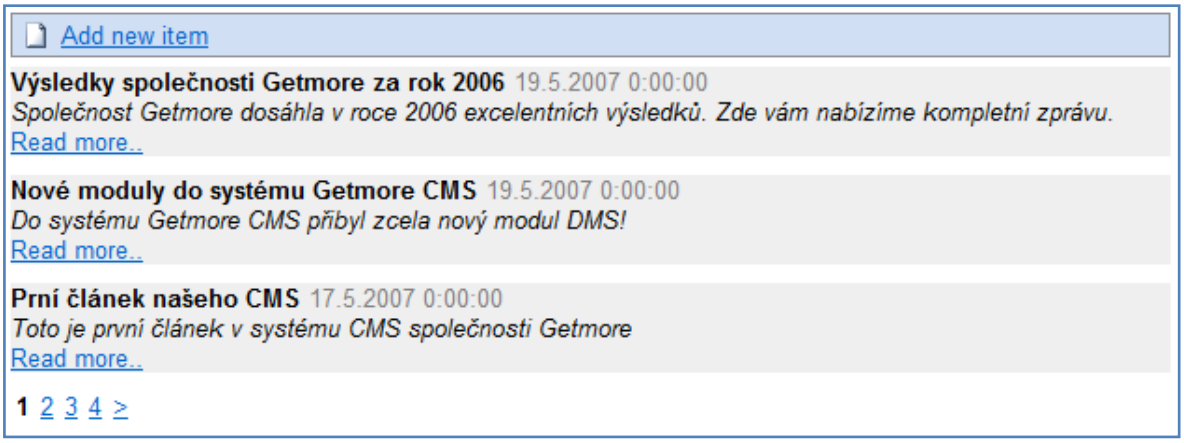
Tato třída dědí abstraktní třídu `CMSModule`, což z ní činí serverový ovládací prvek. Tyto ovládací prvky je možno naprosto libovolně rozmisťovat do webové aplikace. Na jedné ASPX stránce se dokonce může nacházet více instancí této třídy.

Třída nabízí tři základní pohledy. Tyto pohledy jsou realizovány pomocí klasického serverového ovládacího prvku `MultiView`. Prvním pohledem je seznam článků. Tento pohled umožňuje zobrazit články jako stránkovaný seznam. Programátor může tomuto prvku nastavit maximální počet článků zobrazený na jedné stránce. Toto nastavení se provádí pomocí veřejné položky `MaxPageItemCount`. Položka musí být nastavena na hodnotu větší než 0. Stránkování v rámci seznamu je řízeno vlastním ovládacím prvkem `PagingNavigator`. Tento prvek umožňuje na základě informací ze seznamu (jako například celkový počet položek, maximální počet položek na jedné stránce) vygenerovat velmi sofistikovaný ovládací prvek umožňující pohyb mezi stránkami. Hlavní výhodou tohoto prvku je schopnost skrývat některá čísla stránek, čímž je dosaženo velmi kompaktního vzhledu i v případě velkého počtu stránek.



Obrázek 14 – Sofistikovaná inteligentní komponenta PagingNavigator

Velmi důležitou funkcí seznamu článků je schopnost uživatelsky stanovit vzhled jedné položky seznamu pomocí HTML šablony. Programátor implementující tento modul do své webové aplikace má tedy kompletní kontrolu nad vzhledem modulu. Šablona nabízí datové a ovládací elementy, které může programátor libovolně rozmístit do svého layoutu. Pokud šablona není explicitně definována, je použita defaultní šablona. Vzhled defaultní šablony není možno měnit, protože je napevno implementována jako třída `ArticleListItemControlTemplate`.



Obrázek 15 – Ukázka GUI – seznam článků pomocí výchozí šablony

Druhý pohled tohoto modulu je tzv. „detail článku“. Tento pohled slouží pro zobrazení kompletního článku. Vzhled pohledu je opět plně v rukou programátora implementujícího modul. Je možno definovat libovolnou šablonu detailu článku a rozmístit v ní datové a ovládací elementy. Opět je k dispozici defaultní šablona definovaná jako třída. Obsah tohoto pohledu je definován třídou `ArticleDetail`, která je popsána níže.

Třetím pohledem je „editovatelný detail článku“. Tento pohled umožňuje provádět přidávání, editaci nebo mazání článků. Pohled je dostupný jen pro uživatele s patřičným oprávněním. V rámci tohoto pohledu je implementován WYSIWYG editor pro editaci hlavního textu článku. Obsahem tohoto pohledu je třída `ArticleEditableDetail`, která je popsána níže.

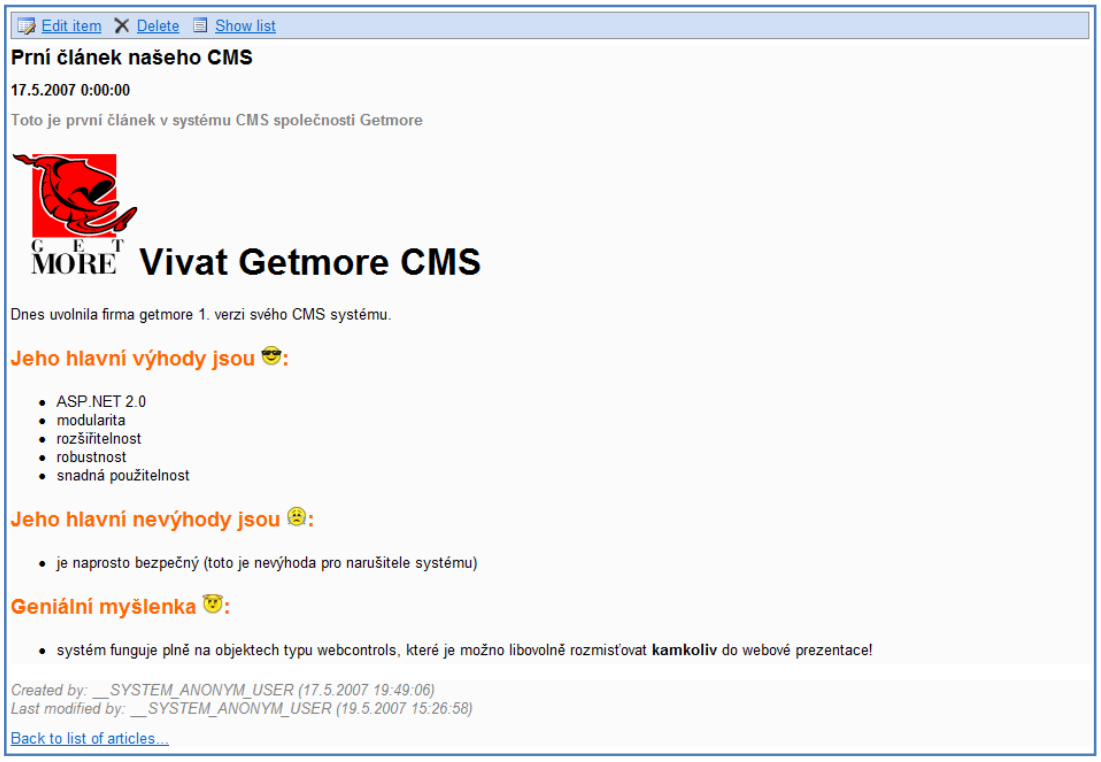
Pro navigaci mezi jednotlivými pohledy a pro provádění administrátorských operací je v modulu zobrazen prvek typu `ToolBar` – nástrojová paleta. Tento prvek na základě oprávnění zobrazuje příkazová tlačítka pro provádění administrativních operací. Panel je zobrazen jen v případě, že uživatel má alespoň nějaká administrativní oprávnění. V současné době modul disponuje jen základní sadou oprávnění – tj. práva s kódy „VIEW“, „ADD“, „EDIT“ a „DELETE“.

2.5.4.2 Třída ArticleDetail

Tato třída popisuje objekt, který tvoří obsah pohledu „detail článku“ v modulu „články“. Jedná se o šablonový serverový ovládací prvek. Do konstruktoru tohoto prvku vstupuje business objekt typu `ArticleContainer`, který drží data jednoho článků. Tyto business objekty jsou vytvářeny při generování seznamu článků a jsou vázány ke každé položce v seznamu. Objekt je součástí argumentů události indikují žádost o vstup do pohledu „detail článku“. Je proto velmi vhodné a jednoduché předat detailu článku tento „hotový“ business objekt než celý objekt `ArticlesDataSource`, ve kterém by bylo nutno hledat vybranou položku podle indexu článku. Tato třída neumožňuje žádnou editaci dat, a proto je tento způsob předání dat na úrovni business objektu velmi vhodný. Dalším vstupním parametrem konstruktoru je objekt typu `ITemplate`. Jedná se o šablonu prvku, která je však předávána z objektu `ArticlesModule`, v rámci kterého ji programátor při implementaci definuje. Pokud tato šablona není definována (objekt nabývá hodnoty `null`), pak je vytvořena instance šablony `ArticleDetailDefaultTemplate`.

Tato třída také obsahuje veřejnou událost (`event`) `ArticleDetailCommand`. Tato událost umožňuje zasílat nadřazenému objektu `ArticlesModule` různé příkazy. V současné době je definován jediný příkaz „ViewArticlesList“, který zajistí přechod zpět na seznam článků. Programátor může do své šablony umístit libovolný ovládací prvek umožňující stanovit vlastnost `CommandName` (`Button`, `LinkButton`…) a nastavit tuto vlastnost na hodnotu „ViewArticlesList“. Třída `ArticleDetail` zachytává události jdoucí ze šablony a provádí jejich analýzu. V případě, že se jedná o událost nesoucí atributy události typu `CommandEventArgs` a vlastnost `CommandName` je nastavena na jeden ze známých podporovaných příkazů, pak je vyvolána událost `ArticleDetailCommand`, která je obsloužena v objektu `ArticleModule` změnou aktuálního pohledu na pohled „seznam článků“. Z výše uvedeného textu je vidět, že programátor není při nasazení modulu ničím

limitován a je mu dokonce umožněno libovolné rozmístění navigačních prvků triviálním nastavením vlastnosti `CommandName`.

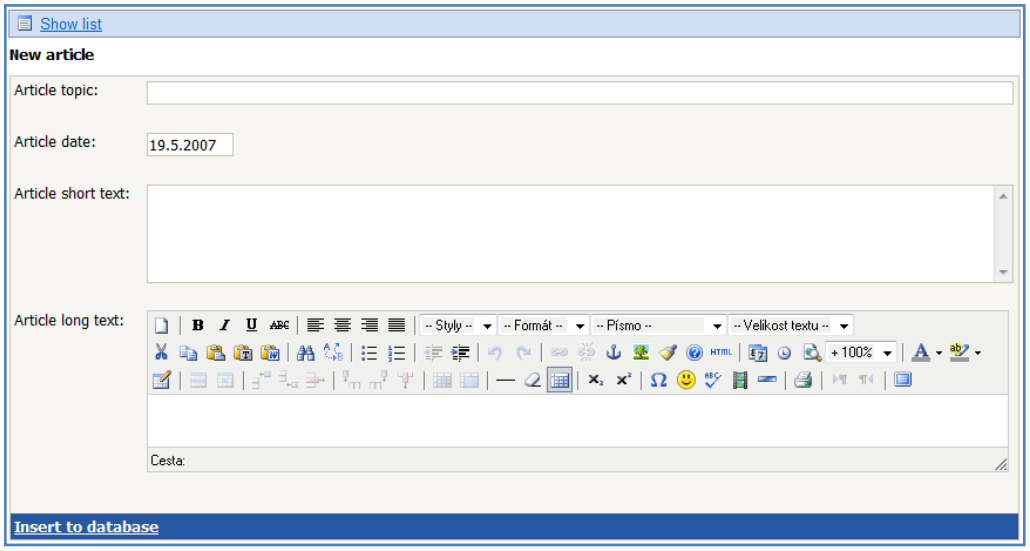


Obrázek 16 – Ukázka GUI – detail článku realizovaný pomocí třídy ArticleDetail

2.5.4.3 Třída ArticleEditableDetail

Tato třída popisuje objekt pro editaci dat jednoho článku a je obsahem pohledu „editovatelný detail článku“ v modulu `ArticlesModule`. Tento objekt není na rozdíl od předchozích objektů uživatelsky přizpůsobitelný. Jedná se totiž o plně administrativní ovládací prvek, který vidí jen uživatelé s patřičnými oprávněními. Vnitřně tento objekt využívá standardní ovládací prvek `FormView`, který slouží k zobrazení jedné položky z datového zdroje. Prvek obsahuje čtyři pohledy – pro zobrazení, editaci, vložení a mazání. Pohled pro zobrazení není využit, protože zobrazení článku je realizováno pomocí třídy `ArticleDetail`. Prvek `FormView` je šablonový prvek a je nutno definovat šablony pro jednotlivé pohledy. V prvku je to řešeno vytvořením instance jediné šablony, které se přidá indikace, o jaký pohled se jedná. Na základě nastavení pohledu pak šablona generuje různá datová pole a různé ovládací prvky. Tato šablona je definována třídou `ArticleDetailEditItemTemplate`.

V konstruktoru prvku je nutno zadat zdroj dat typu `ArticlesDataSource`, index aktuálního řádku ve zdroji a indikaci aktuálního pohledu (editace, přidávání, mazání). Zdroj dat je napojen na vnitřní ovládací prvek `FormView` (toto přiřazení je možné, protože `ArticlesDataSource` dědí ze třídy `SqlDataSource`). Veškeré CRUD operace jsou pak již prováděny standardní cestou, tak jak je nabízí ovládací prvek `FormView`.



Obrázek 17 – Ukázka GUI – prvek ArticleEditableDetail

2.5.4.4 Poznámka k využití šablon

Šablony jsou velmi silným nástrojem pro vývoj serverových ovládacích prvků. Bohužel však stále chybí detailnější popisy a návody, jak vytvářet šablony pomocí programového kódu. Většina článků zabývajících se šablonami v ASP.NET popisuje jejich vytváření tzv. deklarativní metodou. Tato metoda spočívá v sestavování šablony pomocí HTML elementů a elementů datového zdroje. Vytváření šablony pomocí programovací techniky je už o něco málo náročnější. V zásadě existují dva typy šablon. První typ je pouze pro zobrazení. Pokud chceme vytvořit takovou šablonu, je nutno implementovat rozhraní `ITemplate`. Příkladem takové šablony je třída `ArticleDetailDefaultTemplate`. Z kódu této třídy je zřejmé, jak celá záležitost funguje. Druhý typ šablon je tzv. obousměrně vázaný a umožňuje data nejen zobrazovat, ale také vracet hodnoty zpět do datového zdroje. Pokud chceme vytvořit takovou šablonu je nutno implementovat rozhraní `IBindableTemplate`. Toto rozhraní stanovuje mimo jiné metodu, která slouží pro vyparsování dat ze šablony. Příkladem obousměrně vázané šablony je třída `ArticleDetailEditItemTemplate`.

Protože vytváření šablon pomocí programování je poměrně časově náročná záležitost, nalezl jsem způsob, jakým je možno vytvářet objekty šablon z ASPX souborů. ASP.NET nabízí metodu `LoadTemplate()`. Tato metoda však slouží pro načtení šablony, která je pouze pro čtení (tj. vrací objekt typu `ITemplate` a nikoliv `IBindableTemplate`). Toto omezení je možno obejít metodou `LoadControl()`. Tato metoda dokáže načíst definici ovládacího prvku ze souboru ASPX. Pokud do souboru vložíme nějaký datový ovládací prvek, který využívá obousměrně vázané šablony (například `FormView`) a definujeme jednu z jeho šablon (např. `EditItemTemplate`), můžeme pak po načtení prvku pomocí metody `LoadControl()` přistoupit k jeho položce `EditItemTemplate` a tu vrátit jako obousměrně vázanou šablonu. Pro tuto funkcionalitu byla v systému vytvořena třída `TemplatesHandling`, která obalovala výše popsanou funkcionalitu a zároveň poskytovala cachování šablon pro jejich rychlé načítání (přístup do souboru je poměrně časově náročná operace). Tato funkcionalita však v současné době není nikde použita, protože veškeré šablony byly nakonec realizovány pomocí programového kódu. [2]

2.5.4.5 Implementace WYSIWYG editoru

V modulu články je využit WYSIWYG editor pro editaci hlavního textu článku. K tomuto účelu byl použit volně šiřitelný produkt TinyMCE. Editor je napsán kompletně v javascriptu. Jeho integrace do webové stránky je velmi snadná a spočívá v přiřazení konkrétního názvu třídy kaskádových stylů k elementu typu TEXTAREA. Implementace do rámce rozhraní ASP.NET nebyla však zcela bez problémů. Bylo nutno napsat speciální validátory pro prvky využívající tento editor. Prvek totiž nelze validovat standardními validátory, protože editace obsahu pole probíhá v oddělených datových strukturách a výsledný HTML kód z editoru je do pole přenesen až při odesílání webového formuláře. Tato událost však nastává až po standardních validačních událostech, kdy objekt typu TEXTAREA obsahuje stále výchozí text, jaký byl do něj načten při vygenerování stránky. Validátor pro prvek s editorem TinyMCE musí před započetím validace zavolat metodu editoru pro přenesení HTML kódu do objektu TEXTAREA.

2.5.5 DMS

Dokumentový systém funguje podobně jako modul články. Opět je možno v rámci webové aplikace vytvářet libovolné množství instancí tohoto typu modulu a každé instan-

ci přidělit ModuleCode a oprávnění. Všechny instance však sdílejí jeden adresářový strom. Při vytváření instance modulu je také nutno zadat kód složky, která se stane kořenem adresářového stromu pro danou instanci dokumentového systému. Instance dokumentového systému pak zobrazuje jen pod-složky a soubory ve specifikované složce. Na základě oprávnění je také možné zcela potlačit zobrazení procházení složek a DMS tak slouží jenom jako seznam souborů. Nejčastěji bude v rámci webové aplikace existovat jeden DMS modul, který bude určen jen pro administraci a bude zobrazovat celý adresářový strom. Dále budou v aplikaci další instance DMS, které však budou sloužit jenom jako seznamy souborů. Do budoucna se plánuje s přidáním nové funkcionality, která umožní přiřazovat dokumenty z DMS k dalším elementům v systému (například ke článkům apod.)

2.5.5.1 Třída DMSModule

Jedná se o veřejnou třídu, která vystupuje jako serverový ovládací prvek a dědí třídu `CMSModule`. Třída jen logicky spojuje objekty `DMSFoldersPanel` a `DMSDocumentsPanel`. Jedním ze základních principů je poslouchat události od objektu `DMSFoldersPanel` a na základě nich aktualizovat objekt `DMSDocumentsPanel`. Například pokud v objektu `DMSFoldersPanel` dojde ke změně aktuální složky, musí dojít k aktualizaci seznamu souborů v objektu `DMSDocumentsPanel`. Třída dále poskytuje oprávnění pro tyto dva objekty.

2.5.5.2 Třída DMSFoldersPanel

Tato třída vystupuje jako serverový ovládací prvek a dědí třídu `WebControl`. Nejedná se tedy o samostatný CMS modul. Ovládací prvek má za úkol zajistit procházení adresářovým stromem dokumentového systému. Konstruktor této třídy přebírá kód složky, která má sloužit jako kořenový adresář pro tuto instanci modulu. Dále přebírá instanci rodičovského modulu - `DMSModule` pro přístup k oprávněním a k datové vrstvě[1]. Z datového zdroje načítá seznam složek a filtruje jej podle aktuální zvolené složky. Prvek opět nabízí více pohledů. Prvním pohledem je seznam složek, ve které je možno se pohybovat. Kliknutím na název složky je možné se zanořit do další úrovně. Tlačítkem „Nahoru“

[1] Nejedná se o CMS modul – nemůže sám přistupovat do datové vrstvy nebo do vrstvy zabezpečení

je možno se dostat o úroveň výš. Nejvýše však do kořenové složky stanovené pro tuto instanci modulu. Tento pohled je realizován datovým prvkem `GridView`. Druhým pohledem je „editace složky“. Tento pohled umožňuje upravovat existující složky (název, popis, kód), přidávat nové složky do aktuální složky nebo smazat zvolenou složku. Tento pohled je realizován prvkem `FormView`. Implementace prvku `FormView` a jeho šablon je zcela totožná jako v modulu „články“. Jak již bylo zmíněno výše, prvek obsahuje jednu veřejnou událost. Jedná se o událost `CurrentFolderChanged`. Tato událost nastává, pokud dojde ke změně aktuální složky. Konzumentem této události je nadřazený objekt `DMSModule`, který na jejím základě aktualizuje `DMSDocumentsPanel`. Tento prvek využívá objekt typu `DMSFodlersDataSource` jako datový zdroj. Tento datový zdroj opět dědí ze třídy `SqlDataSource` a ve svém konstruktoru definuje SQL příkazy pro CRUD operace. Datový zdroj obsahuje vždy seznam všech složek a při procházení složkami dochází jen ke správnému nastavení položky `FilterExpression`. Tento datový zdroj není prozatím ukládán do paměti cache.

2.5.5.3 Třída DMSDocumentsPanel

Jedná se opět o serverový ovládací prvek, který dědí ze třídy `WebControl`. Jeho úkolem je zobrazovat seznam souborů v aktuální složce. Po inicializaci modulu DMS jsou zobrazeny soubory, které se nacházejí ve stanovené kořenové složce. Modul poskytuje veřejnou vlastnost `CurrentFolderId`, která udává ID složky, jejíž souborový obsah je právě zobrazen v modulu. Prvek nabízí dva pohledy – první pohled pro zobrazení seznamu souborů a druhý pohled pro administrativní operace s jedním souborem (editace, vkládání nového, mazání). Seznam souborů je realizován datovým prvkem `GridView` a administrativní pohled prvkem `FormView`. Seznam souborů také umožňuje stahovat soubory kliknutím na název souboru.

Jak již bylo popsáno výše, DMS neumožňuje uploadovat nové soubory. Umožňuje pouze vytvářet virtuální obrazy již existujících souborů v internetu. Při přidávání nového dokumentu do systému stačí pouze zadat jeho URL. Do budoucna se počítá s rozšířením funkcionality o upload nových souborů. Tato funkcionalita však bude vyžadovat podrobně prostudovat možnosti portálu Microsoft SharePoint, aby bylo umožněno přidávat nové soubory přímo do aplikace SharePoint přes webové rozhraní CMS.

ZÁVĚR

V teoretické části práce se podařilo podat ucelený stručný pohled na technologii ASP.NET 2.0 jako na nástroj vhodný pro vývoj webových aplikací. Druhý oddíl teoretické části rozebral téma CMS a představil tři existující řešení redakčních systémů. Tato tři řešení byla stručně srovnána.

V rámci praktické části práce se podařilo navrhnout a vytvořit Content management system pro společnost Getmore s.r.o.. Při vývoji softwaru bylo postupováno podle doporučených postupů. Nejprve byla provedena hrubá analýza řešení a byly stanoveny základní myšlenky, vize a cíle projektu. Na základě této analýzy bylo vypracováno uživatelské zadání od společnosti Getmore s.r.o.. Podle tohoto zadání pak bylo navrženo programátorské řešení. Návrh tohoto řešení je nedílnou součástí této bakalářské práce. Do návrhu programátorského řešení patří celkový koncept, rozvržení projektů, rozvržení tříd a rozhraní, návrh databázové struktury apod. Po schválení tohoto konceptu společností Getmore s.r.o. bylo zahájeno programování. Velmi podstatnou částí práce bylo připravit prostor pro vývoj rozšiřitelné aplikace. To obnášelo dobře promyslet především vrstvu zabezpečení a opravňování a také datovou vrstvu. Během vývoje došlo v těchto základních implementacích k několika změnám oproti původnímu návrhu. Je však nutno poznamenat, že se jednalo o změny k lepšímu. Implementace popsané v této bakalářské práci odpovídají současnému skutečnému stavu software.

Při implementaci jednotlivých modulů došlo k výraznému vylepšení funkcionalit oproti původnímu uživatelskému zadání společnosti Getmore. Především se jedná o implementaci uživatelsky definovatelných šablon pro maximální přizpůsobení vzhledu CMS modulů. Původní zadání předpokládalo změnu vzhledu jen pomocí kaskádových stylů. Práce se v tomto ohledu stává velmi cenným zdrojem pro programátory, kteří chtějí implementovat vlastní šablonové ovládací prvky. V této oblasti totiž selhaly veškeré knižní zdroje, které jsem měl k dispozici. Zdrojové kódy jsou bohatě okomentovány a obsahují využití šablon snad se všemi možnými funkcionalitami včetně například „events bubbling“. Ve své práci jsem se snažil o využívání objektových vlastností jazyka jako dědění, polymorfismus, zapouzdření, události apod. Domnívám se také, že aplikace je dobrým příkladem maximálního využití výhod prostředí ASP.NET 2.0.

ZÁVĚR V ANGLIČTINĚ

In the theoretical part was got a compact brief view to the ASP.NET technology. The technology was described as a good tool to create web applications. The second section of the theoretical part discussed the CMS theme and introduced three existing solutions of content management systems. These three solutions were briefly compared.

In the practical part of this dissertation was created the content management system for the Getmore s.r.o. Corporation. During the development of the system was act upon to recommended orders. At first was done a gross analysis of the project. Main ideas, views and points were determined. On the basis of the analysis was created a user submission from the Getmore s.r.o. Corporation. According to the submission was made a programmer concept. The programmer concept is integral part of this dissertation. Into the programmer concept belongs whole vision, project design, classes and interfaces design, concept of database structure etc. Programming was started after the concept authorization by Getmore s.r.o. Corporation. The very important point of the work was to clean the ground for development of an extensible application. This contained to think out well especially permission and data layer. During the development the original concept was a bit changed. But I cannot help remaking that all the changes leaded to make the application better. Implementations described in this dissertation agree with the current software state.

During implementation of particulars modules came to improvement of some features in comparison with the original submission. First of all were implemented user defined templates, to let programmers implement their own layout of the CMS modules. The original concept allowed changing the module layout only using the CSS styles. In this scope is the dissertation a good material for programmers who wants to develop their own template controls. All books I kept at disposition were not able to help me. The source codes of the application are richly commented and contain a lot of examples of using templates including “events bubbling” etc. I sought to use all features of the object oriented programming such inheriting, polymorphism, encapsulation, events etc. I take the application is good example of maximum using of the advantages of the ASP.NET 2.0 technology.

SEZNAM POUŽITÉ LITERATURY

[1] **Castagnetto, Jesus, a další.** *PHP Programujeme profesionálně.* [překl.] Ludvík Roubíček. 2. vyd. Praha : Computer Press, 2002. str. 656. ISBN 80-7226-310-2.

[2] **Crowley, James.** Dynamically loading an IBindableTemplate. *Developer fusion.* [Online] Developer Fusion Ltd, 16. 12 2006. [Citace: 10. 5 2007.] http://www.developerfusion.co.uk/show/4721.

[3] **Evjen, Bill, Hanselman, Scott a Muhamad, Farhan.** *ASP.NET 2.0 Programujeme profesionálně.* [překl.] Karel Voráček. 1. vyd. Brno : Computer Press, a.s., 2006. str. 1224. ISBN 80-251-1286-1.

[4] **Guthrie, Scott.** Atlas Project . *ScottGu's Blog.* [Online] 28. 6 2005. [Citace: 30. 4 2007.] http://weblogs.asp.net/scottgu/archive/2005/06/28/416185.aspx.

[5] **Holzner, Steven.** *JavaScript profesionálně.* [překl.] Vojtěch Krmíček, Ing. Aleš Thiemel Jan Gregor. 1. vydání, Praha : Mobil Media, a.s., 2003. str. 1071. ISBN-80-86593-40-1.

[6] **Kačmář, Dalibor.** *Web cast - Visual Studio 2005 a ASP.NET 2.0 díl 1.* [Webcast] místo neznámé : Microsoft s.r.o., 28. 4 2007.

[7] —. *Co je nového ve Visual Studiu 2005 a ASP.NET 2.0, část II.* [Webcast] místo neznámé : Microsoft s.r.o., 3. 11 2005.

[8] **Khanine, Dmitri.** Tuning Up ADO.NET Connection Pooling in ASP.NET Applications. *http://www.15seconds.com.* [Online] Jupitermedia Corporation. [Citace: 19. 5 2007.] http://www.15seconds.com/issue/040830.htm.

[9] **Malik, Sahil.** A Global SqlConnection Instance - BAD BAD BAD !! *http://codebetter.com/blogs/sahil.malik.* [Online] 30. 1 2005. [Citace: 19. 5 2007.] http://codebetter.com/blogs/sahil.malik/archive/2005/01/30/48948.aspx.

[10] **Nagel, Christian, a další.** *C# 2005 Programujeme profesionálně.* [překl.] Petr Dokoupil Jakub Mikulaštík. 1. vyd. Brno : Computer Press, a.s., 2006. str. 1398. ISBN 80-251-1181-4.

[11] **Prosise, Jeff.** *Programování v Microsoft .NET Webové aplikace.* [překl.] Karel Voráček. 1. vydání, Brno : Computer Press, 2003. str. 712. ISBN-80-7226-879-1.

[12] **Sharp, John a Jagger, Jon.** *Microsoft Visual C# .NET - krok za krokem.* Brno : Mobil Media, a.s., 2002. str. 654. ISBN 80-86593-27-4.

[13] **Suchý, Jakub.** Redakční systém Drupal. *Drupal.cz.* [Online] [Citace: 30. 4 2007.] http://www.drupal.cz/.

[14] CMS. *Wikipedie otevřená encyklopedie.* [Online] 10. 5 2007. [Citace: 14. 5 2007.] http://cs.wikipedia.org/wiki/CMS.

[15] MSDN library. [Online] Microsoft. [Citace: 1. 5 2007.] http://msdn.microsoft.com/library/.

[16] Redakční systém MultiCMS. *MultiCMS.NET.* [Online] Smartware s.r.o. [Citace: 30. 4 2007.] http://www.multicms.net/.

[17] Redakční systém Vizus CMS. *VIZUS Internet partner.* [Online] VIZUS. [Citace: 30. 5 2007.] http://www.vizus.cz/redakcni-system-vizus-cms.html.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AJAX	Asynchronous JavaScript and XML; technologie pro vývoj interaktivních aplikací pro webový prohlížeč
ASP	Active Server Pages; technologie společnosti Microsoft pro vývoj webových stránek
BCL	Base Class Library; knihovna základních tříd platformy .NET
CLR	Common Language Runtime; společné běhové prostředí pro aplikace na platformě .NET
CRUD	Označení pro operace vytváření, získávání, editace nebo mazání dat
CSS	Cascading style sheets; jazyk pro předepsání vzhledu dokumentů psaných pomocí značkových jazyků
DHTML	Dynamic HTML; technologie pro vývoj interaktivních webových stránek
DOM	Document Object Model; objektový model dokumentu – objektově orientovaná reprezentace XML nebo HTML dokumentu
GUI	Graphical User Interface – grafické uživatelské rozhraní
HTML	HyperText Markup Language; značkový jazyk pro tvorbu webových stránek
HTTP	HyperText Transfer Protokol; protokol pro přenos objektů libovolného typu mezi webovým serverem a prohlížečem
HTTPS	Zabezpečený protokol http
IDE	Integrated Development Environment; integrované vývojové prostředí
IIS	Internetová Informační Služba; webový server společnosti Microsoft
RSS	Formát pro zasílání informací o často se měnícím digitálním obsahu
SEO	Search Engine Optimalization; optimalizace pro webové vyhledávače
WYSIWYG	What You See Is What You Get; označení pro editory, kde editační režim odpovídá výsledné grafické podobě dokumentu
XHTML	Extensible HyperText Markup Language; technologie HTML dodržující syntaxi XML
XML	eXtensible Markup Language; rozšiřitelný značkový jazyk; umožňuje vytváření konkrétních značkových jazyků
XMLHTTP	Soubor API funkcí, které umožňují klientským programovacím jazykům přenášet XML data ze serveru a zpět; je důležitou součástí technologie AJAX

SEZNAM OBRÁZKŮ

Obrázek 1 – Zdrojový kód webového formuláře (WebForm) 17
Obrázek 2 – Schéma architektury CMS se společným GUI 26
Obrázek 3 – Schémata architektur CMS s odděleným GUI 28
Obrázek 4 – Ukázka uživatelského rozhraní redakčního systému MultiCMS 34
Obrázek 5 – Schéma vrstvené architektury CMS 41
Obrázek 6 – Struktura tabulky cis_system 43
Obrázek 7 – Databázová struktura pro membership a user roles 44
Obrázek 8 – Databázová struktura pro oprávnění 45
Obrázek 9 – Databázová struktura pro články 47
Obrázek 10 – Struktura pro dokumentový systém 49
Obrázek 11 – Struktura projektů a složek v solution Getmore.CMS 50
Obrázek 12 – Struktura solution Getmore.CMS 53
Obrázek 13 – Ukázka použití direktivy „Register“ v ASPX souboru 54
Obrázek 14 – Sofistikovaná inteligentní komponenta PagingNavigator 62
Obrázek 15 – Ukázka GUI – seznam článků pomocí výchozí šablony 62
Obrázek 16 – Ukázka GUI – detail článku realizovaný pomocí třídy ArticleDetail 64
Obrázek 17 – Ukázka GUI – prvek ArticleEditableDetail 65

SEZNAM TABULEK

Tabulka 1 – Porovnání CMS 38