VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

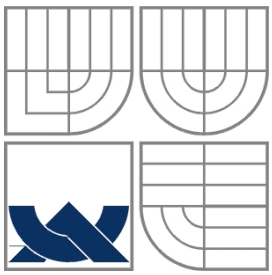
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

CMS FRAMEWORK

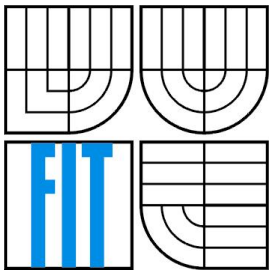
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE JIŘÍ TOMEK
AUTHOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

CMS FRAMEWORK

CMS FRAMEWORK

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

JIŘÍ TOMEK

VEDOUCÍ PRÁCE
SUPERVISOR

ING. PAVEL JURKA

BRNO 2007

Zadání bakalářské práce/5838/2006/xtomek04

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2006/2007

Zadání bakalářské práce

Řešitel: **Tomek Jiří**
Obor: Informační technologie
Téma: **CMS framework**
Kategorie: Databáze

Pokyny:

1. Seznamte se s problematikou tvorby systémů pro správu obsahu (CMS).
2. Analyzujte požadavky na framework podporující tvorbu těchto systémů.
3. Navrhněte tento systém pomocí UML.
4. Implementujte navržený systém.
5. Zhodnoťte dosažené výsledky.

Literatura:

- Dle pokynů vedoucího

Při obhajobě semestrální části projektu je požadováno:

1. Seznamte se s problematikou tvorby systémů pro správu obsahu (CMS).
2. Analyzujte požadavky kladené na tento systém.
3. Navrhněte tento systém pomocí UML.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Jurka Pavel, Ing.**, UITS FIT VUT
Datum zadání: 1. listopadu 2006
Datum odevzdání: 15. května 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Jiří Tomek**
Id studenta: 84128
Bytem: Pod Cestou 13, 628 00 Brno
Narozen: 16. 02. 1985, Brno
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: CMS framework
Vedoucí/školitel VŠKP: Jurka Pavel, Ing.
Ústav: Ústav inteligentních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☒ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

.....	.....
Nabyvatel	Autor

Abstrakt

Tato práce se zabývá návrhem a popisem frameworku pro tvorbu internetových redakčních systémů. Framework je implementován za použití jazyka PHP5 a databáze MySQL 4.1. Má za cíl zjednodušit vytváření rozmanitých webových prezentací za použití jednoduchých principů, které se v podobných projektech objevují. Blíže se zde seznámíme s fází návrhu celého systému za použití UML a rozebereme volbu použitých technologií. Dále jsou zde popsány principy funkce celého systému, jeho struktura a použité technologie. Pro praktické použití obsahuje tato práce i ukázkovou implementaci rozšíření systému.

Klíčová slova

redakční systém, cms, framework, php, mysql, Smarty, web, UML, internet, databáze

Abstract

This thesis is concerned with design and description of framework for internet content management systems. Framework is implemented using PHP5 programming language and MySQL 4.1 database engine. It is aimed to simplify creation of various internet presentations using simple principles that are repeated in similar projects. We closely look on concept of framework using UML and we analyse choice of used technologies. There are principles of function of whole system, its' structure and used technologies described further in text. This work also contains sample implementation of framework module for practical use.

Keywords

content management system, cms, framework, php, mysql, Smarty, web, internet, database

Citace

Jiří Tomek: CMS framework, bakalářská práce, Brno, FIT VUT v Brně, 2007

CMS framework

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Pavla Jurky
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jiří Tomek
3.5.2007

Poděkování

Chtěl bych poděkovat Ing. Pavlu Jurkovi za jeho pomoc při psaní této práce.

Obsah

1 Úvod

Význam internetu jako média je v dnešní době větší než kdy dříve. Pokud chce dnes jakákoliv firma prezentovat svoji činnost, je internetová prezentace jedním z nejúčinnějších prostředků, které jí mohou tuto službu poskytnout. Zároveň s rozšiřováním internetu a počtu jeho uživatelů ovšem také rostou požadavky na formu prezentace. Dnes již nestačí jedna statická stránka s adresou a předmětem činnosti firmy. Zákazník očekává, že na internetu nalezne mnohdy vyčerpávající popis firmy se spoustou informací, katalogů, obrázků a dalších dat.

Zabývám se již nějakou dobu tvorbou internetových prezentací a všiml jsem si, že velká většina jich má určité společné prvky a rysy. Například se skládají z jednotlivých stránek, které jsou řazeny v jakési stromové struktuře. Často je také požadavkem zákazníka určitá forma registrací, obsahu pro registrované klienty a podobně. Napadlo mne tedy, vytvořit si jakýsi základní soubor tříd a funkcí pro zjednodušení tvorby těchto internetových prezentací – framework pro redakční systémy.

Tato publikace má sloužit jako popis uvedeného frameworku, má za cíl vysvětlit jeho funkčnost, principy použité při jeho tvorbě a také poskytuje návod na používání a rozšiřování tohoto systému. Vzhledem k tomu, že framework je převážně praktické dílo, ve kterém jde hlavně o samotnou implementaci a ne tolik o teoretické pozadí, je i tato publikace orientována spíše prakticky a nezabývá se příliš teorií.

V první části se blíže seznámíme s pojmem „redakční systém“. Druhá část se zabývá procesem návrhu frameworku a ve stručnosti se zmíníme o technologiích použitých při tvorbě systému. Třetí část potom obsahuje popis samotného systému z hlediska implementace a použitých principů, rozdělený podle jednotlivých kategorií. Ve čtvrté části se nachází popis modulů a principů pro rozšiřování funkcí frameworku.

2 Co je to redakční systém?

Pokud chceme hovořit o frameworku pro tvorbu redakčních systémů, je nezbytné vysvětlit, co vlastně pojem „redakční systém“ představuje.

Redakčním systémem (anglicky content management system - CMS) se rozumí aplikace, která slouží ke správě obsahu, zpravidla internetových stránek. Tímto obsahem mohou být články, obrázky, soubory a další rozmanitá data. Součástí redakčního systému bývá často i určitá forma správy uživatelů.

Hlavní úlohou redakčního systému je poskytování přehledného rozhraní pro správu požadovaného obsahu. Toto rozhraní by mělo poskytovat všechny funkce, které jsou potřeba k manipulaci se spravovanými daty a celý systém by měl být přehledný a spolehlivý.

Výstupem redakčního systému jsou ve většině případů stránky, které si může uživatel prohlížet prostřednictvím webového prohlížeče.

3 Návrh

V této kapitole se blíže podíváme na fázi návrhu popisovaného frameworku, proč byly zvoleny konkrétní postupy a technologie.

Na základě studia již existujících internetových prezentací vyplynulo několik bodů, které se prakticky bez výjimky objevují ve všech internetových redakčních systémech.

- systém se skládá z jednotlivých stránek, které jsou většinou uspořádány ve stromové struktuře
- téměř vždy se v systému vyskytují statické stránky s textem
- často obsahuje systém i dynamické části jako například aktuality, knihu návštěv a podobně
- je-li možná registrace uživatelů, obsahuje systém i části přístupné pouze těmto registrovaným uživatelům
- rozhraní redakčního systému je děláno s ohledem na přehlednost a jednoduchost použití

3.1 Použité technologie

Vzhledem k tomu, že se jedná o framework pro internetové redakční systémy, bylo nutno podle toho zvolit odpovídající implementační prostředí.

3.1.1 Programová část

Pro implementaci programové části systému byly brány v úvahu programovací jazyky PHP, ASP.NET, Perl a Python.

Jazyky Perl a Python byly zamítnuty hned zpočátku, hlavně kvůli jejich menší podpoře ze strany poskytovatelů hostingu. Pokud má být systém široce použitelný, je podpora použitých technologií důležitým hlediskem.

Zbývá tedy dvojice PHP a ASP.NET. Obě technologie byly srovnány z několika pohledů.

3.1.1.1 Dostupnost

Pokud se podíváme na dostupnost hostingů s podporou PHP a ASP.NET, zjistíme, že PHP je rozšířeno více a hosting s podporou PHP je ve většině případů i levnější. To je dáno také tím, že PHP běží většinou na Linuxu, který je zdarma. Oproti tomu ASP.NET běží na MS Windows a jako databázi většinou používá MSSQL. V tomto bodě je tedy vhodnější PHP.

3.1.1.2 Rychlost

Skripty v jazyce PHP jsou ve většině případů rychlejší než v ASP.NET. Jedná se sice o interpretovaný jazyk, ale v dnešní době obsahují hostingové servery výkonné cache, takže neustálá interpretace skriptu není často nutná.

3.1.1.3 Složitost

Jazyk PHP je oproti ASP.NET velice jednoduchý na naučení. ASP.NET může používat jako svůj jazyk například VB, C#, J#, ale všechny tyto jazyky jsou komplikovanější než samotné PHP.

Nakonec byl zvolen jazyk PHP ve verzi 5.0 nebo vyšší pro jeho nejširší podporu, relativní jednoduchost, přehlednost a rychlost.

3.1.2 Databázová část

Vzhledem k účelu redakčního systému je nutné použít některou z technologií pro ukládání dat. V tomto směru je jednoznačně nejvhodnější databáze pro ukládání textových dat, v kombinaci se soubory pro ukládání obrázků a dalších multimediálních dat.

3.1.2.1 Engine

Jako databázový engine pro běh systému byla zvolena databáze MySQL verze 4.1 nebo vyšší. Hlavní výhodou MySQL oproti většině jiných databází, jako například Oracle, je její volná dostupnost. Pokud se tedy omezíme na volně dostupné databáze, nabízí se nám databáze PostgreSQL. Provedeme-li ovšem srovnání obou databází, vyjde lépe právě MySQL protože:

- MySQL je ve většině případů rychlejší než PostgreSQL

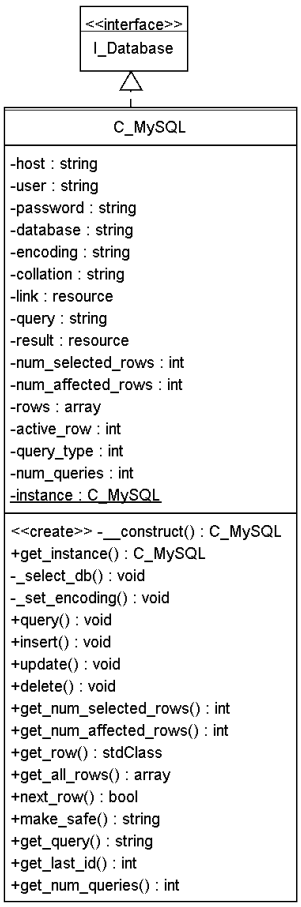
- MySQL má lepší podporu pro více platforem
- MySQL lépe zvládá velký počet připojení a je tedy vhodnější pro webové aplikace
- návrh databáze pro MySQL je jednodušší

3.1.2.2 Přístup k databázi

PHP poskytuje celou řadu funkcí pro práci s MySQL databází. Pro zjednodušení práce s databází a vytvoření jednotného rozhraní byla pro framework navržena třída, která práci s databází zapouzdřuje. Odpadne tak například složité zpracování výsledků pomocí několika funkcí.

Aby bylo v budoucnu možné změnit databázový engine za jiný, bude framework obsahovat definici jednotného rozhraní, které musí třída pro práci s databází implementovat.

Nejčastějšími dotazy na databázi jsou dotazy typu SELECT, INSERT, UPDATE a DELETE. Bylo by tedy vhodné, aby třída obsahovala přímo metody pro tyto dotazy. U dotazů INSERT a UPDATE se budou data předávat jako dvojice sloupec-hodnota. Pro dotaz typu SELECT není, vzhledem k jeho možné složitosti, navržení speciální metody reálné. Bude se tedy používat standardní metoda pro obecné dotazy.



Obrázek 1: Class diagram databázové části

3.1.3 Prezentační část

I když samotný framework, který je předmětem této práce, neposkytuje přímo výstup pro běžné uživatele, obsahuje prostředky pro jeho podporu. Pro oddělení aplikační logiky od prezentační části je vhodné použít některý existující šablonový systém.

V tomto směru se ukázal jako nejlepší šablonový systém Smarty. Tento sytém je velice rychlý, je podporován přímo tvůrci PHP a má mnoho pokročilých funkcí.

3.2 Základní struktura systému

3.2.1 Obsahová část

První fáze návrhu systému se týkala struktury a jednotlivých prvků, které budou celý systém tvořit.

Vzhledem k požadované všestrannosti frameworku bylo nutné, aby jeho struktura obsahovala co nejobecnější prvky, které bude možné vhodně použít v libovolném modulu pro rozšíření

frameworku. Tyto prvky by měly obsahovat pouze základní vlastnosti, které budou potřeba prakticky vždy. Další vlastnosti mohou být přidány dalším rozšiřováním.

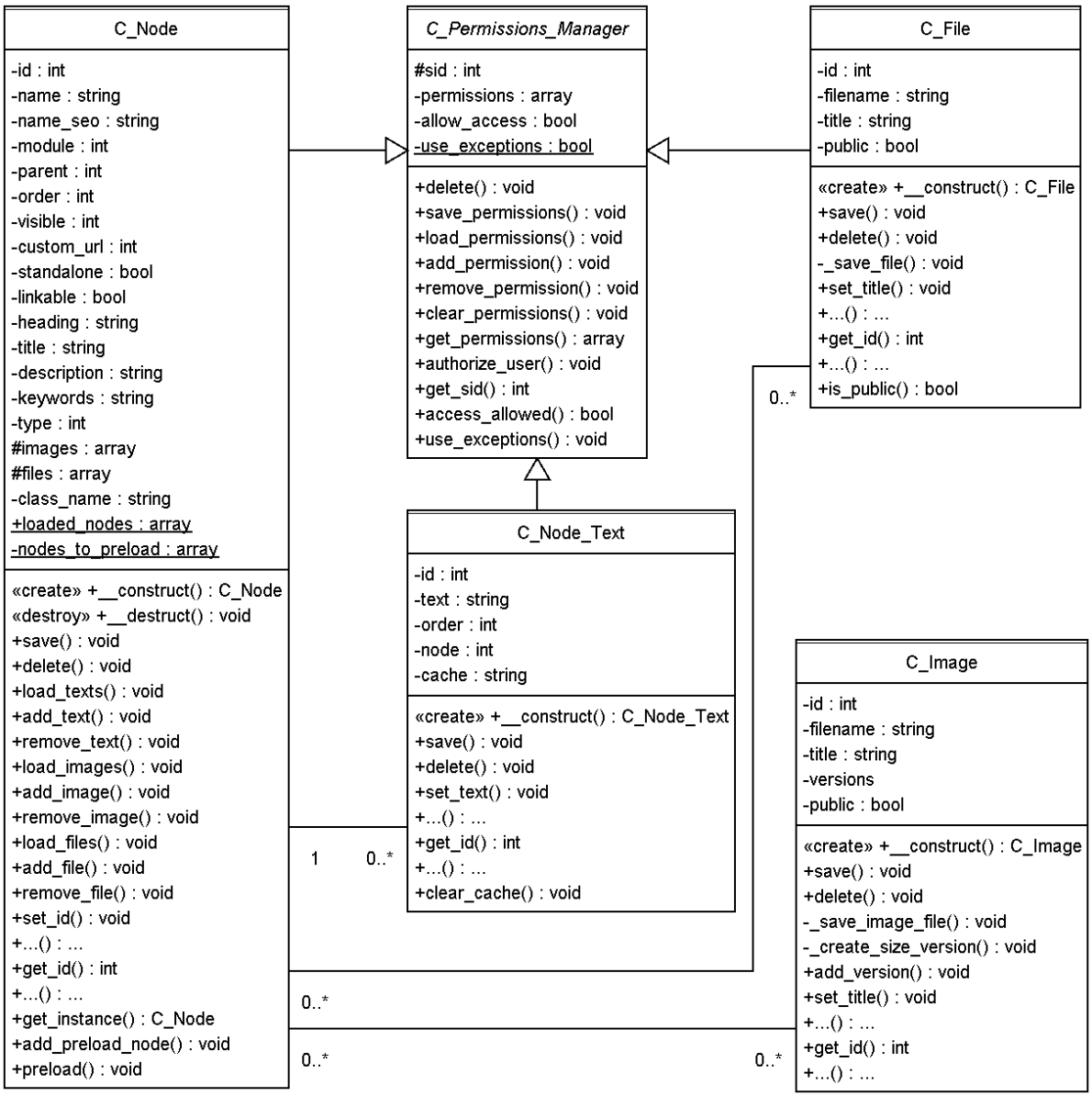
Jako vhodným základním prvkem obsahu systému se ukázal prvek použitý například v systému Drupal. Tímto prvkem je tzv. uzel.

Uzel musí být možno jednoznačně identifikovat, musí obsahovat několik základních vlastností pro určení jeho chování a hlavně je třeba mít možnost jednotlivé uzly uspořádat. Mimo několika základních vlastností by uzel neměl obsahovat žádné zbytečné prvky, které nejsou nezbytně nutné pro jeho předpokládané použití. Aby bylo možno tohoto dosáhnout, počítá se v návrhu s tím, že veškeré podružné vlastnosti uzlu se z něj vyčlení a budou reprezentovány zvláštními objekty.

Samotný uzel sice vytváří základ libovolného obsahového prvku, ovšem pro plnou funkci je třeba připojit některá další data. Součástí každé prezentace na internetu jsou i texty, v návrhu ovšem uzel žádné vlastnosti pro text neobsahuje. Proto je třeba vytvořit další prvek struktury, který bude sloužit pro práci s textem a bude nějakým způsobem svázán s uzlem.

Pro rozšíření možností obsahu uzlů byly přidány také prvky pro reprezentaci obrázků a souborů. Tím se pokryly základní požadavky na obsahovou část systému.

Protože jedním z požadavků na systém je možnost kontrolovat přístup k jednotlivým objektům, byla navržena třída, která tuto funkčnost poskytne. Díky tomu, že třída je vyčleněna samostatně, je možné ji použít podle potřeby s jakýmkoliv objektem. Uzel bude tedy od této třídy odvozen a tím bude možno kontrolovat přístup k prvkům obsahové části frameworku.



Obrázek 2: Class diagram obsahové části frameworku

3.2.2 Uživatelská část

Druhá fáze návrhu se týkala uživatelů výsledného systému, jejich organizaci a správě.

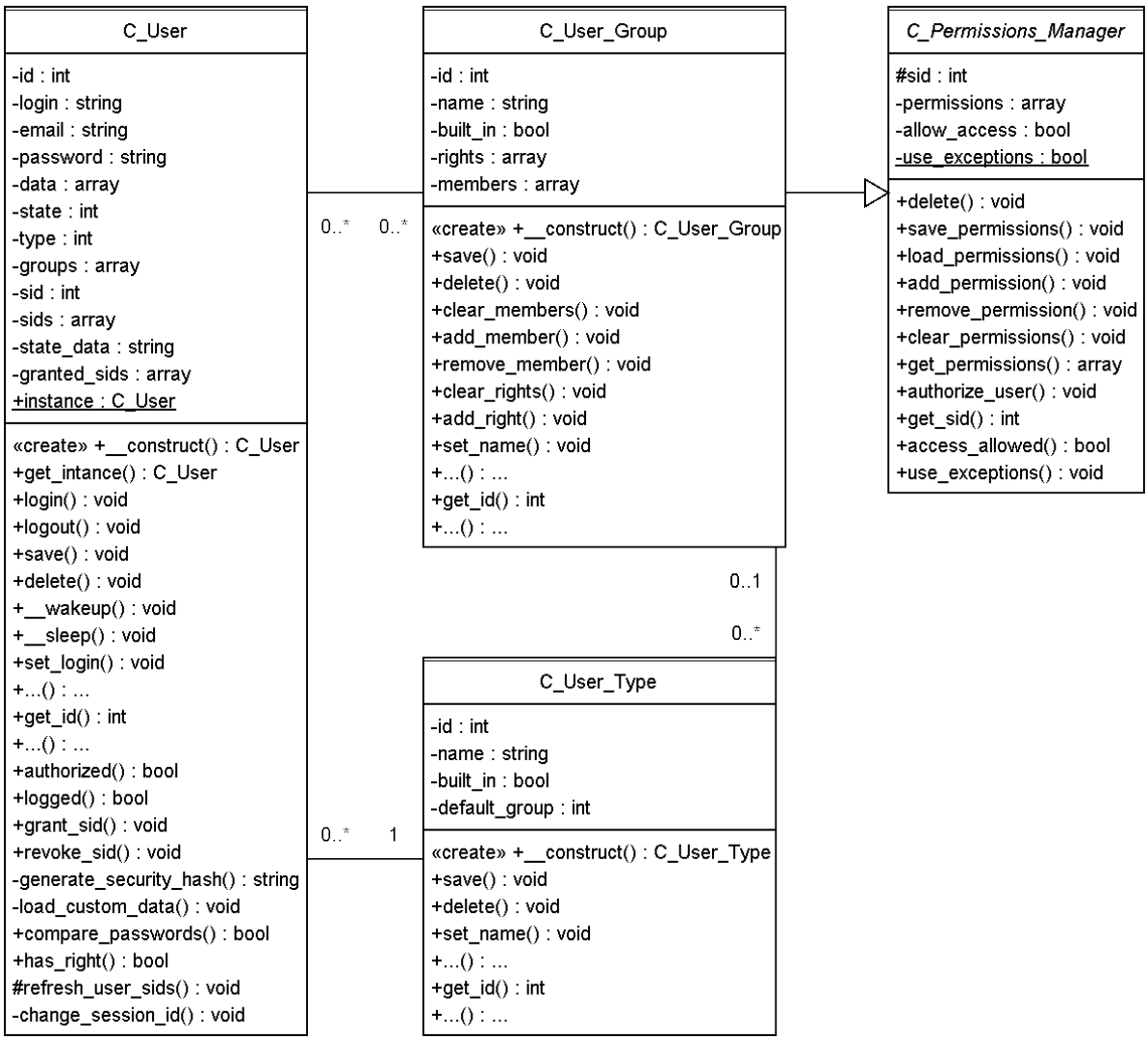
Aby bylo možno dosáhnout co největší volnosti ve správě uživatelů, postupovalo se v návrhu podobně jako u obsahové části. Hlavní myšlenkou bylo obsáhnout u každého uživatele pouze nejzákladnější atributy, ale zároveň poskytnout možnost pro přidávání dalších dat.

První sadou údajů které musí uživatel mít jsou údaje nezbytné k případnému přihlášení do systému.

Dále je třeba, aby bylo možné kontrolovat u každého uživatele jeho oprávnění a přístup k objektům. V této oblasti je systém částečně inspirován operačními systémy MS Windows. Každý uživatel bude mít jedinečný identifikátor, který se bude používat v procesu kontroly oprávnění. Pro zjednodušení správy bude možno vytvářet skupiny uživatelů, které budou v procesu kontroly oprávnění vystupovat jako jeden objekt.

U většiny redakčních systémů a internetových aplikací se objevuje potřeba členit uživatele nejen podle oprávnění, ale i podle příslušnosti k určité skupině. Jako příklad mohou sloužit internetové obchody, kde se objevují zákazníci a správci. Bylo tedy navrženo další dělení uživatelů - dělení podle typu.

Každý typ uživatele by měl mít možnost definovat skupinu rozšiřujících vlastností, které se budou u uživatele evidovat.

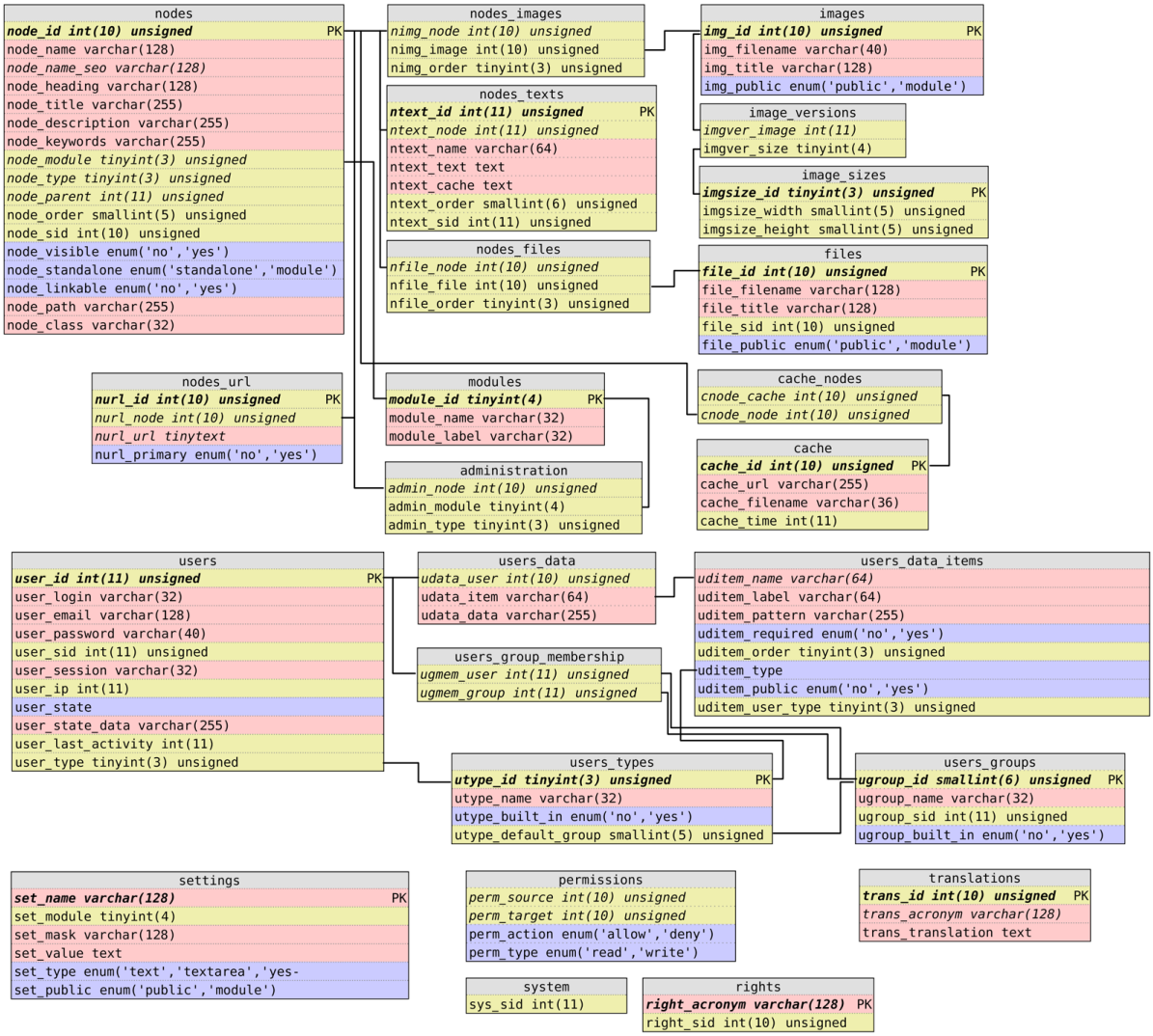


Obrázek 3: Class diagram uživatelské části frameworku

3.3 Návrh databázové struktury

Po zpracování návrhu celkové struktury systému přichází na řadu návrh struktury databáze. Protože jednotlivé prvky struktury frameworku budou tvořeny samostatnými objekty, nabízí se řešení ukládat jednotlivé typy objektů do vlastních tabulek.

Stačí tedy vzít vlastnosti konkrétního objektu a vytvořit na jejich základě tabulku pro jeho uložení. Takto byla tedy navržena celková struktura databáze, která ovšem neměla ideální podobu z hlediska normy. Po normalizaci má výsledná struktura následující podobu.



Obrázek 4: Schéma struktury databáze

3.4 Další prvky systému

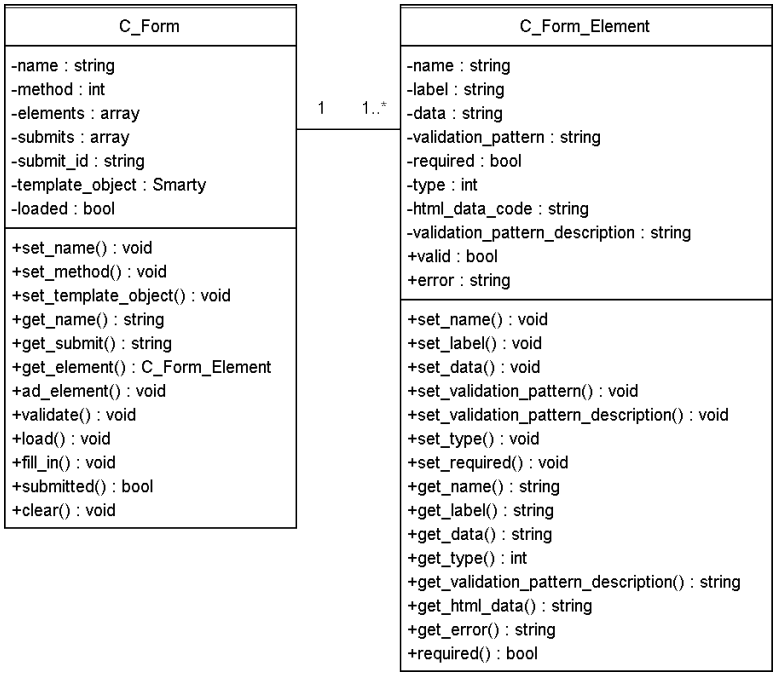
Mimo výše popsanou základní strukturu systému by měl systém obsahovat také další části, které usnadní práci s ním.

3.4.1 Formuláře

Prakticky každý redakční systém obsahuje formuláře pro zadávání a úpravu dat. Bez nějakého komplexního řešení je zpracování a validace formulářových dat většinou těžkopádná a komplikovaná a vyžaduje složitý programový kód. Proto bude systém obsahovat prostředky pro definici formulářů, jejich automatické zpracování a validaci.

Každý formulář se skládá z jednotlivých elementů jako jsou tlačítka, textová pole, přepínače. Pro reprezentaci těchto elementů bude systém obsahovat třídu, která bude definovat typ elementu, povolené hodnoty, zda je nutno zadat do elementu hodnotu a podobně. Zároveň si na vyžádání třída automaticky načte data zadaná do elementu a v případě potřeby je i ověří. Neplatnost dat je vhodné reprezentovat vyvoláním specifické výjimky.

Jednotlivé elementy formuláře budou sdruženy do většího celku, který bude reprezentovat samotný formulář. Tomuto celku bude možno nastavit různé vlastnosti, jako například název a metodu pro odesílání dat.



Obrázek 5: Class diagram formuláře

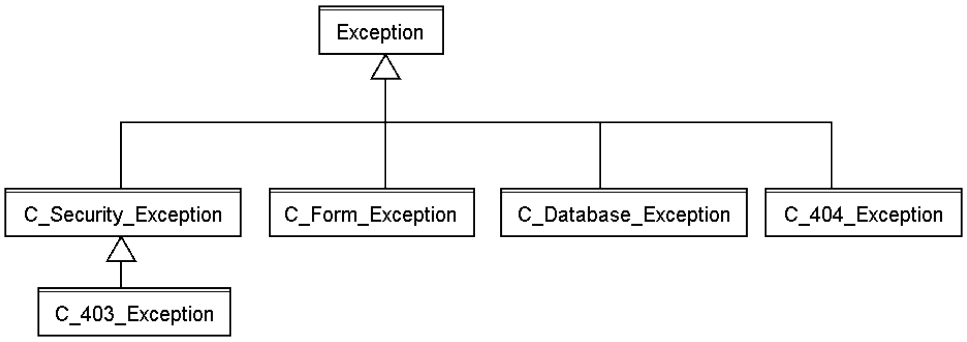
3.4.2 Zpracování chybových stavů

Pro zpracování a signalizaci chybových stavů lze s úspěchem využít novinky v PHP5, kterou jsou výjimky. Použití výjimek namísto návratových hodnot funkcí zjednoduší kód a umožní reagovat na chyby až ve chvíli jejich skutečného výskytu. Většina operací prováděných ve frameworku bude reagovat na chybový stav právě vyvoláním výjimky, která bude poté podle potřeby zachycena a zpracována.

Pro pokrytí možných stavů bude systém obsahovat následující výjimky:

- C_404_Exception - objekt nenalezen
- C_403_Exception – odepřen přístup k objektu
- C_Security_Exception – obecná chyba zabezpečení
- C_Database_Exception – chyba při práci s databází
- C_Form_Exception – chyba při zpracování formuláře – neplatná data

Tyto výše popsané výjimky by měly pokrýt všechny předpokládané chybové stavy.



Obrázek 6: Diagram hierarchie výjimek

3.4.3 Správa nastavení

Každý systém potřebuje nějakým způsobem spravovat své nastavení. V tomto případě jsou dvě možnosti, jak nastavení ukládat a načítat. Položky nastavení mohou být uloženy v souboru na disku, nebo mohou být v databázi.

Pro framework byla zvolena kombinace těchto možností. Nastavení týkající se například přístupových údajů k databázi, definice tabulek, cest a podobně, budou ukládány v souboru jako definice konstant. Tyto konstanty se potom budou používat v kódu namísto konkrétních hodnot.

Zbylá nastavení, jako například emailová adresa administrátora a další údaje, které je třeba měnit jednoduše například v administraci, budou ukládána do databáze a v administraci frameworku se pro ně bude generovat rozhraní pro jejich úpravu. Ke správě těchto nastavení bude systém obsahovat třídu, která bude poskytovat potřebné metody.

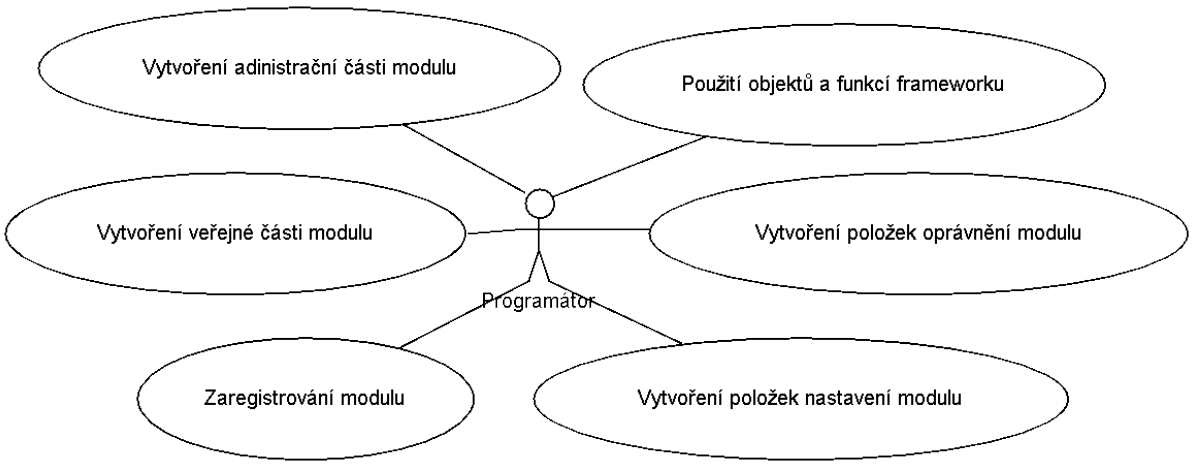
3.5 Použití frameworku

3.5.1 Z hlediska programátora

Jedním z pohledů na užívání frameworku je pohled z hlediska programátora. Programátor vyžaduje od frameworku podporu pro tvorbu modulů, kterými lze rozšířit jeho funkčnost.

Při tvorbě modulu potřebuje programátor nástroje pro vytvoření dvou oddělených částí modulu – administrační a veřejné části. Mimo to je nutné začlenit modul do systému pomocí jeho registrace a vytvoření položek jako jsou nastavení nebo oprávnění.

Při samotné tvorbě očekává programátor dostupnost vnitřních funkcí a tříd frameworku pro usnadnění některých úkonů.



Obrázek 7: Use-case diagram programátora

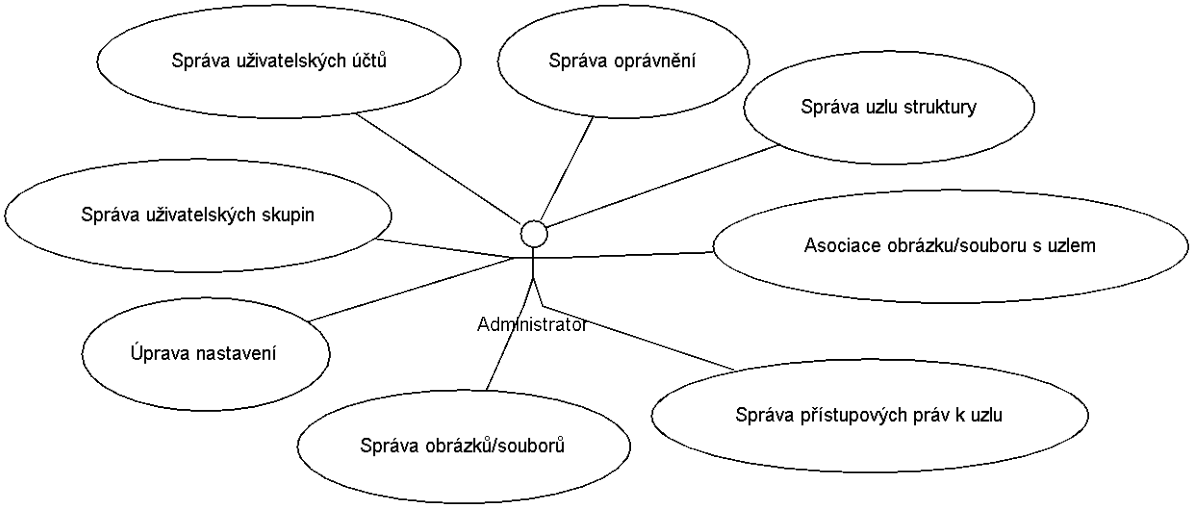
3.5.2 Z hlediska správce

Správce výsledného systému již nezajímají detaily tvorby modulů, ale naopak vyžaduje určité funkce týkající se vytvořeného systému. Nelze předvídat, jaké funkce budou mít případné moduly, ale jádro frameworku by mělo poskytovat minimálně sadu funkcí a nástrojů pro obecnou správu systému.

První z potřebných funkcí je správa celkové struktury systému. Pokud jsou objekty reprezentovány uzly, je možno je i centrálně spravovat. S tím souvisí také možnost kontroly přístupu k objektům, která se bude dít právě na úrovni jednotlivých uzlů.

Další důležitou součástí systému je správa uživatelů. Správce musí mít možnost spravovat alespoň základní uživatelské účty nezávisle na používaných modulech. Spolu s uživatelskými účty by měl správce rovnou spravovat i uživatelské skupiny a typy uživatelů.

Konečně by měla být možnost centrálně spravovat ukládané obrázky a soubory nezávisle na jejich použití v modulech.



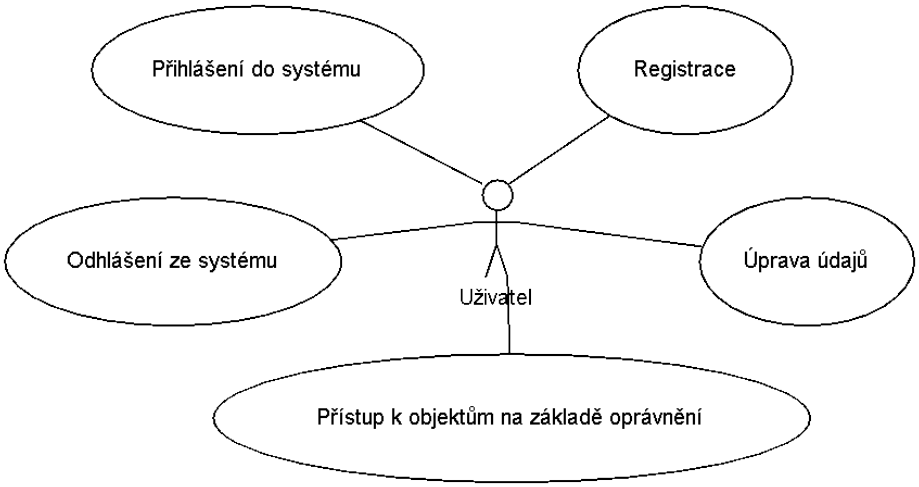
Obrázek 8: Use-case diagram administrátora

3.5.3 Z hlediska uživatele

Možnosti uživatele systému závisí do značné míry na použitých modulech a nastavení. Pokud pomineme jakékoliv moduly, jsou dvě možnosti.

Buďto bude registrace uživatelů zakázána a tím pádem se běžný uživatel dostane pouze k veřejným částem systému.

Nebo bude registrace uživatelů povolena, a uživatel se bude moci zaregistrovat. Po registraci se může uživatel přihlásit do systému a na základě přidělených oprávnění přistupovat k objektům a provádět povolené akce. Zároveň by měl mít uživatel možnost upravit alespoň některé svoje údaje, jako například změnit heslo. Nakonec by se měl uživatel ze systému odhlásit.



Obrázek 9: Use-case diagram uživatele

4 Popis frameworku

4.1 Struktura frameworku

Framework je složen z několika hlavních tříd, které reprezentují hlavní objekty z návrhu frameworku. Tato skupina je doplněna několika dalšími podpůrnými třídami, které zapouzdřují nejrůznější funkce jako je formátování řetězců, přístup k databázi a podobně.

4.1.1 Vstupní skript

Vstupním bodem systému založeného na frameworku je takzvaný „vstupní skript“. Tento skript je až na výjimky jediným místem, kudy se může uživatel do systému dostat. Typický tvar adresy URL používané v systému vypadá následovně:

`http://www.domena.cz/vstupni_skript/stranka/podstranka`

Vstupním skriptem je jeden PHP soubor, který zpracuje vloženou adresu a podle jejího tvaru zobrazí požadovanou stránku. Tento přístup zajišťuje dnes tolik oblíbený tvar odkazů pro vyhledávače, protože odkaz vypadá, jakoby ukazoval cestu v adresářové struktuře.

Zadáním URL ve výše uvedeném tvaru je zavolán vstupní skript, který obdrží v globální proměnné část „/stranka/podstranka“. Tento řetězec rozdělí na jednotlivé položky a získá tak cestu k cílovému uzlu.

Díky tomu, že celá akce se odehrává za použití PHP, není potřeba pro vytvoření hezkých URL žádná podpora na serveru, jako například mod_rewrite u serveru Apache.

4.1.2 Rozhraní databáze

Jako rozhraní pro přístup k databázi obsahuje systém třídu **C_MySQL**, která obsahuje metody potřebné ke spolupráci s databází.

Aby bylo zajištěno, že v celém systému bude vytvořena pouze jedna instance této třídy, byl použit návrhový vzor Singleton. Instanci třídy tak lze získat voláním statické metody `get_instance()` nebo použitím funkce `get_db_class_instance()`, která teprve zavolá statickou metodu třídy. V systému se používá pouze funkce `get_db_class_instance()`. Díky tomu je možné v případě potřeby nahradit třídu C_MySQL jinou třídou pouze změnou těla této funkce.

Pro zjednodušení tvorby jiných tříd pro přístup k databázi, například z důvodu změny databázového systému, je ve frameworku třída rozhraní s názvem **I_Database**. Při tvorbě nové třídy je nutné odvodit ji od tohoto rozhraní pro zajištění správné funkčnosti.

4.1.3 Princip činnosti

Jak bylo řečeno výše, vstupem do systému je zadání URL v daném formátu. Tím dojde na straně serveru k zavolání vstupního skriptu.

Na začátku tohoto skriptu je načten soubor s konfiguračními konstantami, které definují například cesty v systému, názvy databázových tabulek a podobně.

V dalším kroku je zavolán skript, který vykoná nejrůznější inicializace, vytvoří připojení k databázi a načte rozšiřující moduly.

V tuto chvíli je již systém ve stavu, kdy je možno přistoupit k samotnému načtení požadovaného obsahu. Vezme se tedy zadaná adresa a postupně je po prvcích zpracována. Jsou vyhledávány jednotlivé uzly a pokud skript narazí na uzel, který patří některému rozšiřujícímu modulu, předá řízení tomuto modulu.

Pokud systém nemůže nalézt odpovídající uzel, je vyhozena výjimka a uživatel je přesměrován na stránku s oznámením, že jím zadaná adresa neexistuje.

Nakonec je ještě zavolán skript, který provede poslední úpravu načtených hodnot a zobrazí pomocí šablon finální výstup.

4.2 Objekty systému

4.2.1 Uzly

Základním prvkem celé obsahové části systému je objekt pojmenovaný **uzel**. Tento uzel je v systému reprezentován instancí třídy **C_Node**.

Node
id : int name : string name_seo : string module : int parent : int order : int visible : int custom_url : int standalone : bool linkable : bool heading : string title : string description : string keywords : string type : int

Uzel je reprezentován celočíselným identifikátorem ID, který je v rámci ostatních uzlů jedinečný, bezpečnostním SID (viz. 4.4) a může mít i svůj název. Každý uzel obsahuje ukazatel, který lze použít k navázání na jiný uzel jako na svého rodiče. Díky tomu je možno řadit uzly do stromové struktury. Pro určení pozice uzlu v dané větvi stromu obsahuje uzel pořadí v rámci uzlů ve stejné větvi.

K určení chování konkrétního uzlu slouží několik atributů:

- viditelnost – určuje, zda je uzel viditelný nebo skrytý
- odkazovatelnost – určuje, zda má uzel vlastní jedinečné URL pomocí kterého je na něj možno odkazovat
- samostatnost – určuje, zda může být uzel při průchodu stromem zpracován jádrem systému, nebo zda je třeba přenechat zpracování na příslušném modulu (viz. 5).

Jak již bylo řečeno, samotný systém neobsahuje žádné prostředky pro prezentaci obsahu, ale tuto činnost přenechává modulům. Proto má každý uzel položky pro určení modulu ke kterému náleží a typu pro specifikaci jeho konkrétní funkce v modulu.

Protože se dnes více než kdy jindy hledí na internetu na SEO konkrétních stránek, obsahuje i uzel několik položek, které s tím souvisí. Každému uzlu je možno definovat titulek okna prohlížeče, který se pro daný uzel použije, klíčová slova, která se vloží do tagu <meta> ve stránce, a konečně i popis (description) pro <meta> tag stránky.

Další věcí související se SEO jsou URL stránek. Pokud je uzel odkazovatelný, je jeho URL ve tvaru: http://www.domena.cz/vstupni_skript/prarodic/rodic/uzel. Část „/prarodic/rodic/uzel“ se nazývá **cestoa k uzlu**, protože určuje umístění uzlu ve struktuře, stejně jako je tomu například u souborového systému.

K vytvoření cesty se používá název uzlu převedený do formy použitelné v URL. Odkaz potom tvoří cestu stromem k danému uzlu a pro člověka i pro vyhledávače je jednodušší určit, kde se ve struktuře nacházejí.

Protože generování cesty k uzlu je relativně náročná operace a musela by se provádět při každém dotazu na umístění uzlu ve struktuře, je tato cesta vygenerována pouze jednou a uložena spolu s uzlem. Změní-li se umístění nebo název samotného uzlu nebo některého jeho předka, stane se ovšem cesta neplatnou. Aby se předešlo těmto případům, při každé takové změně jsou vymazány uložené cesty u uzlů, kterých se změna týká, a při prvním následném dotazu na umístění uzlu je cesta vygenerována znovu.

Při návrhu bylo pamatováno i na možnost, že uživatel systému bude z nějakého důvodu potřebovat jinou URL pro daný uzel. Proto je u každého uzlu možnost nadefinovat uživatelské URL, pod kterým potom bude uzel vystupovat. Pokud návštěvník stránek zadá odpovídající URL, je daný uzel podle něj nalezen.

Pokud se podíváme na stránky na internetu, téměř každá stránka má nadpis čí titulek. Proto je možno ukládat k uzlu i položku pro tento nadpis.

4.2.1.1 Vytváření objektů uzlu

Velmi často můžeme narazit na situaci, kdy je jeden konkrétní uzel během jednoho skriptu potřeba několikrát, nevíme ovšem dopředu, kdy to bude a za jakých podmínek. Aby se zabránilo opakovanému načítání stejného uzlu, byl použit návrhový vzor Singleton.

Třída C_Node obsahuje statickou třídu `get_instance($id)`, která vrátí instanci uzlu podle zadaného ID. Při prvním zavolání této metody je uzel uložen do pole a později je již pouze vracen tento načtený uzel.

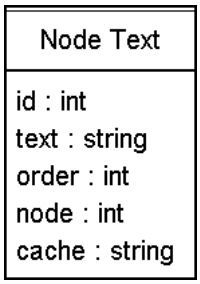
Při tvorbě nejrůznějších modulů (viz. 5) je často potřeba rozšířit funkčnost základního uzlu o některé další funkce. Jako příklad může sloužit třída pro aktualitu, která vznikne odvozením od uzlu a přidáním položek jako například datum. Jádro frameworku ovšem při práci s uzly používá pouze

bázovou třídu C_Node a pokud je tedy jádrem načten uzel reprezentující aktualitu, byl by za normálních okolností pouze instancí C_Node a ne potřebné aktuality. Proto se při prvním uložení uzlu z odvozené třídy automaticky uloží i název této třídy a následné volání metody `get_instance()` rovnou vrací instanci odvozené třídy.

4.2.1.2 Hromadné načítání uzlů

Jelikož je uzel reprezentován instancí třídy, která načítá data uzlu z databáze, je při každém vytvoření této instance provedeno několik dotazů. V případě, že je potřeba načíst několik uzlů hromadně, obsahuje třída **C_Node** i soubor statických metod, které umožní načtení několika uzlů zároveň a tím ušetří dotazy na databázi.

4.2.2 Texty



Samotný uzel ovšem nemá žádnou vlastnost, která by se dala použít pro uložení textu stránky. Je tomu tak proto, že texty mohou nabývat velkého objemu a ukládat je spolu s uzlem snižuje rychlost při práci s ním. Ve většině případů je třeba znát pouze název případně nadpis uzlu, jeho ID, umístění a atributy. Kompletní obsah uzlu se většinou zobrazuje pouze u právě prohlížené stránky a uzlů, které s ní bezprostředně souvisejí.

Texty k uzlům jsou tedy ukládány odděleně. Každý text uzlu je v systému reprezentován instancí třídy **C_Node_Text**.

Text je opět reprezentován celočíselným identifikátorem ID, který je v rámci ostatních textů jedinečný, bezpečnostním SID (viz. 4.4) a dále potom uzlem ke kterému náleží a pořadím vzhledem k textům daného uzlu.

K jednomu uzlu lze tedy přiřadit více textů na různé pozice. Vezmeme-li jako příklad uzlu aktualitu, můžeme jí přiřadit jeden text jako náhled do přehledu aktualit a druhý text jako kompletní text aktuality.

Texty by se neměly vytvářet samostatně, ale měly by se k manipulaci s nimi používat metody uzlů k tomu určené.

4.2.2.1 Automatické formátování

Často potřebnou součástí textů jsou odkazy v rámci stránek. Je samozřejmě možno zadat do textu konkrétní odkaz, ale pokud se změní umístění cílového objektu, stane se odkaz nefunkčním. Proto obsahuje framework systém dynamických odkazů. Pokud uživatel zadá do textu značku **[n X]**, kde X určuje ID cílového uzlu, je z této značky automaticky vygenerován odkaz na daný uzel.

Podobným způsobem funguje i vkládání obrázků (viz. 4.2.3) do textu. Stačí zadat značku **[i X]**, kde X určuje ID obrázku. Pokud se použije značka jak je uvedena výše, je obrázek vložen na střed.

Pro určení pozice je možné použít dvě další verze značky: **[i X <]** pro umístění vlevo a **[i X >]** pro umístění vpravo. Použitá značka je potom nahrazena následujícím blokem:

```
<a class="in_text_image_container"
   href="javascript:void(0);"
   onclick="popup_image('URL_PLNE_VERZE_OBRAZKU');">
<img class="in_text_image_image"
     src="URL_ZMENSENE_VERZE_OBRAZKU" alt="ALTERNATIVNI_TEXT"
     title="NAZEV_OBRAZKU" />
<span>NAZEV_OBRAZKU</span>
</a>
```

Při použití umístění obrázku vlevo je k elementu <a> přidána ještě třída **in_text_image_left**, při umístění vpravo je to potom třída **in_text_image_right**. Díky použití tohoto přístupu je možné pomocí CSS stylů kompletně ovlivnit vzhled a pozici obrázku bez zásahu do kódu systému. Vkládaná JavaScriptová funkce `popup_image()` slouží k zobrazení plné verze obrázku do nového okna a může být implementována libovolně.

Pro zlepšení funkčnosti textu je ve frameworku implementována funkce pro automatický převod emailových adres na odkazy. Pokud uživatel zadá do textu stránky libovolnou emailovou adresu, je tato adresa převedena na odkaz. Velkým problémem volně přístupných emailových adres v textech stránek je ovšem fakt, že je snadno naleznou roboti pro sběr těchto adres za účelem rozesílání nevyžádané pošty. Jako ochranu před tímto sběrem byl zvolen relativně účinný způsob kódování adres. Znaky adresy a odkazu jsou před zobrazením převedeny na desítkové a šestnáctkové entity a díky tomu se adresa stane pro většinu robotů neviditelnou. Protože webové prohlížeče převádějí před zobrazením entity zpět na znaky, je pro návštěvníka stránek adresa stále stejná a lze na ni kliknout jako na obyčejný odkaz.

Příklad:

Moje emailová adresa s odkazem vypadá v základním formátu následovně:

```
<a href="mailto:xtomek04@stud.fit.vutbr.cz">
xtomek04@stud.fit.vutbr.cz</a>
```

Po zakódování se místo standardního odkazu v kódu stránky objeví následující řetězec:

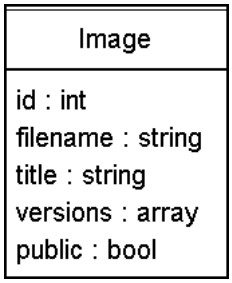
```
<a href="&#x6d;&#97;&#x69;&#108;&#x74;&#111;&#x3a;&#x78;&#116;&#x6f;
&#109;&#x65;&#107;&#x30;&#52;&#x40;&#115;&#x74;&#117;&#x64;&#46;&#x6
6;&#105;&#x74;&#46;&#x76;&#117;&#x74;&#98;&#x72;&#46;&#x63;&#122;">&
#x78;&#116;&#x6f;&#109;&#x65;&#107;&#x30;&#52;&#x40;&#115;&#x74;&#11
7;&#x64;&#46;&#x66;&#105;&#x74;&#46;&#x76;&#117;&#x74;&#98;&#x72;&#4
6;&#x63;&#122;</a>
```

Pro omezení opakovaného formátování textu při každém zobrazení se ukládá spolu s původním textem i text formátovaný a při žádosti o zobrazení se již pouze načte uložená verze. Při takovémto ukládání výstupního textu se ovšem objeví problém, jak udržovat aktuální odkazy zadané pomocí značek uvedených výše.

Nabízí se možnost ukládat spolu s textem seznam uzlů, které jsou v textu odkazovány a při jejich změně vygenerovat text znovu. V takovém případě by ale bylo třeba ukládat i uzly které jsou předky odkazovaných uzlů ve struktuře a tím už se celá věc komplikuje. Proto bylo zvoleno jednodušší řešení. Pokud je změněn libovolný uzel v systému takovým způsobem, že to ovlivní jeho odkaz, smažou se všechny uložené formátované texty a při jejich dalším zobrazení jsou znovu vygenerovány.

Pro průměrné stránky je tento přístup dostatečně jednoduchý a efektivní, protože existující stránky jen zřídka mění svoje umístění a k novému generování textů tedy prakticky nedochází.

4.2.3 Obrázky



Mimo textových dat obsahuje framework také podporu pro práci s obrázky ve formátech .JPG, .GIF a .PNG. Po nahrání obrázku do systému je obrázek uložen do definované složky a je pro něj vytvořen záznam v databázi.

4.2.3.1 Verze obrázku

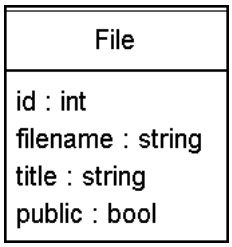
Často se můžeme setkat se situací, kdy je jeden obrázek používán na stránkách vícekrát, pokaždé v jiné velikosti. Aby nebylo nutné do systému nahrávat několik verzí obrázku, stačí nahrát pouze jednu verzi a podle potřeby jsou potom z této verze generovány potřebné zmenšené nebo zvětšené varianty. Tyto varianty jsou definovány v databázi a je možno přidávat podle potřeby nové. Při zobrazení obrázku se potom pouze použije číslo příslušné verze a je automaticky zobrazena odpovídající velikost obrázku.

4.2.3.2 Obrázky a uzly

Kromě vložení obrázku do textu uzlu je možno přiřadit obrázky přímo k uzlu samotnému. Při přiřazení obrázku k uzlu se pomocí čísla určí pozice, na kterou se má obrázek přiřadit, a pomocí tohoto čísla se potom k obrázku přistupuje.

Pokud opět použijeme příklad s aktualitou, tak můžeme na pozici 0 přiřadit obrázek, který se použije v přehledu aktualit a na pozici 1 potom obrázek, který se zobrazí v detailu aktuality.

4.2.4 Soubory



Posledním typem dat, které framework v základní podobě podporuje, jsou soubory.

Při vložení je soubor, podobně jako obrázek, uložen do definované složky a je pro něj vytvořen záznam. Na rozdíl od obrázku ale může být soubor libovolného typu a po uložení není nijak upravován.

Soubory mohou sloužit v systému k nejrůznějším účelům. Mohou být použity například jako dokumenty ke stažení k produktu v e-shopu a podobně.

4.2.4.1 Soubory a uzly

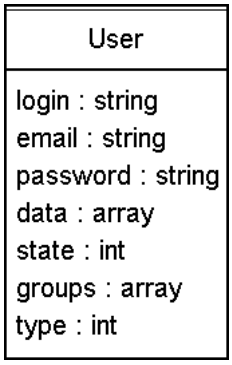
Soubor je posledním objektem, který může být k uzlu přiřazen. Na rozdíl od obrázků a textů se soubory uzlu získávají jako celé pole a není možnost přímo přistoupit ke konkrétnímu souboru.

4.3 Správa uživatelů

Framework obsahuje také několik tříd pro správu uživatelů. Na základě studia několika projektů redakčních systémů a požadavků zákazníků byl sestaven následující seznam požadavků na funkce a chování správy uživatelů.

- každý registrovaný uživatel systému musí mít jedinečný identifikátor
- pro přihlášení uživatele se použije jedinečné uživatelské jméno a heslo
- mimo základní systémové údaje uživatele musí být možnost ukládat k uživateli i další údaje, jako například jméno, adresu, telefon, …
- uživatelský účet každého uživatele musí být možno zablokovat a znemožnit tak uživateli přístup do systému bez jeho úplného smazání
- v systému musí existovat více druhů uživatelů a u každého druhu uživatele musí být možnost definovat rozdílné položky které se mají u uživatele evidovat
- pro snadnější kontrolu přístupových práv uživatelů musí systém podporovat skupiny uživatelů podle oprávnění, každý uživatel může být členem více těchto skupin

4.3.1 Reprezentace uživatele v systému



Objekt uživatele v systému je reprezentován instancí třídy **C_User**. Základním identifikátorem uživatele je jeho ID tvořené celočíselnou hodnotou, které je jedinečné v rámci ostatních uživatelů. Toto ID je pro konkrétního uživatele neměnné. Pro rozlišení uživatele v lidsky čitelné formě má každý registrovaný uživatel definováno uživatelské jméno, které musí být také jedinečné v rámci ostatních uživatelů. Mimo tyto dva údaje má ještě každý uživatel přidělené heslo a musí mít definovanou emailovou adresu.

Heslem uživatele může být libovolný neprázdný řetězec, který si uživatel zvolí při registraci do systému. Z důvodu vyšší bezpečnosti je heslo ukládáno pouze jako kontrolní součet za použití algoritmu SHA1. Pokud tedy uživatel zapomene heslo, není možné jej zjistit a musí mu být nastaveno heslo nové. Toto může provést uživatel s příslušným oprávněním.

Dále má každý uživatel definován svůj typ (viz. 4.3.3) a může být členem uživatelských skupin (viz. 4.3.2)

Uživatelský účet je možno v případě potřeby zablokovat a znemožnit tak jeho používání. Je vhodné, aby se uživatel dozvěděl, že má účet zablokován a případně i z jakého důvodu. Proto je zde možnost při blokaci uživatelského účtu zadat i důvod zablokování. Pokud je uživatelský účet zablokován a uživatel se s jeho pomocí pokusí přihlásit do systému, zobrazí se mu zpráva o zablokování a je-li zadán důvod, zobrazí se také.

Při prvním vstupu libovolného uživatele na některou stránku systému je automaticky vytvořena instance třídy, která reprezentuje anonymního uživatele. Tím je zajištěno, že i nepřihlášený uživatel má svoji instanci, která se může použít pro kontrolu přístupových práv.

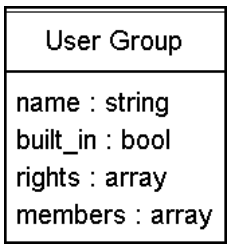
Ve chvíli, kdy se uživatel přihlásí do systému pod svým uživatelským jménem a heslem, je anonymní instance nahrazena instancí, která již reprezentuje konkrétního uživatele se všemi jeho oprávněními.

Pro zachování personalizované instance byla zvolena technologie sessions pro její transparentní použití a relativní spolehlivost. Z tohoto důvodu je třeba, aby měl uživatel ve svém prohlížeči pro dané stránky povoleny cookies.

Na začátku každého skriptu je automaticky provedena autentizace uživatele na základě jeho ID a vygenerovaného kontrolního součtu. Kontrolní součet se generuje z řetězce identifikujícího uživatelův prohlížeč. Zároveň je kvůli částečné ochraně před ukradením session změněno session ID.

Po odhlášení uživatele je jeho instance zrušena a tím se opět stává pro systém pouze anonymním uživatelem.

4.3.2 Uživatelské skupiny



Z hlediska uživatele je základní jednotkou pro správu oprávnění a přístupových práv uživatelská skupina. Veškerá oprávnění a přístupová práva pro uživatele lze přidělit pouze uživatelské skupině, nikoliv konkrétnímu uživateli. Tento přístup byl zvolen kvůli větší přehlednosti přidělování práv a snadnější správě.

Uživatelská skupina je v systému reprezentována instancí třídy **C_User_Group**. Skládá se z ID, SID, názvu skupiny, členů a přiřazených oprávnění. ID skupiny je její interní reprezentací, je tvořeno celým kladným číslem a je jedinečné v rámci ostatních skupin. Název skupiny slouží pouze pro rozlišení skupiny z hlediska uživatele a měl by být pokud možno popisný.

Aby mohl uživatel využívat oprávnění přiřazená skupině, musí mu být uděleno členství v této skupině. Zároveň může být uživatel členem libovolného počtu skupin a tím lze docílit prakticky libovolného rozdělení práv mezi uživatele. Samotný proces zpracování a uplatnění práv je popsán v kapitole 4.4.

4.3.2.1 Vestavěné uživatelské skupiny

Mimo nově definované uživatelské skupiny obsahuje systém několik vestavěných skupin a pseudoskupin, které jsou používány ke speciálním účelům.

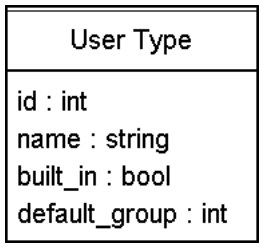
První z nich je pseudoskupina **EVERYONE**. Tato skupina má systémově přidělené **SID=1** (viz. 4.4). Jak napovídá název, reprezentuje tato skupina všechny uživatele v systému, včetně nepřihlášených uživatelů. Pokud přijde na stránky systému libovolný anonymní uživatel, je automaticky členem skupiny EVERYONE.

Druhou je pseudoskupina **AUTHENTICATED**, která má **SID=2**. Členové této skupiny jsou všichni přihlášení uživatelé. Za použití této skupiny je tedy možno rozlišit části systému pro registrované uživatele jednoduše tak, ze k neveřejným stránkám se povolí přístup pouze členům této skupiny. V tu chvíli se již nepřihlášený uživatel k těmto stránkám nedostane.

Poslední skupinou je skupina **ADMINISTRATORS** jejíž **SID=3**. Toto je také na rozdíl od dvou předchozích skupin jediná skupina, u které je možno definovat její členy. Účelem této skupiny je poskytnout svým členům neomezený přístup do celého systému. U členů této skupiny jsou ignorována veškerá přístupová oprávnění a tím mají členové přístup ke všem objektům v systému. Proto je vhodné mít v systému co možná nejméně členů této skupiny a, pokud to je jen trochu možné, raději nadefinovat uživatelům konkrétní oprávnění pomocí uživatelsky vytvořených skupin.

4.3.3 Typy uživatelů

Jedním z požadavků na systém je možnost mít v systému více typů uživatelů. Jako příklad můžeme použít jednoduchý e-shop, ve kterém je třeba mít zákazníky a správce. U zákazníků je třeba evidovat různé údaje jako adresa, číslo zákazníka, telefon. Naproti tomu u správce tyto položky potřeba nejsou. Z tohoto důvodu jsou v systému **typy uživatelů**.



Typ uživatele je definován svým ID, názvem a souborem údajů, které se u uživatelů daného typu evidují. Typ slouží pouze k definici údajů uživatele a nemá sám o sobě nic společného s oprávněními. Může mít ovšem přiřazenu jednu výchozí uživatelskou skupinu a všichni uživatelé daného typu jsou potom automaticky členy této skupiny.

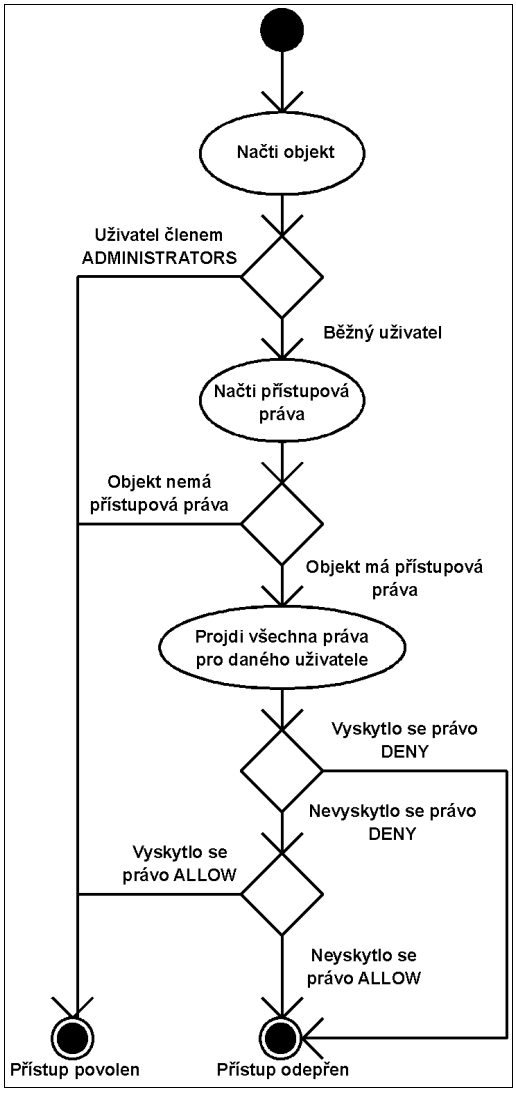
4.4 Systém oprávnění a přístupových práv

Jedním z požadavků na systém byla možnost kontrolovat přístup uživatelů do různých částí systému a možnost přidělovat různým uživatelům různé úrovně oprávnění.

Hlavním prvkem na kterém je založen systém kontroly a přidělování práv a oprávnění je bezpečnostní identifikátor - **Security Identifier – SID**. SID je celočíselná hodnota, kterou má přidělenu každý objekt v systému, který se nějak účastní kontroly oprávnění nebo přístupových práv.

Tento identifikátor je pro každý objekt jedinečný **v celém systému**. Pro jeho generování existuje v systému globální funkce `generate_sid()`, která zaručuje při každém volání vygenerování jedinečné hodnoty. Samotná funkce přečte hodnotu čítače uloženého v databázi, zvýší jeho hodnotu o jedna a přečtenou hodnotu vrátí. Před čtením hodnoty systém uzamkne danou databázovou tabulku a zámek uvolní až po aktualizaci hodnoty. Tím je zajištěno, že vrácená hodnota je vždy jedinečná a nemůže se stát, že dva souběžně běžící skripty přečtou stejnou hodnotu. Jediným omezením tohoto způsobu je rozsah použitého číselného typu integer, který je ovšem pro uvažované použití systému dostatečný.

4.4.1 Způsob zpracování



Při návrhu systému připadaly v úvahu dva opačné přístupy kontroly, inkluzivní a exkluzivní přístup.

Inkluzivní kontrola funguje na principu, kdy k danému objektu je povolen přístup pouze v případě, kdy existuje konkrétní pravidlo, které tento přístup povoluje. Tento typ kontroly je obecně bezpečnější, protože v případě opomenutí definice pravidel je přístup odepřen. Nevýhodou ovšem je, že pro každý objekt, který má být přístupný, je třeba ukládat alespoň jedno pravidlo povolující přístup.

Exkluzivní kontrola je opakem kontroly inkluzivní. Při pokusu o přístup k objektu je přístup povolen, pokud neexistuje pravidlo, které by tento přístup zakazovalo. Tento případ je méně bezpečný než inkluzivní kontrola, protože pokud programátor zapomene vytvořit pravidlo pro odepření přístupu k objektu, může k tomuto objektu přistupovat

libovolný uživatel včetně anonymních. Další nevýhodou je relativně vyšší složitost pravidel pro konkrétní objekt, kdy je třeba pro povolení přístupu úzké skupině uživatelů vytvořit několik pravidel pro odepření přístupu všem ostatním. V prostředí internetu má tento přístup výhodu ve svojí menší prostorové náročnosti. Většina stránek na internetu je veřejná a pouze menší část je určena pro omezenou skupinu uživatelů. Stačí tedy ukládat méně pravidel pro omezení přístupu k těmto stránkám.

Nakonec byla pro systém zvolena kombinace obou výše uvedených principů. Aby se ušetřil prostor a nemusela se ukládat pravidla pro všechny objekty v systému, používá se částečně exkluzivní kontrola. Pokud se uživatel pokouší o přístup k objektu a k tomuto objektu neexistuje žádné pravidlo, je mu přístup povolen. Díky tomu nemusejí mít veřejně přístupné objekty žádný záznam v tabulce pravidel přístupu.

V případě, že chceme omezit k objektu přístup pro konkrétní skupinu uživatelů, stačí vytvořit jedno pravidlo týkající se dané skupiny a povolující přístup k danému objektu. Jakmile má objekt definováno alespoň jedno libovolné pravidlo, použije se inkluzivní kontrola a je povolen přístup pouze definovaným skupinám uživatelů.

4.4.2 Přístupová práva (permissions)

První částí systému práv a oprávnění jsou přístupová práva. Přístupová práva se používají ke kontrole přístupu k objektům a jsou tvořena čtveřicí údajů v následujícím formátu:

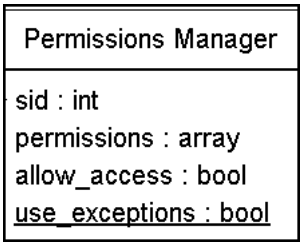
```
(SID zdroje, SID cíle, akce, typ)
```

SID zdroje jednoznačně definuje uživatele nebo skupinu, který se pokouší o přístup.

SID cíle jednoznačně určuje objekt, ke kterému se zdroj pokouší přistupovat. Může to být například uzel reprezentující stránku.

Akce je jedna ze dvou hodnot **ALLOW** a **DENY**. Pokud je akce=ALLOW, pravidlo říká, že zdrojový objekt může přistupovat k cílovému objektu. Pokud je akce=DENY, pravidlo přístup zdrojového objektu k cílovému zakazuje. Aby se vyřešily případné konflikty pravidel ALLOW a DENY, má akce DENY vyšší prioritu než akce ALLOW. Pokud tedy existují pro jednu dvojici zdroj-cíl dvě opačná pravidla, je přístup k cílovému objektu odepřen, protože pravidlo DENY převáží pravidlo ALLOW.

Typ v pravidle určuje, zda se pravidlo vztahuje na čtení nebo zápis objektu a může nabývat hodnot READ nebo WRITE. V systému je používán pouze typ READ, ale pro možné potřeby rozšiřujících modulů je implementována i možnost použít druhý typ přístupu.



Pro správu přístupových práv k objektům slouží v systému třída **C_Permissions_Manager**. Pokud je potřeba, aby některý objekt

využíval kontroly přístupu uvedené výše, stačí odvodit jeho třídu z třídy C_Permissions_Manager. Tím získá metody a vlastnosti, které pokryjí kontrolu přístupu pomocí přístupových práv.

Jedinou věcí, kterou si musí nová třída zajistit sama, je vytvoření, uložení a načítání SID. Je tomu tak proto, že nelze dopředu určit, jakým způsobem bude třída implementována a jak bude ukládat svá data. Vytvoření SID se provede jednoduše pomocí funkce `generate_sid()` (viz. 4.4).

4.4.2.1 Kontrola přístupových práv

Ke kontrole přístupových práv slouží metoda `authorize_user($type)`, která je obsažena ve třídě C_Permissions_Manager. Tato metoda po zavolání ověří, zda má aktuální uživatel přístup k objektu, ze kterého je volána. Pokud má uživatel k objektu přístup, nestane se nic a pokračuje se ve vykonávání kódu. V případě, že uživateli je přístup k objektu odepřen, závisí chování na nastavení.

V základním nastavení je při pokusu o přístup k zakázanému objektu vyhozena výjimka třídy **C_403_Exception**. Tím dojde k přerušení prováděného skriptu a pokud není výjimka zpracována, je zachycena až na konci celého skriptu a uživatel je přesměrován na stránku s informací o odepření přístupu.

Toto chování lze využít například u jednotlivých stránek, kdy při pokusu o zobrazení stránky, ke které nemá uživatel přístup, je automaticky přesměrován.

Nevýhodou výjimek v základním nastavení je jejich relativně vysoká režie. To se projeví například v případě, kdy je velká část stránek pouze pro registrované uživatele. Přijde-li na stránky neregistrovaný uživatel, dojde při generování navigace k několikanásobnému vyhození a zpracování výjimky a při větším počtu se tak znatelně prodlouží doba zpracování skriptu. Z tohoto důvodu je možno změnit chování metody `authorize_user($type)` tak, že místo vyhození výjimky se pouze nastaví příznak v objektu.

Nastavení se provádí pomocí statické metody `use_exceptions(false)`, která je také obsažena ve třídě C_Permissions_Manager. Pokud se zakáže použití výjimek touto metodou, nedojde při zavolání metody k vyhození výjimky, ale nastaví se příznak. Tento příznak je potom možno testovat metodou `access_allowed()`, která vrací false, pokud není k objektu povolen přístup. Je důležité nakonec opět povolit výjimky opětovným zavoláním statické metody `use_exceptions(true)`, jinak může dojít k neočekávanému chování na jiných místech skriptu.

Funkce pro potlačení výjimek je vhodná například pro již zmiňovaný proces generování navigace nebo pro jakýkoliv proces, ve kterém dochází k načítání většího množství objektů, které mohou mít omezený přístup.

4.4.3 Oprávnění (rights)

Druhou částí systému práv a oprávnění jsou oprávnění. Oprávnění se v systému používají ke kontrole, zda může daný uživatel či skupina provádět požadovanou operaci. Oprávnění je definováno textovým řetězcem, který by měl být popisný pro dané oprávnění, a SID.

Textový řetězec, také zvaný **akronym**, slouží jako identifikátor daného oprávnění a používá se k určení konkrétního oprávnění.

SID je standardní bezpečnostní identifikátor a díky němu je oprávnění objektem, ke kterému se možno kontrolovat přístup.

Samotné přidělování a kontrola oprávnění probíhá za použití přístupových práv k danému oprávnění. Pokud má mít uživatel nebo skupina dané oprávnění, vytvoří se přístupové právo k tomuto oprávnění za použití jeho SID jako cíle, akce na ALLOW a typu na READ. Kontrola potom probíhá stejně jako u každého jiného objektu. Ověří se, zda má uživatel přístup k oprávnění a pokud ano, je mu povolena akce daná oprávněním.

5 Moduly

Hlavním účelem frameworku je jeho používání pro tvorbu redakčních systémů. Vzhledem k tomu, že framework sám neposkytuje přímo možnost tvorby stránek, je jediným možným použitím tvorba modulů, které mu přidávají požadovanou funkčnost.

5.1 Co je to modul

Z hlediska adresářové struktury frameworku se modulem rozumí skupina souborů v předem definovaném formátu a struktuře, která obsahuje skripty, šablony a další potřebná data.

Z pohledu programu je to potom skupina tříd, funkcí a dat, které přidávají nové funkce do frameworku, případně modifikují funkce existující.

Příkladem modulů pro framework mohou být moduly pro tvorbu obyčejných statických stránek, pro správu aktualit, e-shopu a další.

5.2 Struktura modulu

Modul pro framework musí dodržovat určité předepsané konvence a strukturu, které se týkají například názvů souborů, adresářové struktury a dalších dat.

5.2.1 Programové skripty

Adresářová struktura modulu musí obsahovat minimálně následující adresáře a soubory:

```
nazev_modulu
  |- administration
  | \- nazev_modulu.php
  |- classes
```

```
|- include
| |- prologue.php
| \- epilogue.php
\- nazev_modulu.php
```

Soubor `nazev_modulu/nazev_modulu.php` je hlavním skriptem modulu. Tento skript je volán pokaždé, když uživatel požaduje uzel, který náleží příslušnému modulu. Narazí-li vstupní skript systému na uzel modulu, zavolá právě tento skript a další chování již závisí plně na daném modulu.

Soubor `nazev_modulu/administration/nazev_modulu.php` je potom hlavní skript administrační části pro modul. Podobně jako předchozí skript je tento zavolán v případě, kdy se uživatel nachází v administraci v části patřící modulu.

Konečně soubor `nazev_modulu/include/prologue.php` je vykonán na začátku každé stránky a měl by obsahovat potřebnou inicializaci modulu, naopak skript `nazev_modulu/include/epilogue.php` je vykonán na konci každé stránky a měl by obsahovat kód pro poslední úpravu dat před jejich zobrazením.

Kterýkoliv z uvedených skriptů může být prázdný, ale musí existovat kvůli správné funkčnosti celého systému.

Pokud budou v modulu použity vlastní třídy, měly by být uloženy v adresáři `classes`. Není potom třeba vkládat je ručně, ale framework načte tyto třídy automaticky.

Celá výše uvedená adresářová struktura se musí nacházet v adresáři `/modules`.

5.2.2 Šablony

Mimo programové skripty je většinou potřeba, aby modul obsahoval i šablony pro zobrazení svého výstupu. Tyto šablony již mohou mít libovolné názvy a mohou se nacházet na libovolném místě, protože jejich načítání si kontroluje sám modul. Je ale vhodné dodržovat jednotný způsob ukládání. Doporučuje se ukládat šablony do `/templates/nazev_modulu`.

5.3 Funkce modulu

Modul slouží jako rozšíření frameworku a jako takový může poskytovat prakticky libovolné funkce. Je pouze na tvůrci modulu, co bude jeho modul dělat a jak se bude v systému projevovat.

Může se jednat o modul přidávající jistý druh obsahu, například aktuality. Modul také může doplňovat a rozšiřovat již existující moduly. Pokud to vezmeme do krajnosti, nemusí modul vůbec využívat funkcí frameworku a může sám definovat vše, od rozhraní k databázi až po zobrazování obsahu bez použití Smarty.

Tento přístup umožňuje rozšiřovat framework prakticky libovolným směrem a použít jej tak pro jakýkoliv účel.

5.4 Instalace modulu

Pokud chceme modul použít v systému, je třeba jej nejdříve nainstalovat.

V prvé řadě je potřeba umístit soubory modulu do adresáře `/modules` a případné šablony například do adresáře `/templates`. Aby byl modul frameworkem vůbec vyhledáván a načten, je třeba vytvořit v databázové tabulce „modules“ pro daný modul záznam. Tento záznam musí obsahovat jedinečné ID modulu, pod kterým bude modul veden v systému, dále potom kompletní název modulu a jeho název použitý u souborů.

V případě potřeby zbývá vytvořit texty modulu, nová oprávnění, nastavení a databázové tabulky.

Podle složitosti modulu je vhodné vytvořit spolu s modulem i jeho instalační skript, který požadované akce provede automaticky.

Vzhledem k možné složitosti a rozmanitosti modulů neobsahuje framework sám žádný instalátor pro instalaci modulů, vše je v režii tvůrce modulu.

Po dokončení instalace modulu je modul automaticky načítán při zpracování skriptů a není třeba jej nijak ručně spouštět.

6 Lokalizace

Pro snadnější lokalizaci frameworku a redakčních systémů na něm založených obsahuje framework rozhraní pro dynamické načítání nejrůznějších textů a popisků, které se v systému používají.

Toto rozhraní je reprezentováno třídou **C_Translator**, která slouží k načítání a úpravě definovaných textů. Text je definován svojí zkratkou a vlastním obsahem, který je možno podle potřeby upravovat. Veškeré takto definované texty se ukládají v databázi a je-li potřeba daný text na stránce použít, jednoduše se načte jeho aktuální verze z databáze.

Pro lokalizaci systému do jiného jazyka potom stačí pouze přeložit obsah jedné databázové tabulky a veškeré popisky, tlačítka a hlášky se změní.

Pro omezení počtu dotazů nutných pro načítání textů, je možno texty, které jsou ve skriptu a na stránce použity, dopředu načíst. To se provede v jediném dotazu a není tak třeba načítat každý text jednotlivě.

7 Cache

Při návrhu frameworku vyvstal problém, jak zajistit jeho rychlost i v situacích, kdy bude používán s mnoha rozšiřujícími moduly, které mohou vykonávat časově náročné činnosti. Protože podobu použitých modulů a jejich optimalizaci nelze dopředu ovlivnit, obsahuje framework i funkci pro cachování výstupu.

Pokud je tato funkce aktivována, je výsledná podoba zobrazované stránky uložena do databáze a při opakovaném požadavku na zobrazení stejné stránky není prováděno žádné generování a je pouze zobrazena uložená verze. Tím je dosaženo velmi rychlého zobrazení obsahu stránky.

V konfiguračním souboru frameworku je nastavena základní doba platnosti uložené verze na jeden den, ale je samozřejmě možno tuto hodnotu v konfiguraci upravit. Po vypršení platnosti uložené verze je vygenerována verze nová.

Pokud ovšem během platnosti cache změní uživatel obsah některé stránky, nastává problém, kdy je nová verze zobrazena až po vypršení platnosti verze staré. Aby se předešlo tomuto nežádoucímu jevu, obsahuje framework pro tento případ jednoduchý mechanismus. Spolu s uloženou verzí stránky se ukládá také seznam všech uzlů, které se na obsahu dané stránky nějakým způsobem podílejí. V případě, že je kterýkoliv uzel nějakým způsobem změněn, vyhledají se všechny uložené verze stránek, ve kterých se daný uzel vyskytuje a tyto verze jsou smazány. Při příští návštěvě je vygenerována verze nová a tím je zajištěno, že se libovolná změna obsahu projeví okamžitě.

Pokud jsou v systému některé stránky, které není vhodné takto ukládat, je jednoduše možné pro danou stránku tuto funkci vypnout pomocí globální proměnné. Typickým případem, kdy je ukládání stránek do paměti nežádoucí, je administrace. Proto je pro administrační část ukládání zakázáno systémově.

Pro identifikaci stránky se používá její kompletní URL, kdy dvě rozdílná URL jsou považována za rozdílné stránky a jsou ukládány odděleně. Do URL se zahrnují i případné proměnné na konci.

Tento systém má ovšem jeden nedostatek který může jeho použití znemožnit. Pokud je třeba, aby na všech stránkách byla určitá část plně dynamická, nelze systém vůbec použít. Příkladem může být zobrazování náhodných produktů pod navigací na všech stránkách.

8 Závěr

Pokud mám zhodnotit dosažené výsledky této práce vzhledem k původní představě, mohu říci, že vytvořený framework je prakticky použitelný a skutečně zjednodušuje tvorbu systémů pro prezentaci na internetu.

Vhledem k jeho modularitě a rozšiřitelnosti je možné jej použít pro mnoho nejrůznějších zadání a požadavků klienta. Když srovnám náročnost tvorby redakčního systému bez použití tohoto frameworku a s ním, dojdu k jednoznačnému závěru, že použití frameworku celý proces výrazně zjednodušuje.

Odpadá opakovaná tvorba základních částí redakčního systému, jako je správa uživatelů, organizace obsahu a další. Nyní již stačí pouze vytvořit konkrétní moduly, plnící specifické požadavky na systém, a po jejich integraci do frameworku je redakční systém připraven k používání. Navíc i tvorba těchto modulů je jednodušší, neboť je možné používat již hotové funkce z frameworku a pouze je rozšiřovat nebo modifikovat.

Tento systém jsem začal tvořit pro použití ve svých vlastních projektech a již v několika takovýchto projektech jsem jej úspěšně použil. Vzhledem k tomu, že každý nový projekt může přinést nové požadavky na tento systém, je velice pravděpodobné, že i v budoucnu se bude funkčnost frameworku podle potřeb rozšiřovat.

Při tvorbě tohoto frameworku jsem poznal v praxi inkrementální vývoj. I když jsem se snažil na začátku v návrhu zohlednit co nejvíce aspektů, po prvním praktickém použití systému jsem objevil mnoho dalších možností a požadavků, které jsem do systému na počátku nezahrnul. Verze, která je předmětem této práce je tedy již několikátou v pořadí a bude se i nadále vyvíjet.

Literatura

[1] PHP Documentation Group. PHP Manual [online]. 1995-2007 , 2007-04-17 [cit. 2007-04-18]. Dostupný z WWW: <http://www.php.net/manual/en/>.

[2] MySQL AB. MySQL 3.23, 4.0, 4.1 Reference Manual [online]. 1995-2007 , 2007-04-26 [cit. 2007-04-26]. Dostupný z WWW: <http://dev.mysql.com/doc/refman/4.1/en/index.html>.

[3] OHRT, Monte, ZMIEVSKI, Andrei. Smarty - the compiling PHP template engine [online]. 2001-2005 , 27-09-2006 [cit. 2007-04-26]. Dostupný z WWW: <http://smarty.php.net/manual/en/>.

[4] Kolektiv autorů. PHP Programujeme profesionálně. [s.l.] : Nakladatelství Computer Press, a.s., 2001. 676 s., CD-ROM. ISBN 8072263102.

[5] Enigma SEO Consulting. Combating Email Harvester Robots - Email Obfuscation [online]. 2002-2006 [cit. 2007-04-26]. Dostupný z WWW: <http://www.seowebsitepromotion.com/obfuscate_email.asp>.

[6] SCHLOSSNAGLE, George. Pokročilé programování v PHP5. [s.l.] : Zoner Press, 2004. 640 s. ISBN 80-86815-14-5.

Seznam příloh

Příloha 1: Popis instalace systému

Pro správnou funkci frameworku je třeba jej před prvním použitím nainstalovat na server. Systém má následující minimální softwarové požadavky:

- PHP 5.0
- MySQL 4.1
- Apache 1.3

V prvním kroku je třeba nahrát všechny soubory aplikace na server, kde chceme aplikaci provozovat. Po nahrání souborů je třeba nastavit oprávnění pro některé adresáře na 0777. Jedná se o:

- `<adresar_aplikace>/templates/compiled`
- `<adresar_aplikace>/config`
- `<adresar_aplikace>/files`
- `<adresar_aplikace>/images`

Framework obsahuje jednoduchý instalátor. Po zkopírování všech souborů aplikace na server se instalátor nachází na adrese `http://<url_do_slozky_aplikace>/installer`.

Po zadání URL instalátoru je uživatel vyzván k zadání údajů pro připojení k databázi a volitelného prefixu k databázovým tabulkám. Prefix slouží pro instalaci systému do jedné databáze vedle jiného systému. Je tím zajištěno, že nedojde ke konfliktu s již existujícími tabulkami jiného systému.

Poté zadá uživatel údaje pro vytvoření prvního administračního účtu pro správu systému.

V dalším kroku musí uživatel zadat absolutní cestu a URL k aplikaci. Následuje kontrola zadaných údajů a po potvrzení se framework nainstaluje.

Pokud nenahlásí instalátor žádnou chybu, je framework připraven k použití.

V základním nastavení používá framework pro svoji funkci skript "cz" bez přípony. Pokud to dovolí nastavení serveru, použije se automaticky pro konfiguraci přiložený soubor ".htaccess", který nastaví zpracování skriptu „cz“ na parser PHP. Pokud je ovšem na serveru zpracování souboru .htaccess zakázané, je nutné nastavení provést ručně přímo na serveru.

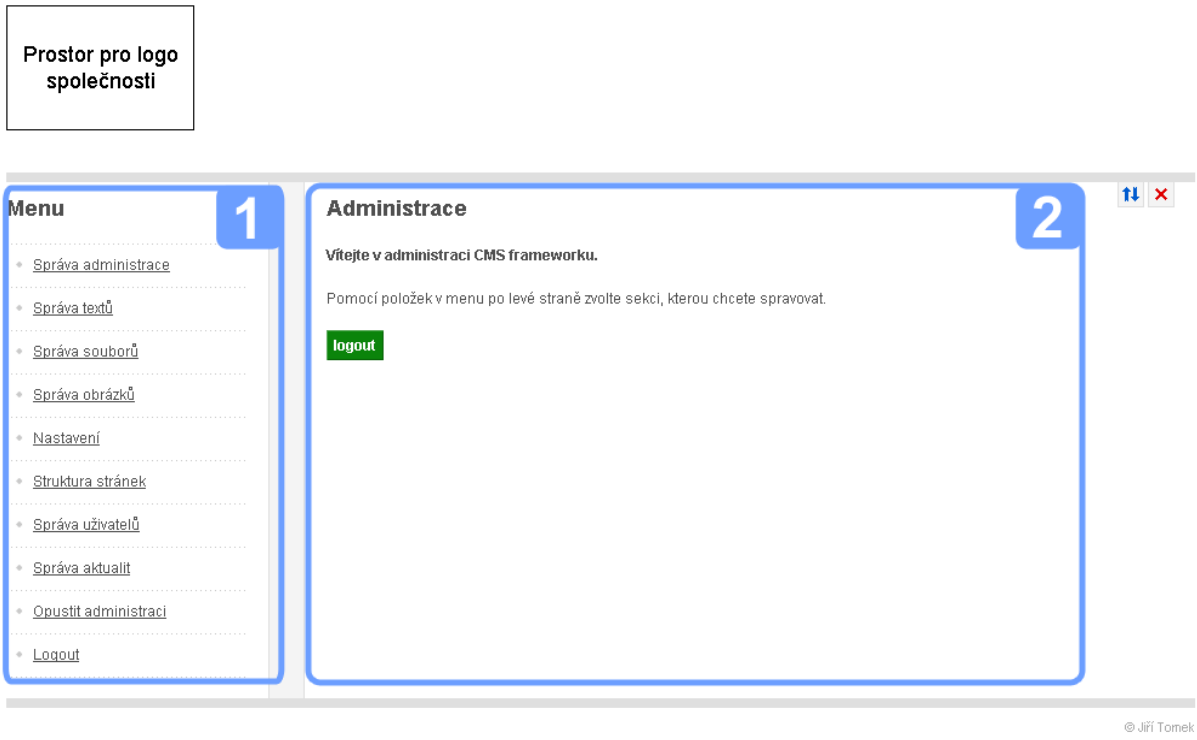
V případě, že nelze z jakéhokoliv důvodu provést konfiguraci na serveru, existuje ještě jedna možnost. Lze přejmenovat soubor `<adresar_aplikace>/cz` na `<adresar_aplikace>/cz.php`. Potom je ovšem nutné po nainstalování aplikace změnit v konfiguračním souboru `<adresar_aplikace>/config/config.php` řádek:

```
[65] define('EXTENDED_URL',                ROOT_URL.'cz/');
```

na

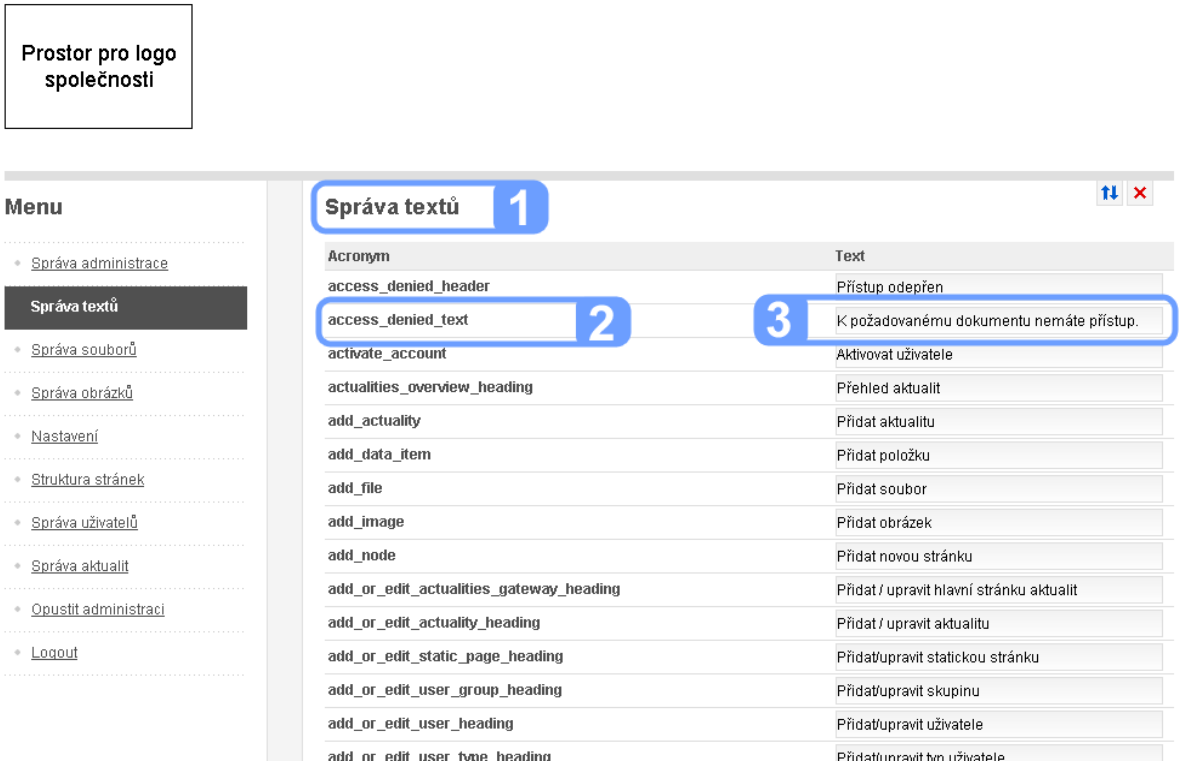
```
[65] define('EXTENDED_URL',                ROOT_URL.'cz.php/');
```

Příloha 2: Popis obrazovek administrace



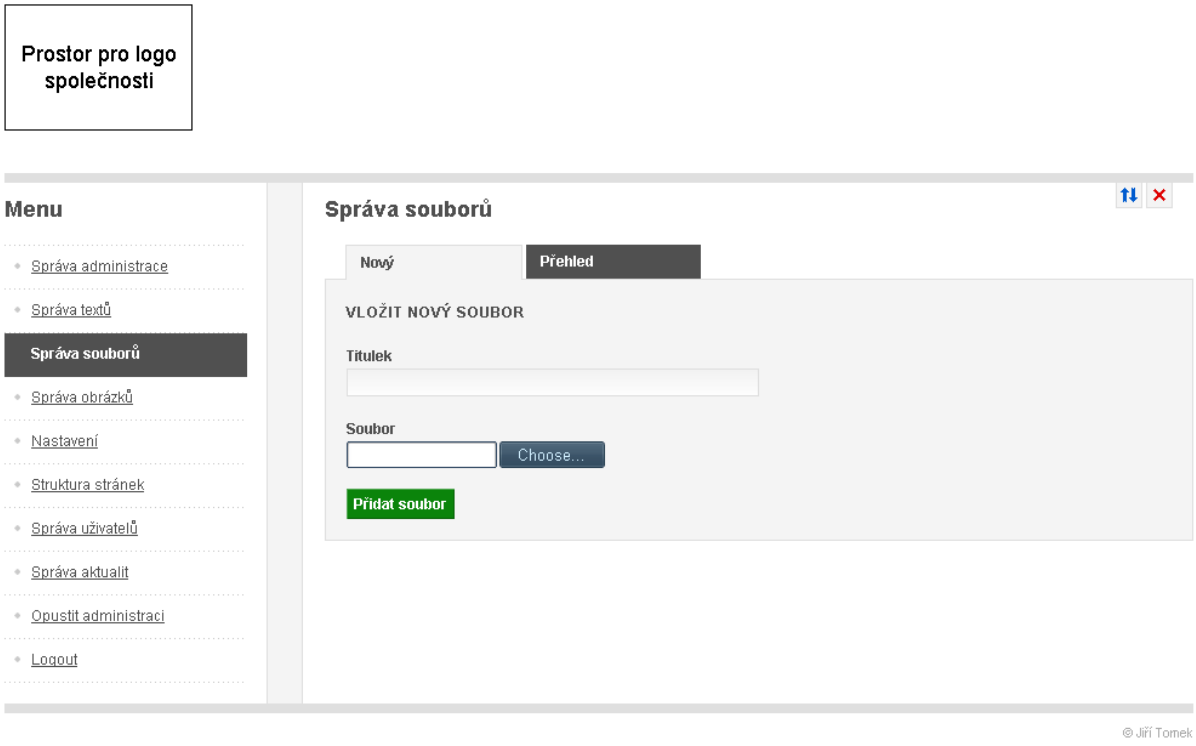
Úvodní stránka - zde se správce objeví po přihlášení do administrace.

- [1] – Hlavní menu. Slouží k navigaci v administraci a obsahuje jednotlivé sekce.
- [2] – Vlastní obsah zvolené sekce administrace.



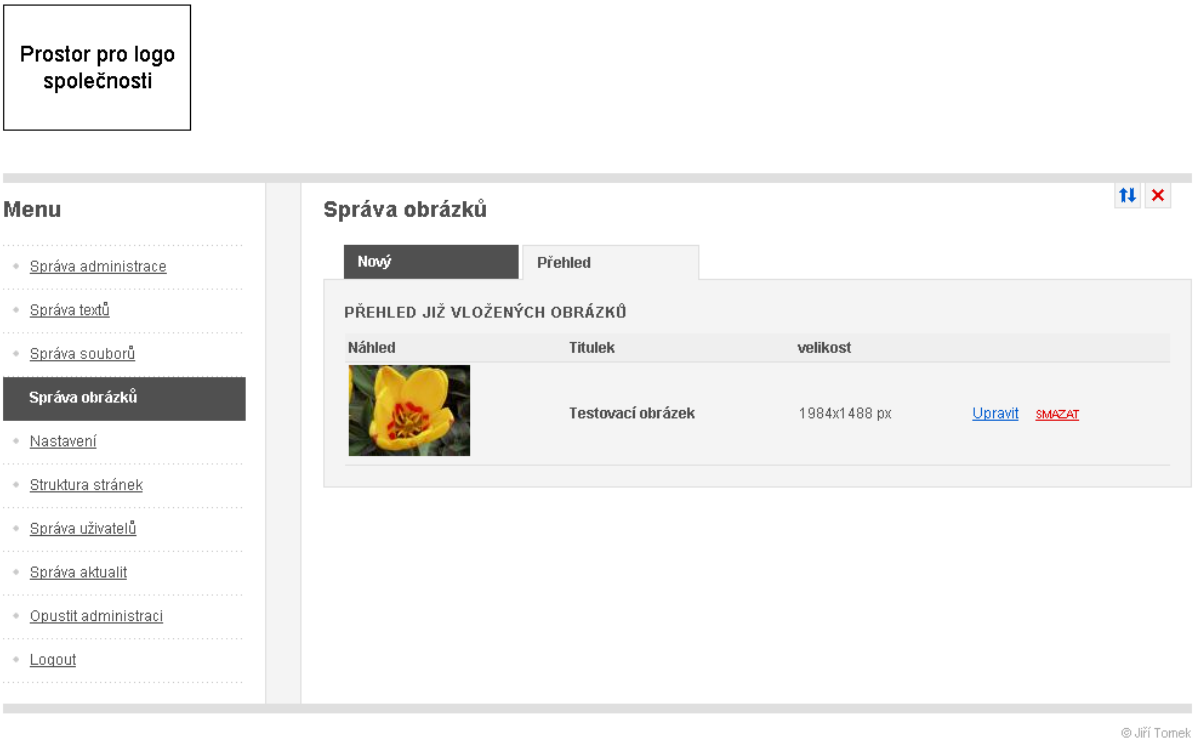
Správa textů a popisků systému.

- [1] – Nadpis zvolené sekce administrace.
- [2] – Identifikátor textového popisku používaný v systému.
- [3] – Vlastní hodnota používaná jako text v místě použití identifikátoru.

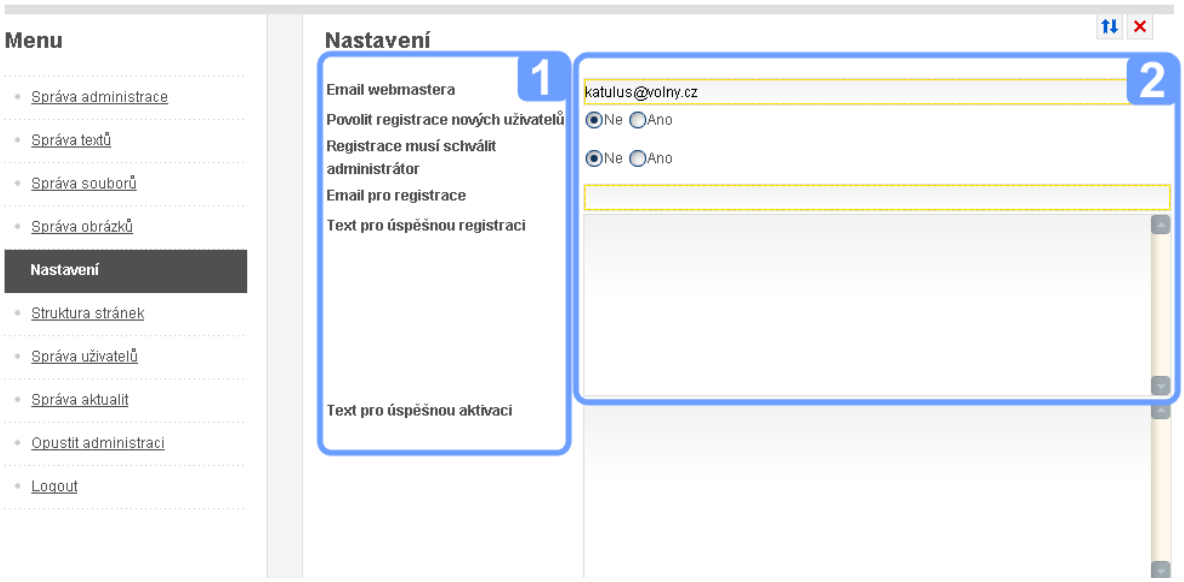


Správa souborů systému. Zde se vkládají a spravují soubory, uložené v systému. Při vkládání souboru je vhodné mu přiřadit titulek, který se potom může používat pro identifikaci souboru.

Na záložce „Přehled“ je potom tabulka obsahující jednotlivé soubory v systému.



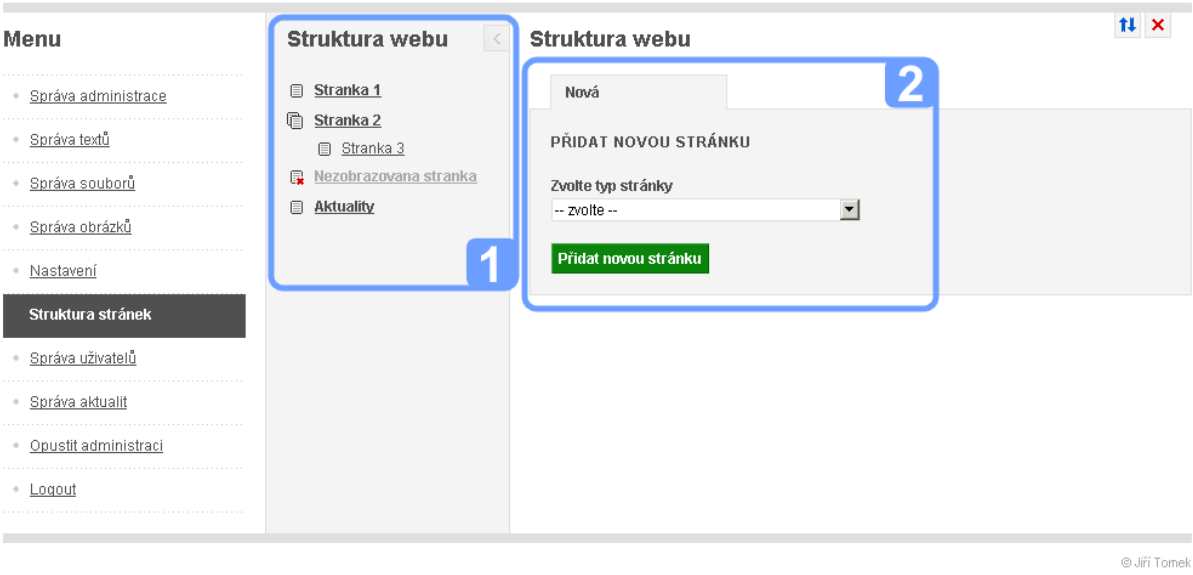
Správa obrázků. Proces vkládání obrázku je stejný jako u souboru (viz. výše). Zde je vidět přehled vložených obrázků s jejich náhledem, popisem a velikostí. Použitím voleb „Upravit“ a „SMAZAT“ je možno měnit popis obrázku, případně celý obrázek odstranit.



Správa nastavení systému. V této sekci se nachází veškeré položky nastavení systému, včetně položek přidaných jednotlivými moduly.

- [1] – Popisky jednotlivých položek
- [2] – Pole pro zadávání hodnot nastavení.

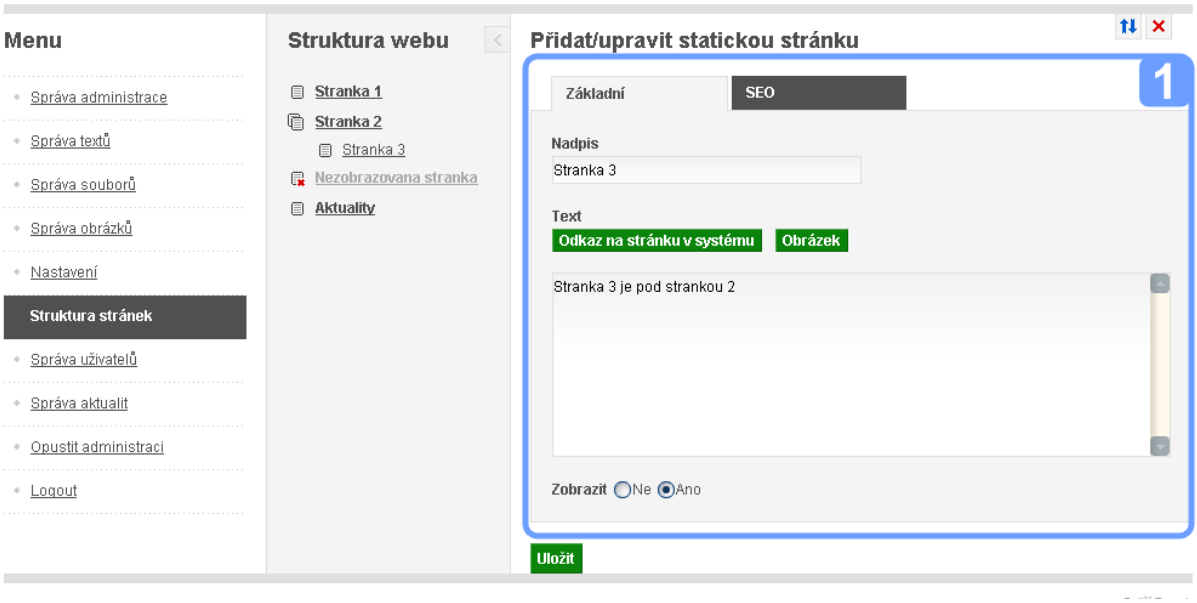
Prostor pro logo společnosti



Struktura stránek webu. Tato sekce slouží ke správě jednotlivých stránek prezentace.

- [1] – Stromová reprezentace struktury prezentace. Kliknutím na položku se dostaneme k její editaci.
- [2] – Volby pro vytvoření nové stránky zvoleného typu.

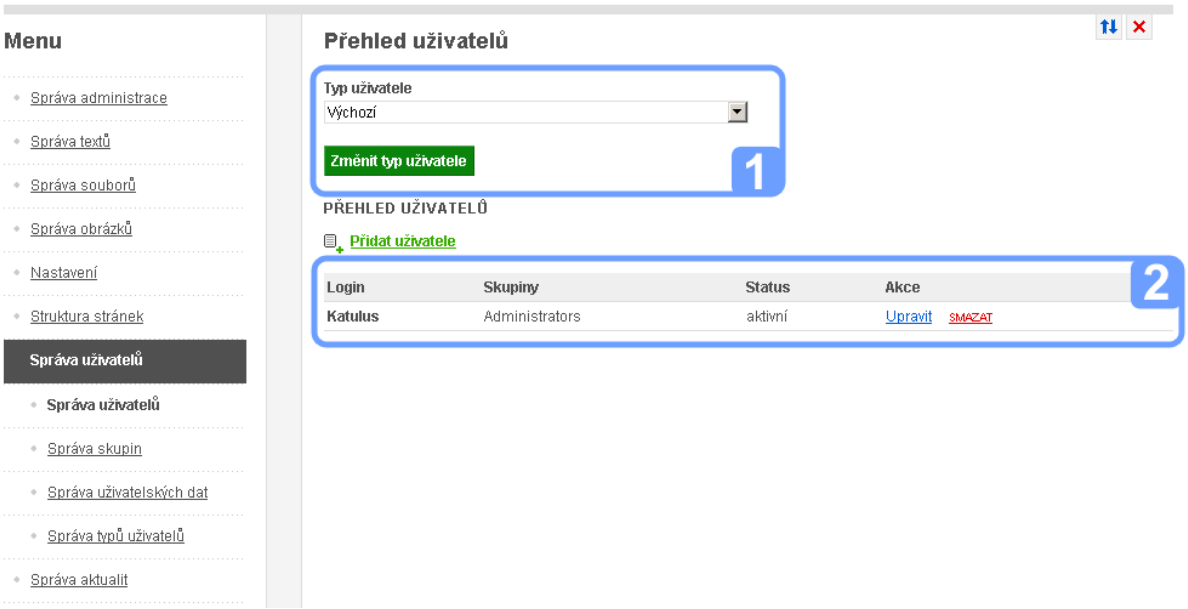
Prostor pro logo společnosti



Editace zvolené stránky ve struktuře.

- [1] – Podle typu stránky se zde zobrazí položky, které může daná stránka mít. Použitím tlačítek „Odkaz na stránku v systému“ a „Obrázek“ je možno vložit dynamický odkaz na jiné místo v prezentaci, případně obrázek.

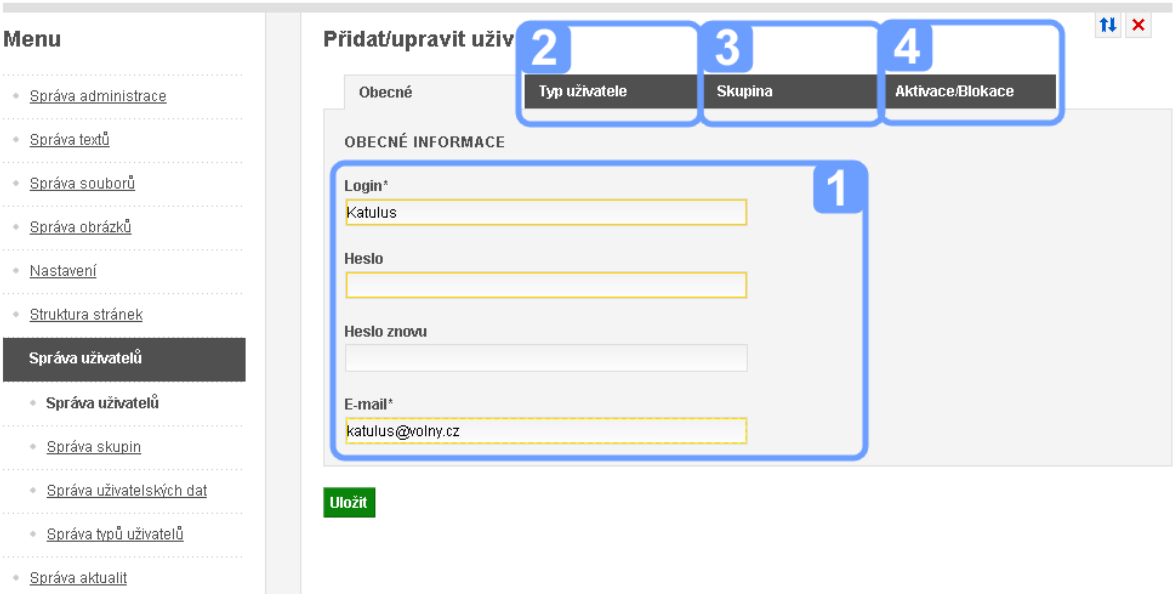
Prostor pro logo společnosti



Správa uživatelů systému.

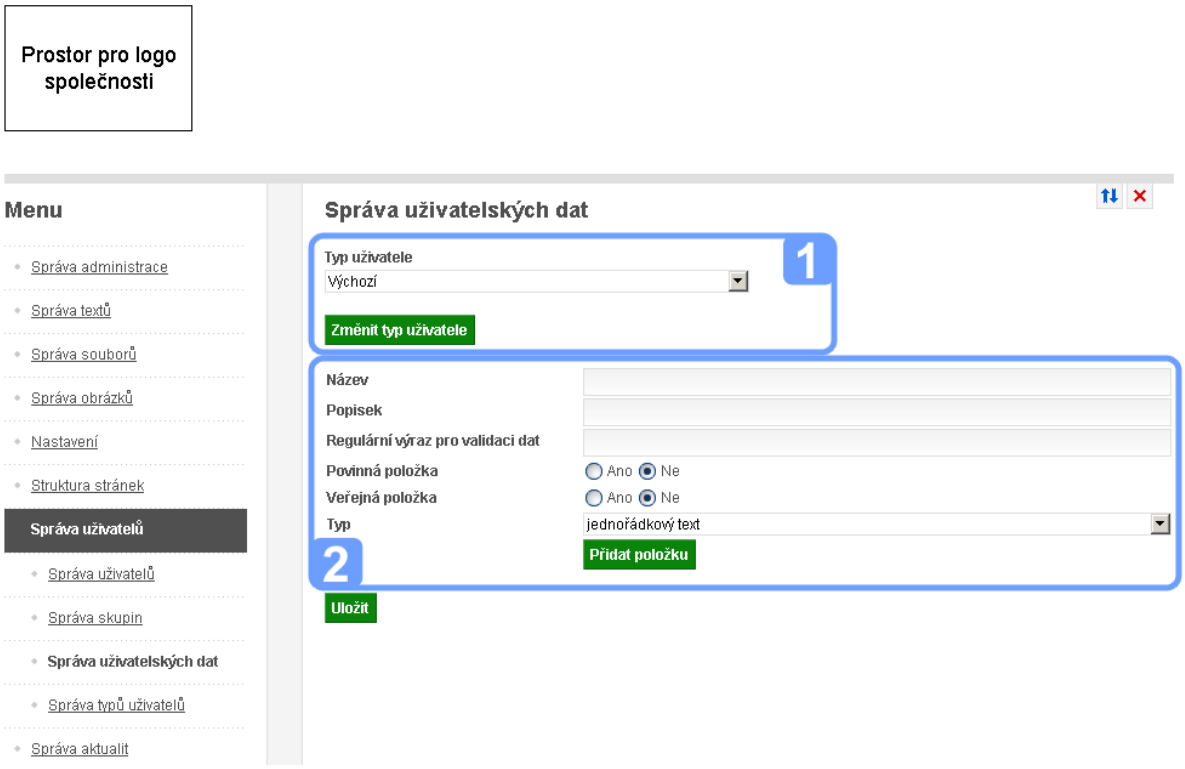
- [1] – Volba typu uživatele. Podle zvoleného typu se zobrazí přehled uživatelů.
- [2] – Přehled uživatelů zvoleného typu.

Prostor pro logo společnosti



Editace uživatele.

- [1] – Vlastnosti daného uživatele. Různé typy uživatelů mohou mít různé položky.
- [2] – Záložka s volbou typu uživatele.
- [3] – Volba skupin, kterých je uživatel členem.
- [4] – Volby pro aktivování či zablokování uživatelského účtu.



Správa položek uživatelských dat pro jednotlivé typy uživatelů.

- [1] – Volba typu uživatele, jehož položky chceme editovat.
- [2] – Jedna položka. Skládá se z názvu, popisku, regulárního výrazu pro ověřování vkládaných dat. Je možno zvolit její typ a přístupnost.

Příloha 3: Ukázková implementace modulu

Nejlepším návodem pro pochopení struktury a tvorby modulu je příklad. Nejjednodušším a vždy potřebným modulem je modul pro zobrazování statických stránek. Abychom ale viděli i některé pokročilejší aspekty tvorby modulu, jako ukázkový použijeme modul pro správu a zobrazování aktualit.

Zadání

Modul má sloužit k zobrazování a správě jednoduchých textových aktualit s možností přiložení obrázku nebo souboru. Aktualita bude mít svůj nadpis, datum vložení, úvodní a hlavní text.

Vytvoření adresářové struktury

Jak bylo popsáno výše (5.2), vytvoříme následující strukturu modulu:

```
actualities
  |- administration
  | |- actualities.php
  | |- edit_actuality.php
  | \- edit_gateway.php
  |- classes
  | |- c_actuality.php
  | \- c_actualities_page.php
  |- config
  | \- config.php
  |- include
  | |- prologue.php
  | \- epilogue.php
  \- actualities.php
```

Návrh třídy aktuality

Pro reprezentaci aktuality v systému s výhodou použijeme třídu „uzel“, kterou rozšíříme o další vlastnosti specifické pro aktualitu. Nazvěme si novou třídu **C_Actuality**, která bude odvozenou třídou z třídy **C_Node**.

Nadpis

Pro nadpis aktuality můžeme využít vlastnost uzlu „název“. Tím zároveň zajistíme hezký tvar URL, kdy bude vypadat například takto: http://www.domena.cz/cz/aktuality/nova-aktualita

Texty

Aktualita má obsahovat dva texty. První text – můžeme jej nazvat ukázkový – bude zobrazen v přehledu všech aktualit a bude sloužit jako stručný obsah aktuality. Druhý text – hlavní – již bude obsahovat celý obsah aktuality a zobrazí se při prohlížení detailu dané aktuality.

Bázová třída **C_Node** obsahuje podporu pro ukládání textů a na nás je, elegantně ji využít ve třídě aktuality. Vytvoříme tedy ve třídě **C_Actuality** čtyři metody, které nazveme:

- `get_sample_text($formatted)`
- `get_main_text($formatted)`
- `set_sample_text($text)`
- `set_main_text($text)`

Pro určení ukázkového a hlavního textu vytvoříme dvě konstanty třídy s názvy SAMPLE_TEXT a MAIN_TEXT s hodnotami 0 a 1. Výše uvedené metody potom budou vypadat například takto:

```
public function set_sample_text($text)
{
  $this->get_text(self::SAMPLE_TEXT)->set_text($text);
}
public function get_sample_text($formatted = true)
{
  return $this->get_text(self::SAMPLE_TEXT)->get_text($formatted);
}
```

Metody tedy nedělají nic jiného, než že zapouzdřují použití metod bázové třídy.

Za zmínku stojí parametr `$formatted` u metody `get_sample_text()`. Jak jsem zmínil v kapitole Texty (4.2.2), každý text má dvě podoby. První je čistý text tak jak byl zadán bez jakéhokoliv formátování. Druhou je formátovaný text s převedenými odkazy, vloženými obrázky a podobně. Pomocí parametru `$formatted` metodě říkáme, jakou verzi textu chceme získat.

Datum vložení

Datum vložení aktuality je položka, pro kterou nemůžeme využít žádnou vlastnost bázové třídy **C_Node**. Proto vytvoříme novou privátní vlastnost, kterou nazveme například `$date`. Pro přístup k této vlastnosti přidáme metody `get_date()` a `set_date($date)`.

Konstruktor

Třída aktuality je odvozená od třídy **C_Node** a proto musíme v konstruktoru ručně zavolat i konstruktor bázové třídy. Vzhledem k tomu, že aktualita obsahuje i položku „datum“, která nemůže být uložena v bázové třídě, je třeba v konstruktoru tuto položku načíst.

Důležitou částí konstruktoru je také nastavení modulu a typu pro uzel. Jak bylo řečeno v kapitole popisující uzly (4.2.1), má každý uzel vlastnost „modul“ a „typ“. Vlastnost „modul“ musí být nastavena na ID modulu aktualit a vlastnost „typ“ může být libovolná, záleží na návrhu modulu. V našem případě nastavíme vlastnost „typ“ na hodnotu 1.

Uložení a odstranění

Pro uložení stavu aktuality vytvoříme metodu `save()`, která požadovanou akci provede. Metoda `save()` přetěžuje stejnojmennou metodu z třídy **C_Node** a proto nesmíme zapomenout tuto metodu na začátku ručně zavolat. Tím dojde k uložení uzlu, textů, obrázku a souboru, ale zbývá uložit datum aktuality, které není v bázové třídě obsaženo. Z toho důvodu potřebujeme vlastní metodu `save()`.

Podobně pro odstranění aktuality vytvoříme metodu `delete()`, která opět přetěžuje metodu bázové třídy. Zde napřed smažeme z databáze záznam s datem aktuality a poté zavoláme metodu bázové třídy, která se postará o odstranění zbytku uzlu.

Návrh třídy stránky aktualit

Druhým objektem v modulu aktualit bude samotná stránka, na které se bude zobrazovat přehled vložených aktualit. Třídu pro tuto stránku nazveme **C_Actualities_Page** a bude opět odvozena od třídy **C_Node**.

Tato stránka bude obsahovat pouze nadpis a text pro popis jejího obsahu. Stejně jako u aktuality tedy vytvoříme metody pro práci s textem a v konstruktoru nastavíme modul a typ uzlu. Pro aktualitu jsme zvolili typ uzlu roven jedné, proto pro stránku zvolíme typ roven dvěma.

Na rozdíl od aktuality neobsahuje stránka žádnou položku, která není obsažena v bázové třídě. Nemusíme tedy vytvářet metody pro ukládání a odstranění stránky, ale můžeme používat metody bázové třídy **C_Node**.

Návrh databázové struktury

Při návrh třídy aktuality jsme narazili na potřebu ukládat k aktualitě datum. Protože toho nelze dosáhnout pouze za použití bázové třídy uzlu, musíme vytvořit novou databázovou tabulku, do které budeme datum ukládat. K tomu nám stačí dva sloupce, jeden pro uložení ID uzlu, který představuje aktuality, a druhý pro samotné datum. Tabulka tedy může vypadat následovně:

```
CREATE TABLE `actualities` (
  `act_node` INT UNSIGNED NOT NULL ,
  `act_date` INT UNSIGNED NOT NULL ,
  PRIMARY KEY ( `act_node` )
) ENGINE = MYISAM ;
```

Pro uložení data jsem zvolil typ integer, protože budeme datum ukládat ve formátu unix timestamp.

Tvorba skriptů

Nyní již máme vytvořeny třídy pro modul a tak se zaměříme na tvorbu ostatních skriptů modulu.

config.php

Jako konfigurační skript modulu zvolíme skript s názvem `config.php`, který umístíme do podadresáře `config`.

Do tohoto souboru uložíme konstanty s definicí názvu databázové tabulky

```
define('DBT_ACTUALITIES', DBTABLE_PREFIX.'actualities');
```

a dále konstanty s typy jednotlivých uzlů, které budou v modulu použity

```
define('ACTUALITIES_NODE_ACTUALITY',    1);
define('ACTUALITIES_NODE_GATEWAY',      2);
```

prologue.php a epilogue.php

Povinnými skripty modulu jsou skripty `prologue.php` a `epilogue.php`. Skript `prologue.php` je inicializační skript modulu a bude vykonáván vždy na začátku hlavního skriptu systému. Na tomto místě je vhodné nastavit nejrůznější konstanty modulu, načíst potřebná data a podobně.

Pro účely všech modulů je dobré uložit si do vhodných konstant ID a název příslušného modulu. Vzhledem ke způsobu načítání je ve skriptu `prologue.php` k dispozici ID modulu v globální proměnné `$GLOBALS['active_module_id']` a název modulu v `$GLOBALS['installed_modules'][$GLOBALS['active_module_id']]['name']`.

Pro uložení ID modulu můžeme použít například konstantu MODULE_ACTUALITIES_ID a pro název MODULE_ACTUALITIES_NAME. Tím máme zajištěno, že při použití těchto konstant v modulu dostaneme vždy správné hodnoty.

Další vhodnou součástí skriptu je načtení dodatečných konfiguračních informací. Konfiguraci jsem umístil do skriptu `config.php`, který nyní v `prologue.php` načteme.

Skript `epilogue.php` je naopak vykonán na konci hlavního skriptu a je vhodné do něj umístit kód pro poslední úpravu dat před výstupem. Pro navržený modul aktualit není třeba tento skript využít, proto ho necháme prázdný.

actualities.php

Hlavním skriptem veřejné části modulu je skript `actualities.php`. Tento skript je zavolán vždy, když uživatel žádá zobrazení uzlu patřícího k modulu aktualit. Zde se skrývá hlavní aplikační logika, která bude zobrazovat potřebný obsah.

První věcí, kterou je třeba udělat je správné nastavení globální proměnné pro definici aktivního modulu. Na základě této hodnoty framework například vyhledává třídy náležející k modulu pro jejich automatické načtení. Nastavení provedeme jednoduše příkazem:

```
$GLOBALS['active_module_id'] = MODULE_ACTUALITIES_ID;
```

V následujícím úseku skriptu vezmeme nezpracovanou část URL, kterou uživatel zadal, postupně procházíme jednotlivé prvky a testujeme, zda existují odpovídající uzly. Pokud ne, prohlásíme URL za neplatnou a vyvoláme výjimku.

Při zpracovávání URL zároveň přidáváme zpracované uzly do menu a vytváříme tak navigaci.

Nakonec vezmeme poslední načtený uzel, který je zároveň uzlem cílovým, a podle jeho typu zobrazíme buď přehled aktualit, nebo konkrétní aktualitu.

administration/actualities.php

Hlavním skriptem administrační části je skript `actualities.php` v podadresáři `administration`.

Stejně jako u předchozího skriptu je nutné správně nastavit proměnnou pro definici modulu. Protože skript je volán vždy při vybrání administrační sekce modulu, je třeba nějakým způsobem určit, jaká akce se má provést. K tomu použijeme GET proměnné v URL. Na základě hodnoty GET proměnné „action“ se ve skriptu pomocí konstrukce `switch` zvolí příslušná sekce.

Pro pokrytí všech potřeb modulu vytvoříme následující akce:

- `new_gateway`
- `edit_gateway`
- `new_actuality`
- `edit_actuality`

První dvě akce slouží v modulu pro vytvoření a editaci stránky přehledu aktualit. Zbylé akce slouží pro vytvoření a editaci aktuality.

Pokud nebude žádná z výše uvedených akcí použita, zobrazí skript přehled všech aktualit.

Tvorba instalátoru

Pro zjednodušení instalace modulu nyní vytvoříme instalační skript. Spuštěním tohoto skriptu potom bude celý modul nainstalován. Skript pojmenujeme `install.php`.

Na začátku načteme konfigurační skript frameworku a soubor s globálními funkcemi. Tím získáme všechny potřebné prostředky pro připojení k databázi a to stačí k instalaci modulu.

Připojíme se tedy k databázi a jako první věc ověříme, zda již není modul nainstalován. Pokud není, vložíme do tabulky modulů nový záznam pro náš modul. Název modulu je „Aktuality“ a jeho název pro soubory je „actualities“.

Vytvoříme v databázi tabulku pro ukládání dat aktualit, kterou jsem popsal výše.

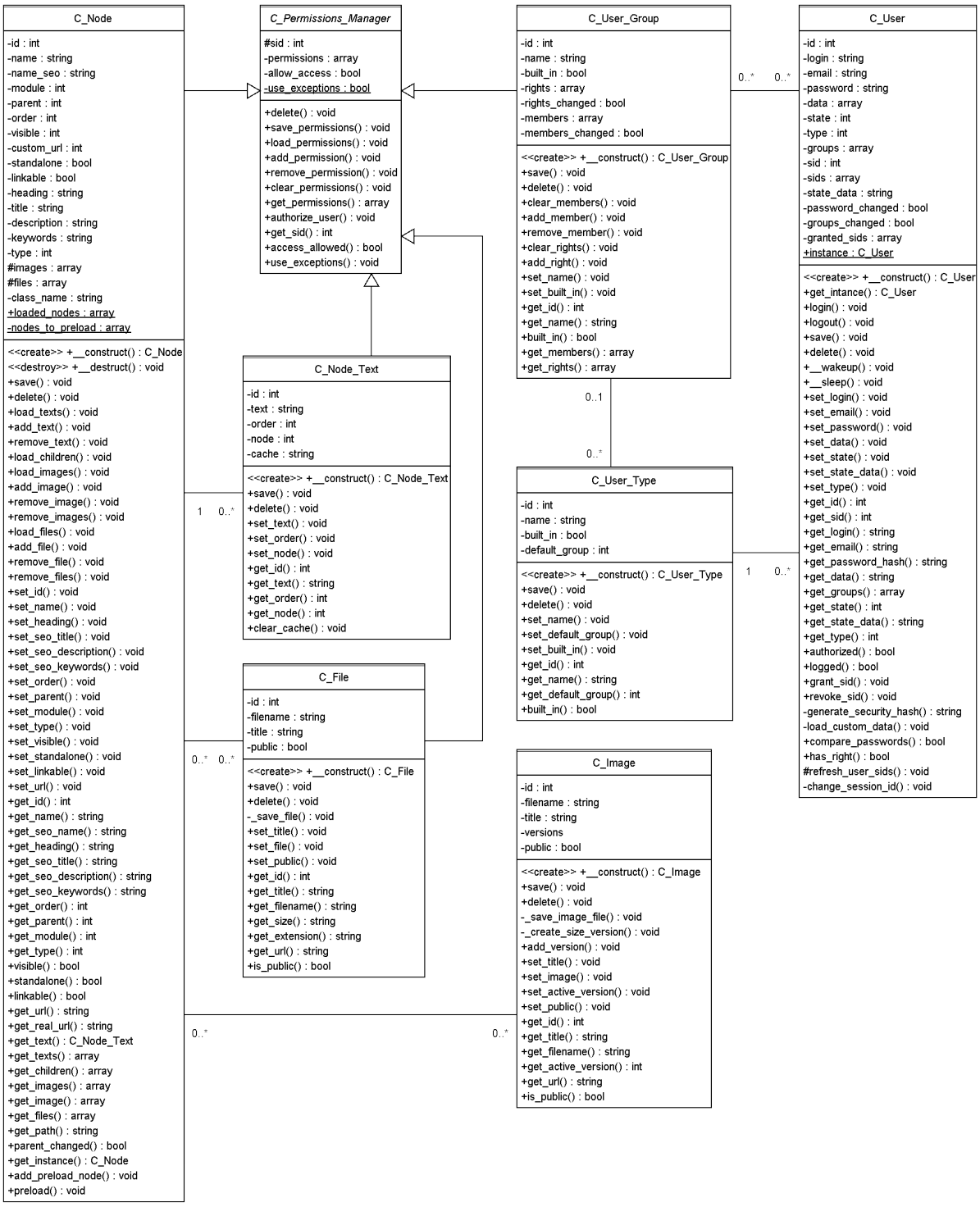
Pro funkci administrační části musíme vytvořit ještě administrační uzel, který zařadíme do administrační sekce a přiřadíme jej modulu aktualit.

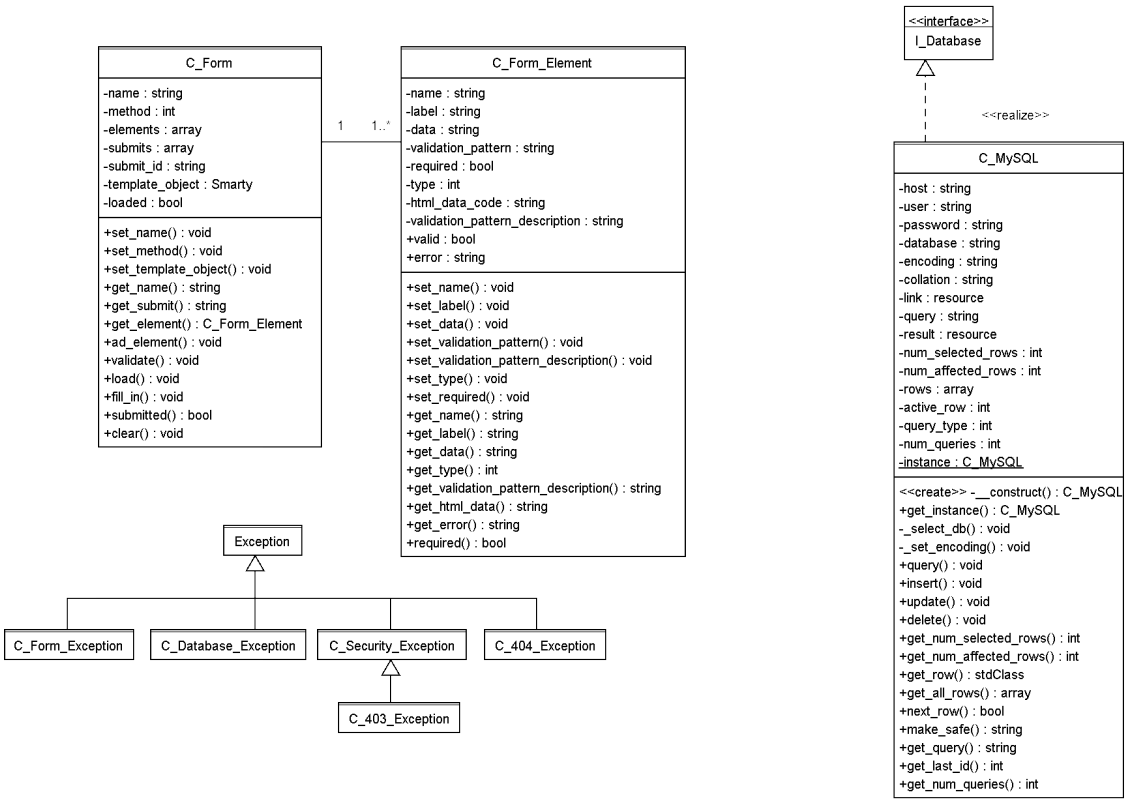
Posledním krokem je vložení popisků do databáze, které potom budou používány v administraci aktualit.

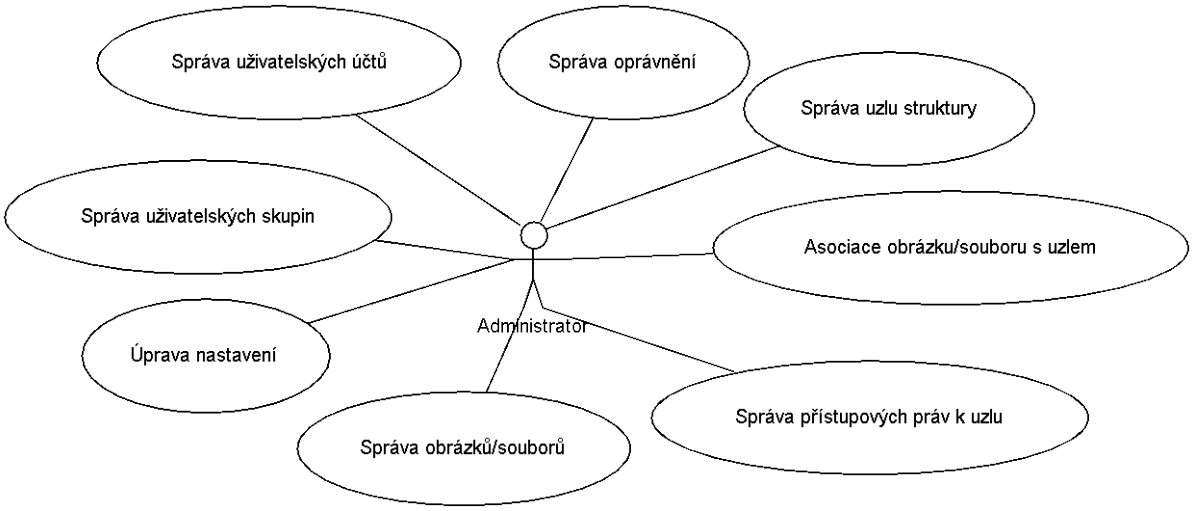
Po spuštění vytvořeného skriptu je modul připravený a může se začít používat. Výše popsaný postup slouží hlavně pro názornost a vytvořený modul plní pouze základní funkce. Pro reálné použití je třeba tvořit moduly podle daleko složitějších požadavků vždy pro konkrétní případ.

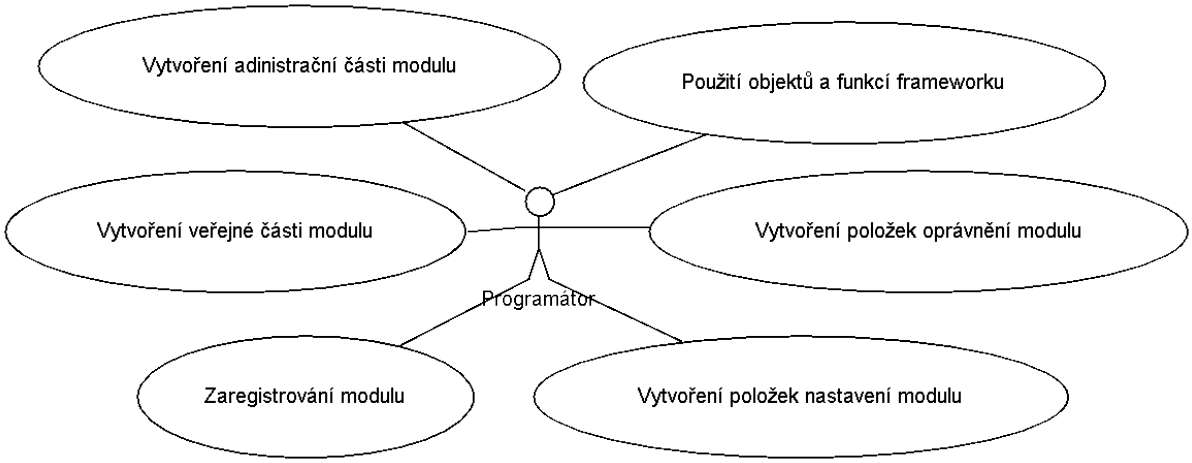
Kompletní modul, který byl předmětem této kapitoly, je pro ukázku již vložen do frameworku.

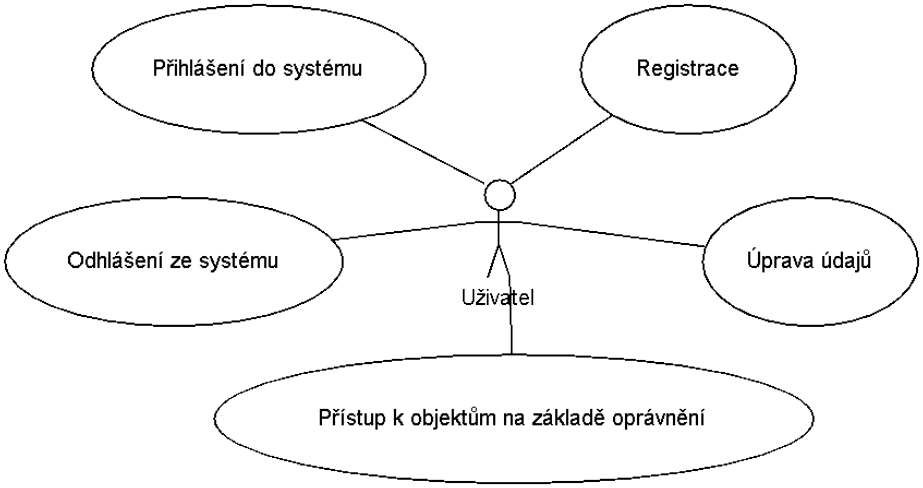
Příloha 4: Class a use-case diagramy




C_Form
-name : string
-method : int
-elements : array
-submits : array
-submit_id : string
-template_object : Smarty
-loaded : bool
+set_name() : void
+set_method() : void
+set_template_object() : void
+get_name() : string
+get_submit() : string
+get_element() : C_Form_Element
+ad_element() : void
+validate() : void
+load() : void
+fill_in() : void
+submitted() : bool
+clear() : void
1
1..*
C_Form_Element
-name : string
-label : string
-data : string
-validation_pattern : string
-required : bool
-type : int
-html_data_code : string
-validation_pattern_description : string
+valid : bool
+error : string
+set_name() : void
+set_label() : void
+set_data() : void
+set_validation_pattern() : void
+set_validation_pattern_description() : void
+set_type() : void
+set_required() : void
+get_name() : string
+get_label() : string
+get_data() : string
+get_type() : int
+get_validation_pattern_description() : string
+get_html_data() : string
+get_error() : string
+required() : bool
<<interface>>
I_Database
<<realize>>
C_MySQL
-host : string
-user : string
-password : string
-database : string
-encoding : string
-collation : string
-link : resource
-query : string
-result : resource
-num_selected_rows : int
-num_affected_rows : int
-rows : array
-active_row : int
-query_type : int
-num_queries : int
-instance : C_MySQL
<<create>> -__construct() : C_MySQL
+get_instance() : C_MySQL
-_select_db() : void
-_set_encoding() : void
+query() : void
+insert() : void
+update() : void
+delete() : void
+get_num_selected_rows() : int
+get_num_affected_rows() : int
+get_row() : stdClass
+get_all_rows() : array
+next_row() : bool
+make_safe() : string
+get_query() : string
+get_last_id() : int
+get_num_queries() : int
Exception
C_Form_Exception
C_Database_Exception
C_Security_Exception
C_404_Exception
C_403_Exception


Správa uživatelských účtů
Správa oprávnění
Správa uzlu struktury
Správa uživatelských skupin
Asociace obrázku/souboru s uzlem
Administrátor
Úprava nastavení
Správa přístupových práv k uzlu
Správa obrázků/souborů


Vytvoření adinistrační části modulu
Použití objektů a funkcí frameworku
Vytvoření veřejné části modulu
Vytvoření položek oprávnění modulu
Programátor
Zaregistrování modulu
Vytvoření položek nastavení modulu


Přihlášení do systému
Registrace
Odhlášení ze systému
Úprava údajů
Uživatel
Přístup k objektům na základě oprávnění