Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE

Martin Svoboda

Informační systém na spolupráci menších skupin uživatelů

Katedra aplikované matematiky

Vedoucí bakalářské práce: **Mgr. Tomáš Bílý**

Studijní program: **Informatika, Programování**

2007

Rád bych poděkoval Mgr. Tomáši Bílému za vedení mého ročníkového projektu, ze kterého vzešla tato bakalářské práce.

Dále bych rád poděkoval skautskému roverskému kmeni Griffins z Liberce, který mi poskytl řadu praxí ověřených poznatků, které se ukázaly stěžejními při návrhu tohoto informačního systému.

V neposlední řadě nesmím zapomenout na všechny, kteří mi v průběhu celé práce byli všestrannou oporou.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů.

Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Liberci dne 31. července 2007 Martin Svoboda

Obsah

Název práce: Informační systém na spolupráci menších skupin uživatelů
Autor: Martin Svoboda
Katedra: Katedra aplikované matematiky
Vedoucí bakalářské práce: Mgr. Tomáš Bílý
Email vedoucího: tomby@kam.mff.cuni.cz

Abstrakt:

Cílem této práce je implementovat konkrétní informační systém na spolupráci a výměnu informací v rámci menších pracovních skupin uživatelů a dále detailněji rozebrat všechny aspekty, které se ukázaly významné při návrhu tohoto informačního systému. Tím se rozumí nejenom návrh struktury poměrně komplexního programu, ale také návrh jeho základních modelů chování a dalších vlastností, které významným způsobem ovlivňují funkcionalitu, kterou pak systém může nabízet svým uživatelům.

Konkrétně jde například o návrh modelu registrace uživatelů, členství v uživatelských skupinách, systému víceúrovňových oprávnění nebo fungování uživatelských relací. Všechny tyto důležité modely jsou v práci rozebrány a jsou diskutovány jejich výhody i případná úskalí. Druhá část práce obsahuje popis implementace všech zajímavých nebo významných částí systému. Podrobněji jsou například rozebírány mechanismy vykonávání databázových dotazů, univerzální stránkované výpisy, editační procesy a vstupy od uživatele, podoba pracovní plochy nebo další funkce jádra systému, stejně jako jsou popsány funkce jednotlivých sekcí systému – plánované akce, diskuzní knihy nebo anketní otázky.

Klíčová slova: informační systém, pracovní skupiny, sdílení informací, plánování

Title: Informational system for cooperation of small groups
Author: Martin Svoboda
Department: Department of Applied Mathematics
Supervisor: Mgr. Tomáš Bílý
Supervisor's email address: tomby@kam.mff.cuni.cz

Abstract:

The main aim of this work is to implement concrete informational system for cooperation and information interchange within small working groups of users and to analyze all aspects which have exposed to be important during the development of this informational system. That means not only to design the structure of relatively complex program, but also to design main concepts of its behaviour and other features, which significantly affect functionality, which is then provided by the system to its users.

For example it comprises the draft of user registration model, membership of those users in groups, multilevel hierarchy of user powers or managing user sessions. All those important models are discussed in this work and there are also analyzed their advantages and potential difficulties. The second part of this work contains the description of implementation of all interesting or important parts of the system. It means for example the mechanisms of database queries execution, generating universal paged listings, editing processes and inputs from users, working desktop layout or other functionality provided by the kernel, as well as the description of system sections such as Actions planning, Discussions, Inquiries etc.

Keywords: informational system, working groups, information sharing, scheduling

1 Úvod

Jistě se nedopustíme velké chyby, pokud prohlásíme, že jedním ze zcela stěžejních pojmů dnešní doby je pojem „informace“. Nejenom z pohledu její znalosti, kvality, pravdivosti nebo úplnosti, ale také tím, jakým způsobem je uchovávána či prezentována a kdo tak činí.

Důkazem důležitosti tohoto pojmu i ve světě informatiky je rozsáhlá teorie kolem informačních nebo databázových systémů, stejně jako neustálá snaha o vývoj nových, výkonnějších a uživatelsky více přívětivých informačních systémů.

Je zřejmé, že čím více je konkrétní informační systém specializovaný a přizpůsobený doslova na míru vybraným požadavkům, tím více bude cílenou skupinou uživatelů využíván a pozitivně hodnocen, ale o to menší tato skupina obvykle bude. Na druhou stranu tato specializace je žádaná a díky ní je možné dosáhnout mnohem kvalitnějších systémů. Univerzální systémy schopné kvalitním způsobem pracovat s informacemi zcela libovolného typu jednoduše nejsou reálné.

Smyslem této práce je implementovat jeden zcela konkrétní informační systém, který by se pokusil v maximální možné míře naplnit požadavky, které s sebou přináší jeho předpokládané nasazení. Plánovaným cílem je tedy poskytnutí kvalitního, snadno přístupného a jednoduchého nástroje, který umožní menším skupinám uživatelů interaktivním způsobem vzájemně vyměňovat informace, které se týkají činnosti takovéto skupiny.

Pokud má být informační systém kvalitní, je potřeba věnovat velkou pozornost jeho návrhu. A to nejenom na úrovni vybrané platformy, struktury implementovaného kódu, ale především také požadované funkčnosti z pohledu koncového uživatele. Pokud systém nedokáže dobře vystihnout potřeby a chování uživatelů, nelze očekávat jeho úspěšné využívání.

Významná část textu této práce je proto věnována návrhu systému – popisu vybraných modelů, odůvodněním těchto výběrů, vysvětlením souvislostí a případným zmínkám o alternativních řešeních. Druhá část textu pak popisuje konkrétní funkčnost jednotlivých implementovaných částí tohoto systému.

2 Koncepce projektu

2.1 Definování cílů projektu

Cílovými skupinami uživatelů, pro které je tento informačního systém určen, jsou obecně skupiny lidí, které spojuje jejich vzájemná a zejména pravidelná činnost. Primární potřebou těchto lidí není prezentovat svou činnost navenek, ale jejich základním požadavkem je kvalitní vzájemná komunikace, která nemůže být realizována osobním kontaktem. Díky časovým možnostem těchto lidí nebo například kvůli jejich velké vzájemné vzdálenosti.

2.1.1 Vymezení cílových uživatelů

V principu tedy není uživatelem tohoto systému jednotlivec, ale již zmíněné celé skupiny lidí. Tyto skupiny jsou poměrně jasně vymezeny. Zejména prováděnou činností, ale třeba také vnitřním smýšlením těchto skupin nebo jejich společenskými vztahy. Přirozeně se proto dá mluvit o pojmu uživatelských skupin.

Tyto skupiny na jedné straně mohou být téměř až uzavřeny do sebe sama a jejich uživatelé nemají potřebu jakýmkoli způsobem komunikovat s jinými skupinami. Na druhé straně skupiny jiného charakteru mohou mít naopak potřebu se s jinými podobně zaměřenými skupinami určitým způsobem spojovat a navzájem se sobě otevřít.

Návrh systému předpokládá, že každá jeho skutečná instalace bude sloužit několika takovýmto skupinám. Můžeme mluvit například o skupinách s nějakým pracovním zaměřením, partách lidí s určitými společnými zájmy, organizačních složkách nějakých velkých občanských sdružení či nejrůznějších jiných kolektivů. V každém případě všechny tyto skupiny v dané instanci systému něco spojuje. Nepředpokládá se proto, že by jedna instance sloužila masovému množství skupin se zcela odlišnými zájmy.

Předpokladem na straně konkrétního uživatele je pak skutečnost, že jeho zájmem není škodit skupině, které je členem. Něco takového si nemůže dovolit, protože skupina sama ho dostatečně kontroluje. Tento předpoklad samozřejmě nesouvisí se zajištěním bezpečnosti systému, ale týká se spíše způsobu chování předpokládaných uživatelů.

2.1.2 Požadavky na funkce systému

Jak již bylo řečeno, celý systém má tvořit komplexní prostředí, které jednotliví členové mohou využívat ke své vzájemné komunikaci. Tím se ovšem nemyslí obyčejná komunikace například ve formě suplování elektronické pošty, ale poskytnutí ucelenějších a zcela konkrétně zaměřených nástrojů, které přímo řeší příčiny, jež potřebu dané komunikace vynucují.

Zmíněnou příčinou je potřeba výměny nejrůznějších informací, které souvisejí s činností dané skupiny. Může jít o plánování společných akcí, sdílení kontaktních

údajů, diskutování o společných problémech, výměny názorů, různá hlasování ale třeba také pohled do minulosti formou zhodnocení proběhlé činnosti či sdílení fotografií z proběhlých akcí.

Tyto skupiny tudíž nežijí jenom v prostředí tohoto informačního sytému, ale v reálném světě, který jim ovšem nedává dostatečné prostředky na to, aby všechny tyto potřeby byly vyřešeny například cestou osobního kontaktu. Cílem tohoto informačního systému je pokusit se dát těmto lidem ve zmíněném smyslu podpůrné zázemí.

Dá se očekávat, že jednotliví uživatelé ve skupinách se rozdělí na aktivnější část, která neustále produkuje nové informace a cítí potřebu jejich distribuce ostatním, a zbylou pasivnější část, která naopak tyto informace konzumuje. Smyslem celé skupiny je rozvoj její činnosti v dlouhodobé perspektivě, a proto tento informační systém musí splňovat takové chování, které nové informace od té aktivnější části vnucuje té pasivnější.

Systém musí obsahovat poměrně silné funkce, které dokáží nejrůznějšími cestami tyto informace předávat zejména těm, kteří by sami o sobě neměli potřebu je nějakými složitými cestami vyhledávat. Jinými slovy musí jít o takové prostředí, kde každá aktuální informace bude snadno dostupná, aby uživatelé nebyli odrazeni už předem. Pokud se tyto cíle podaří dobře naplnit, stane se celý systém každodenní součástí běžného života členů daných skupin a bude se sám o sobě výrazným způsobem podílet na udržení síly takovéto skupiny.

Neméně významným požadavkem je jednoduchost systému. Přesněji řečeno, systém by měl umožňovat provádět netriviální věci, ale uživatel by neměl být příliš zatěžován. Měl by mít pocit, že práce v systému je jednoduchá, a především pak pocit, že svým neodborným zásahem nemůže nic zkazit. To mimo jiné znamená, že systém musí dostatečně kvalitně kontrolovat všechno, o co se jednotliví uživatelé pokoušejí.

2.2 Metody dosažení vytyčených cílů

Cílem popisovaného informačního systému má být výměna aktuálních informací v rámci daných skupin uživatelů. Z tohoto důvodu je nutné, aby samotná platforma, na které bude systém implementován, poskytovala takové prostředky a splňovala takové vlastnosti, které si vynucují samotné důvody využívání systému.

2.2.1 Odůvodnění výběru platformy webu

Pokud má systém dobře splňovat vytyčené cíle, musí být velmi dobře dostupný. V ideálním případě by to mělo znamenat nejenom dostupnost ze standardních míst, kde se uživatelé dennodenně pohybují, ale také ve výjimečných situacích i odjinud.

To poměrně jasně vede k takovému řešení, které nevyžaduje instalace žádných zvláštních programů na straně klienta. Pokud by se totiž uživatel dostal do výjimečné situace, nemusel by mít k dispozici potřebné instalační nástroje a koneckonců by nemusel mít na daném počítači oprávnění takový program vůbec nainstalovat.

Volba klasické webové platformy je proto řešením, které se doslova nabízí. Uživatel není zatížen nutností instalace žádných speciálních programů, může se do informačního systému přihlásit odkudkoliv a navíc může pracovat v prostředí webových prohlížečů, na které je už dostatečně dobře navyklý.

2.2.2 Základní popis funkcí systému

V souladu s předchozím pojednáním má systém nabídnout řadu konkrétních a specializovaných nástrojů, které budou sloužit na požadovanou výměnu informací na vyšší úrovni.

Je otázkou, jak by tento výběr nástrojů měl vypadat. Je totiž zřejmé, že takový výběr může být poměrně silně závislý na konkrétních uživatelích, kteří budou systém využívat. Zatímco jeden nástroj může být pro jednu skupinu velmi cenný, pro druhou skupinu bude téměř nepoužitelný.

Všechny konkrétní nástroje nebo modely chování, které byly nakonec vybrány, jsou detailně rozebrány ve zbylých kapitolách této práce. Namátkou ale lze jmenovat například následující:

Registrace uživatelů, jejich členství ve skupinách, přidělování konkrétních oprávnění, vyhledávání novinek, plánování akcí, potvrzování účasti na akcích, diskutování v knihách a kapitolách, umisťování souvisejících objektů do složek, nahrávání přiložených souborů, hlasování v anketních otázkách, umisťování informací na nástěnku, … a řada dalších funkcí a vlastností.

2.3 Výběr použitých technologií

Vybraná platforma webových stránek předpokládá architekturu založenou na modelu klient/server. Serverová část v tomto případě představuje celou logiku chování systému, přijímá standardní cestou požadavky od klienta a na jejich základě generuje klasické webové stránky, které jsou určeny pro interpretaci klientem.

Na klientské straně uživatel používá svůj oblíbený internetový prohlížeč, který dokáže stránky získané ze serveru interpretovat. Samotné stránky jsou napsány v jazyce XHTML a pro jejich zobrazení se využívají šablony stylů CSS. V drobné míře je také využíván java script.

Na serverové straně byl vybrán skriptovací jazyk PHP, ačkoli ten samozřejmě není jedinou možnou alternativou, která se nabízí. Jako databázová technologie bylo vybráno MySQL. Obě tyto technologie zjednodušeně řečeno umožňují volné nekomerční použití, což je v souladu s cíly celého projektu. Jazyk PHP obsahuje řadu praktických doplňků a v neposlední řadě kombinace PHP a MySQL je prakticky ověřenou záležitostí.

Při vývoji byl použit webový server Apache 2.0.54 nad Win32, nicméně žádné jeho speciální funkcionality nebylo využito, a proto není vyloučen provoz systému i nad jiným severem.

Konkrétní verze použitých technologií

- PHP 5.1.6 – kvůli maximálnímu využití objektového modelu PHP verze 5 nelze počítat s možným nasazením nad PHP 4.
- MySQL 5.0.22 – ačkoli v této verzi už jsou obecně podporovány transakce, nejsou podporovány nad typem tabulek MyISAM, který se musel využít proto, že na rozdíl od InnoDB zase umí pracovat s full-textovými idexy. Rovněž není podporována kontrola integritních omezení ve formě cizích klíčů, přesto jsou tyto závislosti v definicích tabulek zaneseny.
- XHTML 1.0 Strict

- CSS 2.1
- JavaScript – použit byl pro drobnou podporu interaktivity na straně klienta. Pokud prohlížeč klienta tento jazyk plně nebo vůbec nepodporuje, není to na úkor funkčnosti systému.
- RSS 2.0 – využito pro syndikování novinek a změn v systému.

Použitá rozšíření PHP

Kromě standardních součástí PHP byla využita i řada takových, které nejsou součástí běžných distribucí. Jejich získání a začlenění na běžící server ale není nijak složité a lze k tomu využít mechanismy PEAR.

Jmenovat lze například rozšíření EXIF, FileInfo, MIME_magic, ZIP a další.

3 Aspekty návrhu systému

3.1 Popis struktury systému

Především z bezpečnostních důvodů je celý balíček zdrojových kódů distribuce rozdělen na dvě samostatné části – na veřejnou a skrytou.

Veřejná část balíčku je umístěna v takové části webového serveru, která je veřejně přístupná. To prakticky znamená, že každý klient může přímo indexovat jakýkoli soubor, který je v takovéto adresářové struktuře umístěn. Některé servery umožňují selektivně vybrat například různé podadresáře, do kterých je možnost onoho přístupu omezena, nebo úplně zakázána.

Stejné funkce skrytí některých souborů lze docílit i tím, že se umístí do takového místa, které není standardně viditelné mimo server. Dá se říci, že toto řešení je i o něco rychlejší, protože logiku skrývání řeší aplikace sama a není jí zatížen webový server. Pouze je potřeba zařídit, aby webový server měl k oběma místům dostatečná přístupová práva.

Jak již bylo řečeno, veřejný balíček obsahuje všechny ty soubory, u nichž nevadí, pokud se uživatel dostane k jejich obsahu – a to i pokud by se snažil provést nějaký útok a nešlo by o přímý přístup vyžádaný chováním samotné aplikace. Konkrétně tedy balíček obsahuje veškeré obrázky grafických prvků plochy aplikace, stejně jako definice kaskádových šablon stylů a java scriptové zdrojové kódy na podporu interaktivního chování aplikace na straně klienta.

Dále veřejný balíček obsahuje brány, přes které jsou přístupné funkce skrytého balíčku. V podstatě jde o triviální soubory, které pouze obsahují direktivy `include` příslušných souborů na straně skrytého balíčku. Podstatné je to, že tímto způsobem se docílí skrytí skutečných zdrojových souborů, byť na úkor drobného zpomalení.

Pokud by toto rozdělení nebylo vyžadování, lze oba balíčky velmi jednoduchým způsobem sloučit a umístit je společně do jednoho veřejně přístupného místa.

3.1.1 Koncepce jádra, sekcí a modulů

Jak z pohledu chování aplikace, tak i z pohledu uspořádání zdrojových kódů ve skrytém balíčku, je celý systém rozdělen do několika sekcí, které buď tvoří skutečnou část stránek z pohledu uživatele (například sekce plánovaných akcí, diskuzních knih, …), nebo jde jen o fiktivní systémové sekce, které slouží k přehlednějšímu uspořádání kódu jádra systému (například sekce logování či monitorování činnosti).

Původní vize při návrhu celého systému byla, aby jednotlivé sekce tvořily zcela samostatné a oddělitelné balíčky kódu. To by umožňovalo elegantní přidávání či odebírání sekcí z konkrétní instalace systému – správce by si jednoduše vybral ty sekce, které by chtěl provozovat. Je zřejmé, že alespoň minimálního propojení kódu by bylo kvůli způsobu chování samotného PHP nutné, nicméně později se ukázalo, že takto

striktní oddělení by bylo značně na úrok funkčnosti systému resp. efektivitě jeho pracování.

Z tohoto důvodu jednotlivé sekce sice z logického pohledu tvoří ucelené balíčky, ale určitý jejich kód je přímo implementován v souborech jádra. Stejně tak jednotlivé sekce o sobě svým způsobem vědí, což umožňuje různé propojování a využívání funkčnosti jiných sekcí. Typickou ukázkou tohoto jevu jsou například diskuzní komentáře, kde moduly těchto komentářů implementuje sekce diskuzí, ale jejich funkčnost využívá například sekce akcí či nástěnky.

Úplné oddělení kódu jednotlivých sekcí by sice bylo možné, ale velmi výrazným způsobem by se zvýšila časová náročnost vykonávaného kódu.

Přehled funkcí jádra

Využijeme-li terminologie sekcí z výše uvedeného textu, několik z nich tvoří jádro celého systému. Především jde o sekce `system` a `profiles`. Spolu s ostatními sekcemi jádra vytváří společné minimum funkčnosti a návrhu celého sytému.

Zbytek této kapitoly je věnován popisu základních návrhových konceptů, které zcela zásadním způsobem určují chování celého systému. Celá další kapitola je pak věnována popisu prostředků, které jádro systému ostatním sekcím nabízí.

Jen pro stručnou informaci můžeme uvést například podporu databázových dotazů, prvků editačních formulářů na zpracování vstupů od uživatele, zamykání aplikačních objektů, nebo například nástroje na základní monitorování a logování činnosti systému.

Implementované sekce

Kromě jádra obsahuje systém celou řadu dalších sekcí, které naplňují samotné cíle celého projektu. Jejich podrobný popis je uveden v samostatné kapitole. Jako příklad těchto sekcí lze uvést plánované akce, anketní otázky, fotografická alba či diskuzní knihy.

Všechny tyto sekce jsou přímo vystavěny na funkcionalitě, kterou nabízí jádro systému. To s sebou přináší celou řadu výhod a jen malé množství nevýhod. Například implementace velmi univerzálních prvků editačních formulářů se podobně jako ostatní služby jádra velmi osvědčila (a to i přes výrazný nárůst jejich složitosti), neboť ostatní sekce pak tyto třídy mohou velmi snadno a elegantně využívat.

Podoba modulů

Modulem se rozumí samostatný skript sekce, který kompletně realizuje požadavek klienta. Například může jít o skript vypisující nalezené akce daného kalendářního roku, přehled fotografií alba, stejně jako skript zajišťující kompletní průběh editačního procesu.

Obecně není nutné, aby každý modul byl umístěn v samostatném souboru. Typicky tomu také tak není a jednotlivé soubory shlukují více tématicky podobně zaměřených modulů najednou.

Z implementačního hlediska je většina modulů realizována jako třída (potomek abstraktního předka všech modulů) se statickou metodou `main()`, jejíž kód se v okamžiku zpracovávání modulu začne vykonávat.

Systém zároveň poskytuje zjednodušený koncept modulu, který pouze očekává umístění kódu daného modulu přímo v těle určeného zdrojového souboru. Pokud má takovýto soubor obsahovat více modulů, musí se daný soubor sám postarat o vlastní

logiku výběru, přičemž k tomu má k dispozici přístup ke globálním proměnným obsahujícím informace o právě prováděném požadavku klienta.

3.2 Zpracovávání požadavků klienta

Požadavky na zpracovávání skriptů jsou od uživatele přijímány standardní cestou, tedy parametry metody GET protokolu HTTP. Obvykle se nepřepokládá, že by uživatel sám parametry v adrese požadované stránky měnil. Typicky se dostane na výchozí stránku systému a pak už jen využívá odkazy, které sám systém vygeneroval.

Je však nutné podotknout, že celý systém je navrhnut takovým způsobem, aby řádně fungoval i v případě, kdy by se ho někdo touto cestou pokoušel napadnout. V tuto chvíli můžeme zjednodušeně říci, že systém vždy analyzuje takovýto požadavek klienta, ale k vlastnímu vykonávání příslušného skriptu dojde pouze v případě, že je tento požadavek regulérní.

Pokud není požadavek klienta vyhodnocen jako korektní, zopakuje se zpracování posledního regulérního požadavku (takového, který nezpůsobil žádné změny systému).

Spolu s již diskutovaným rozdělením zdrojových kódů systému na veřejný a skrytý balíček tak můžeme mít jistotu, že uživatel nemá možnost nepříznivě ovlivnit činnost systému tím, že by například spekulativně vykonával kód, který k tomu není určen.

3.2.1 Analýza požadavku uživatele

V souladu s rozdělením systému na samostatné sekce, bylo snahou, aby jádro systému nepožadovalo příliš velké konkrétní znalosti o každé jednotlivé sekci. Nicméně alespoň fakt o existenci jednotlivých sekcí bylo nutné do jádra začlenit – s ohledem na již zmíněný prvek bezpečnosti v předchozím odstavci.

Součástí kódu jádra je tak jakási registrace všech sekcí, které jsou v systému aktivní. V okamžiku analýzy požadavku uživatele na zpracování stránky se nejprve zjistí, k jaké sekci daný požadavek přísluší. Pokud se už v této chvíli zjistí, že požadovaná sekce není korektní, celý požadavek se odmítne. V opačném případě se načte hlavičkový soubor příslušné sekce, ze kterého jádro získá detailní přehled o tom, jaké moduly daná sekce poskytuje.

Teprve v tento okamžik systém definitivně rozhodne, jestli je požadavek klienta akceptovatelný. Pokud ano, je tento zařazen do fronty čekajících skriptů (viz dále), jinak je ignorován a tato fronta zůstane beze změny.

Za běhu systému jsou tak načteny všechny potřebné zdrojové soubory jádra a hlavičkové soubory jenom některých sekcí. Jak totiž bude vysvětleno později, těchto sekcí může být v jednom okamžiku více hned z několika důvodů.

Tím prvním z nich je následující vlastnost návrhu celého systému. Jak již bylo řečeno, bylo požadováno, aby jednotlivé sekce měly možnost využívat funkcionality ostatních sekcí. A to nejenom na úrovni deklarací například tříd či funkcí, ale především na úrovni celých modulů. Klient tak ve svém požadavku specifikuje nejenom jaký modul jaké sekce požaduje, ale navíc ještě aktivní sekci, která může být obecně různá od té, ze které je požadovaný modul.

Díky tomuto konceptu je možné docílit například takového chování, aby sekce jako akce či nástěnka mohly kompletně plnohodnotně využívat modul sekce diskuzí na psaní diskuzních komentářů, ale uživatel měl stále představu, že je v sekci například zmíněných akcí (zobrazené hlavní menu bude patřit sekci akcí, nikoli diskuzí).

3.2.2 Vykonávání naplánovaných skriptů

Stěžejní v celém návrhu vykonávání uživatelem požadovaných skriptů je plánovač skriptů. Při každém spuštění systému je nejprve analyzován zmíněný požadavek uživatele. Pokud je akceptovatelný, není hned automaticky zpracován, ale uložen nejprve do zmíněného plánovače skriptů.

Ten je víceméně reprezentován jako obyčejný zásobník. Poslední naplánovaný skript je umístěn na vrchol tohoto zásobníku a je určen ke zpracování jako první. Odchylka je pouze v tom, že jakmile je informace o úloze již nepotřebná, je automaticky ze dna vymazána.

Dalším krokem při běhu systému po analýze požadavku klienta je kontrola bezpečnosti relace přihlášeného uživatele. Tato kontrola může vyústit ve vynucené ověření identity uživatele. Například pokud byl příliš dlouhou dobu nečinný. Tato kontrola není naplánována nijak jinak, než že se přidá požadavek na vykonání příslušeného modulu na vrchol zásobníku, čímž je překryt podavek, který vyslovil klient.

Po všech úvodních inicializacích se běh spuštěného sytému dostane do fáze zpracovávání naplánovaných skriptů. Typicky tato část proběhne takovým způsobem, že je zpracován skript na vrcholu zásobníku a další zpracovávání je ukončeno. Systém nicméně rozeznává hned 3 zcela odlišná schemata vykonávání skriptů. Standardně je očekáváno zmíněné chování, ale konkrétní schéma si může každý jednotlivý modul vybrat sám.

Jednotlivá schemata se liší především v tom, jestli generují nějakou stránku pro klienta, nebo jestli se pouze vykonají bez jakéhokoli výstupu. V prvním případě se očekává, že plánovač svou činnost ukončí, zatímco v druhém případě plánovač naopak začne vykonávat další naplánovanou úlohu. Druhým rozdílem je to, jakým způsobem se plánovač po vykonání dané úlohy zachová ke zbytku zásobníku směrem k jeho dnu.

Následuje podrobný popis chování jednotlivých schémat:

- `EXECUTION_SCHEMA_DEFAULT`
  - Modul vyprodukuje výstup pro klienta, plánovač je proto ukončen a právě zpracovaný modul zůstane na vrcholu zásobníku jako kotva, všechny ostatní položky zásobníku se vymažou.
  - Příkladem takovéhoto chování je výpis kapitol diskuzní knihy, zobrazení informací o plánované akci apod.
- `EXECUTION_SCHEMA_PHANTOM`
  - Modul neprorokuje žádný výstup pro klienta, plánovač není přerušen a následně pokračuje další naplánovanou úlohou, právě zpracovaný skript je z vrcholu zásobníku odstraněn a zbytek zásobníku zůstává beze změny.
  - Příkladem může být hlasování v anketní otázce. Uživatel klikne na požadovanou odpověď v anketě, modul na hlasování realizuje samotné hlasování a plánovač pokračuje zpracováním dalšího skriptu, což je

typicky modul na zobrazení anketní otázky, který tam byl ponechán jako kotva a který sám byl vykonán předchozím schématem.

- `EXECUTION_SCHEMA_INVADER`
  - Modul produkuje výstup pro uživatele, plánovač se ukončí, ale právě zpracovaný skript je smazán z vrcholu zásobníku a zbytek zásobníku není dotčen.
  - Příkladem může být již zmiňované vynucené ověření identity přihlášeného uživatel po delší době nečinnosti. Rovněž všechny editační procesy fungují na tomto principu.

I když se jednotlivá schemata chovají k zásobníku odlišným způsobem, jejich chování je vždy korektní v tom smyslu, že na dně zásobníku zůstává úloha, která sice už jednou zpracována byla (vždy výchozím schématem), ale u níž její opětovné zpracování nevadí.

Navíc samotné schéma nijak tento proces vykonávání neovlivňuje a teprve po skončení dané úlohy ovlivňuje to, jak se má pokračovat dál. To znamená, že o výběru schematu rozhoduje samotný modul – a to dokonce až v jeho průběhu a tudíž v závislosti na něm.

Díky tomu je možné například zařídit, aby v případě, kdy klient chce hlasovat v anketní otázce a dané hlasování je oprávněné, se vrátil podle druhého schematu zpět na stránku se zobrazením ankety, ale v případě, kdy dojde k nějaké chybě, byl podle třetího schematu vyrozuměn chybovým hlášením.

3.3 Uživatelé a uživatelské skupiny

Potřeba identifikovat jednotlivé uživatele při jejich práci v systému se už od samotného počátku vývoje jevila jako stěžejní, a tudíž celý model registrace uživatelů a jejich členství byl zařazen přímo do jádra systému, nikoli jako přídavná sekce.

Obecně se předpokládá, že pracovat v systému může i neregistrovaný anonymní veřejný návštěvník. V některých případech se dokonce takový uživatel přímo očekává, ale takových situací je velmi málo. Anonymní návštěvník tudíž v systému sice pracovat může, ale kromě prohlížení veřejných věcí nemůže dělat vůbec nic. Veškerá oprávnění spojená s vytvářením nových objektů, ať už jsou jakékoli, nebo jejich následnou editací, jsou vždy spojena s konkrétními uživateli, kteří jsou řádně registrováni.

Navíc ve většině případů veškerá činnost daného přihlášeného uživatele je prováděna pod jeho jménem, pouze výjimečně vystupuje anonymně a v tomto případě je jeho úplná anonymita skutečně zaručena. Příkladem může být hlasování a anketních otázkách, které jsou privátního a anonymního typu.

3.3.1 Registrace nového uživatele

Registrace je otevřena každému návštěvníkovi, který o to na projeví zájem. Z tohoto důvodu samotná registrace nemůže automaticky znamenat dramatické zvýšení uživatelských oprávnění, o kterých bude podrobněji řeč v následující podkapitole. Pokud nebude určitým způsobem registrace nového uživatele akceptována, jeho oprávnění budou téměř na stejně nízké úrovni, jako kdyby registrován nebyl.

Nicméně registrací získá daný uživatel v systému trvalou identitu, která je sama o sobě výhodou. Představit si můžeme uživatele, který sice nebude přijat do žádné skupiny, ale tím, že je registrovaný, bude mít možnost pohodlným způsobem sledovat obsah systému, vyhledávat novinky apod. Tudíž registrace má smysl i pro ty uživatele, kteří ani nemohou očekávat přijetí do žádné skupiny. Navíc existence takovýchto uživatelů ty ostatní s vyššími oprávněními nijak neovlivňuje a ani o nich neví.

Samotná registrace nového uživatele probíhá přes standardní editační formulář a pokud proběhne úspěšně až do konce, stane se z daného návštěvníka řádný uživatel systému. Během registrace je nový uživatel dotazován na řadu registračních, kontaktních a osobních údajů, z nichž ovšem jen minimum je povinných. Jen pro zmínku uveďme, že přihlašovací jméno je na rozdíl od většiny běžných systémů možné kdykoli v budoucnosti změnit, protože slouží pouze a jenom k účelu přihlašování.

3.3.2 Model uživatelských skupin

Uživatelské skupiny nejsou nic jiného, než skupiny registrovaných uživatelů. Model skupin použitý v tomto systému předpokládá, že každý uživatel může být členem zcela libovolného množství těchto skupin. Žádné z nich není nijak preferováno a je plnohodnotné vůči ostatním.

Navíc systém umí pracovat se skupinami s ohledem na tok času – pokud je tedy nějaký uživatel členem dané skupiny, je jejím členem od nějakého konkrétního časového okamžiku – a stejně tak už členem být nemusí, protože členství mohlo být k nějakému konkrétnímu datu ukončeno. Toto časové pojetí je také poměrně zásadní pro uživatelská oprávnění diskutovaná později.

Dalším rozšířením jednoduchého modelu jsou tzv. nadskupiny. V praxi se totiž můžeme setkat se situací, kdy několik uživatelských skupin má smysl od sebe navzájem oddělit, nicméně ne tak, aby o sobě uživatelé těchto skupin navzájem nevěděli. Tuto funkcionalitu nabízí právě nadskupiny. Rozlišují se dva druhy nadskupin, z nichž první způsobí právě zmíněnou viditelnost, druhý tuto viditelnost nepřináší a slouží pouze ke skrytému seskupení. Obecně o sobě uživatelé různých skupin neví.

Jak již bylo řečeno dříve, dokončení registrace uživateli neposkytne žádnou zvýšenou úroveň oprávnění a činnost, kterou může v systému provádět, je prakticky srovnatelná s tou, kterou může dělat anonymní návštěvník. Pokud chce uživatel více, musí se stát členem nějaké uživatelské skupiny.

To předpokládá podání žádosti o členství ve vybrané uživatelské skupině. Každá skupina má alespoň jednoho svého správce, který o akceptaci takto podaných žádostí rozhoduje. Stejným způsobem tito správci rozhodují o vyloučení členů ze skupin.

3.4 Hierarchie oprávnění a licence

Systém pracuje se dvěma modely uživatelských oprávnění zároveň. První hierarchie oprávnění je určena pro typické objekty, které se v systému dají vytvářet. Můžeme mluvit například o plánovaných akcích, pracovních složkách nebo fotografických albech. Tato oprávnění ve svém důsledku říkají, kteří konkrétní uživatelé mají právo s daným objektem daný typ činnosti provádět.

Druhým použitým modelem jsou licence. Oproti výše uvedenému modelu představují oprávnění, která jsou časově omezena, a uživatel s nimi může disponovat

pouze v okamžiku, kdy je jejich aktuálním vlastníkem. Příkladem použití může být například licence administrátora systému.

3.4.1 Standardní hierarchie oprávnění

Tento model oprávnění je určen pro použití v souvislosti s již zmíněným typem objektů v systému. Autor takovéhoto objektu (nebo později při editaci každý uživatel mající oprávnění delegovat oprávnění) má možnost zcela konkrétně vyjmenovat všechny uživatele, kteří budou mít oprávnění daný typ činnosti s objektem provádět.

Oprávnění uživatelů a skupin

Důležité ovšem je, že při této delegaci nemusí jmenovat přímo konkrétní uživatele, ale může jmenovat i konkrétní uživatelské skupiny. V takovémto případě si může vybírat ze všech uživatelských skupin, které vidí. Tj. je přímo jejich členem nebo jde o spřátelené skupiny ve smyslu umístění ve stejné nadskupině, jak bylo diskutováno dříve. Z takto viditelných skupin rovněž vyplývá seznam konkrétních uživatelů, který je k dispozici na přidělování jmenovitých oprávnění.

Přidělení jmenovitých oprávnění všem členům vybrané skupiny resp. dané skupině není totéž, jak by se na první pohled mohlo zdát. Pokud je oprávnění přiděleno konkrétnímu uživateli, je toto přidělení trvalé a spojené pouze a jenom s daným uživatelem. V druhém případě je ale přidělené oprávnění spojeno se skupinou. A členství uživatelů ve skupinách není záležitostí, která by byla v čase neměnná.

Každý objekt má proto přiděleno jakési referenční období, které je v tomto smyslu rozhodující. Tímto obdobím je například u plánovaných akcí období začínající dřívějším z okamžiků vytvoření akce nebo jejím začátkem a končící pozdějším z okamžiků vytvoření akce nebo jejího konce konání. Stejnou úroveň oprávnění, jaká byla přidělena vybrané skupině, potom získají všichni uživatelé, kteří v průběhu daného referenčního období alespoň na chvíli byli členy dané uživatelské skupiny.

To s sebou přináší řadu vlastností, které mohou být v určitých situacích posuzovány jako požadované, stejně jako v jiných situacích jako nechtěné. Představme si následující situaci. Vytvoříme plánovanou akci, která se koná někdy v budoucnosti, a přidělíme určitá oprávnění vybrané skupině uživatelů. Posléze se do systému zaregistruje nový uživatel a je přijat do stejné skupiny. Ačkoli tento uživatel nebyl členem dané skupiny v okamžiku vytvoření akce, příslušná oprávnění získá, protože referenční období je určitě alespoň částečně členstvím překryto.

Je otázkou, do jaké míry je toto chování rozumné. Jistě by se našly situace, kdy by vhodné nebylo, nicméně systém přepokládá bezproblémové uživatele a jejich typicky otevřené chování, a tudíž byl zvolen právě tento přístup k problému.

Víceúrovňová hierarchie

Samotná oprávnění pak tvoří jakýsi víceúrovňový model. Typický objekt v systému používá 4-vrstvou hierarchii, ale obecně je systém připraven na více úrovní. Přesněji řečeno, každý typ objektu může využívat svou vlastní hierarchii a je tedy čistě na rozhodnutí dané sekce, jaké jsou její potřeby.

Obvykle můžeme předpokládat následující hierarchii:

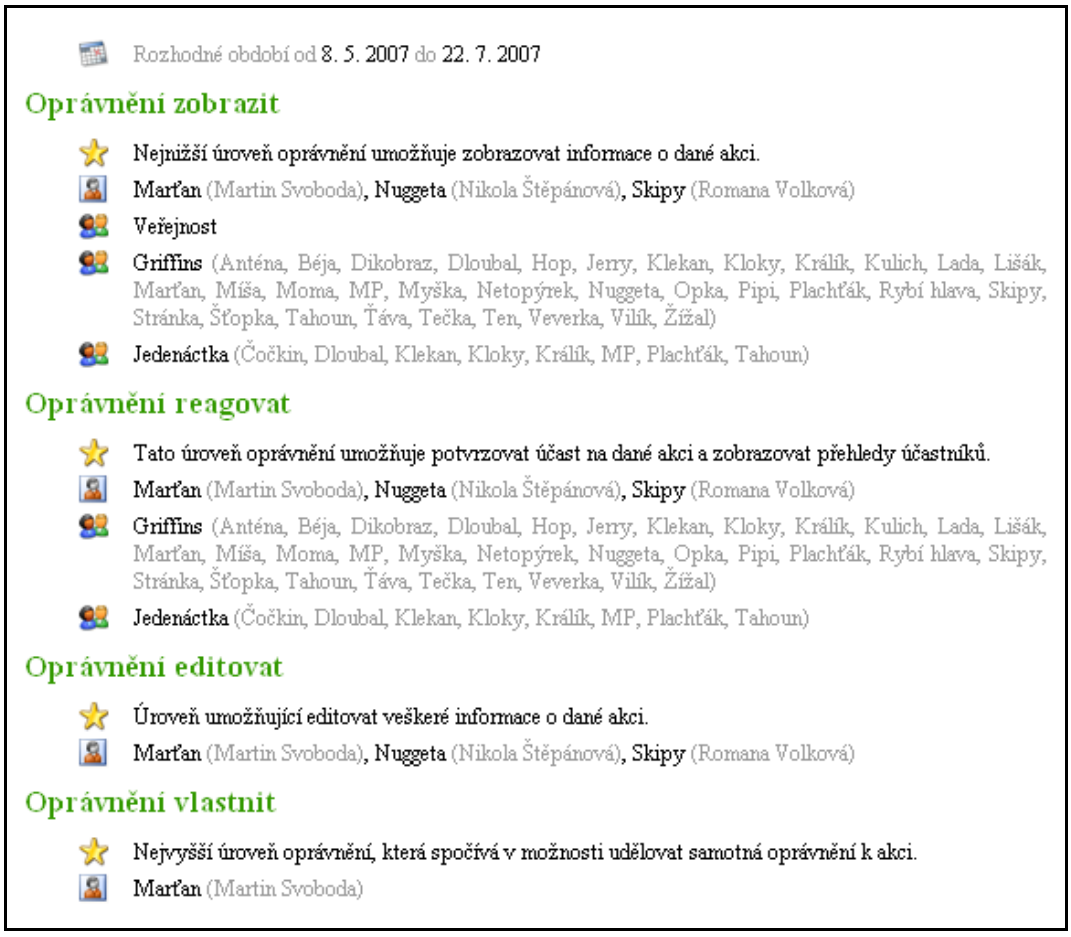
- Oprávnění zobrazit – jde o nejnižší úroveň, která umožňuje pouze zobrazit obsah daného objektu (plánované akce, diskuzní knihy apod.). V některých případech to může znamenat pouze částečné zobrazení. Obecně tato úroveň

dále stačí na to, aby uživatel mohl zobrazit přehled uživatelských oprávnění, pokud je ovšem sám registrován a přihlášen.

- Oprávnění reagovat – navíc umožňuje uživateli specifickým způsobem vyjadřovat své reakce. Konkrétní forma projevu je závislá na konkrétním objektu, ale lze si představit například právo hlasovat v anketní otázce, diskutovat v diskuzní knize, psát diskuzní komentáře k položkám nástěnky nebo potvrzovat účast na plánované akci.
- Oprávnění editovat – úroveň, která poskytuje oprávnění daný objekt editovat. Například v sekci diskuzí to znamená, že daní uživatelé mají oprávnění měnit vlastnosti dané knihy a spravovat všechny kapitoly umístěné v dané knize.
- Oprávnění vlastnit – jde o nejvyšší úroveň oprávnění, která dovoluje všem těmto uživatelům delegovat oprávnění na ostatní uživatele. Typicky by toto oprávnění měl mít jenom autor objektu a v odůvodněných případech i několik málo dalších uživatelů. Původnímu autorovi toto oprávnění nelze odebrat.

O hierarchii můžeme mluvit proto, že vyšší oprávnění automaticky implikuje nižší úrovně, ale ani toto chování není podmínkou, protože jak systém, tak používané databázové tabulky, tak i formulářové prvky, jsou schopny pracovat s obecnějším modelem. V tom případě se pouze korektně nadefinuje, která oprávnění implikují jiná. Podmínka inkluze není nutná.

Příklad přidělených oprávnění k plánované akci



Obr. 3.4.A: Výpis přidělených oprávnění k plánované akci

Struktura databázové tabulky členství ve skupinách (powers_tokens)

	Jméno atributu	Význam atributu a případné poznámky
0	`ti_token`	Identifikační číslo členství.
1	`ti_user`	Číslo uživatele.
2	`ti_group`	Číslo vybrané uživatelské skupiny.
3	`tt_from`	Datum začátku členství.
4	`tt_until`	Datum konce členství. Údaje je prázdný, pokud je členství stále platné.

Tab. 3.4.A: Struktura databázové tabulky členství uživatelů ve skupinách

Struktura databázové tabulky deskriptorů objektů (powers_objects)

	Jméno atributu	Význam atributu a případné poznámky
0	`oi_object`	Identifikační deskriptor oprávnění k objektu.
1	`ox_section`	Kód sekce.
2	`ox_object`	Kód typu objektu v rámci dané sekce.
3	`ox_item`	Identifikační číslo popisovaného objektu.
4	`ot_from`	Datum začátku referenčního období.
5	`ot_until`	Datum konce tohoto období.

Tab. 3.4.B: Struktura databázové tabulky deskriptorů objektů

Struktura databázové tabulky oprávnění uživatelů (powers_users)

	Jméno atributu	Význam atributu a případné poznámky
0	`px_object`	Deskriptor oprávnění.
1	`pi_user`	Číslo uživatele.
2	`ps_powers`	Celkové oprávnění.

Tab. 3.4.C: Struktura databázové tabulky přidělených oprávnění uživatelů

Struktura databázové tabulky oprávnění skupin (powers_groups)

	Jméno atributu	Význam atributu a případné poznámky
0	`px_object`	Deskriptor oprávnění.
1	`pi_group`	Číslo uživatelské skupiny.
2	`ps_powers`	Celkové oprávnění.

Tab. 3.4.D: Struktura databázové tabulky přidělených oprávnění uživatelských skupin

3.4.2 Model rolí a licencí

Druhým modelem, který je v systému použit na udělování oprávnění, je systém licencí. Tento model není určen jako konkurence předchozímu vyjmenovanému, ale používá se v jiných situacích a především pro potřeby systému. Kromě jiného je každá licence výlučně spojena s konkrétním uživatelem, tudíž s pojmem uživatelské skupiny se zde vůbec nepracuje.

Tato oprávnění si můžeme představit tak, že daný uživatel vystupuje pod určitou rolí nebo že má přidělenu určitou licenci pro výkon určité činnosti. V každém případě je platnost každé licence časově omezena. Platí od okamžiku jejího vydání – a buď neomezeně až do jejího odebrání, nebo do předem daného konce platnosti. Dále každá licence může být univerzální (v tom smyslu, že garantuje daný typ činnosti všeobecně), nebo je přímo spojená s konkrétním objektem v systému (v tomto případě je možné daný typ činnosti provádět pouze s daným objektem).

Praktickými příklady licencí a jejich smyslu je licence administrátora nebo licence pro správce uživatelské skupiny. Právě druhý zmíněný příklad je licence typu přímo spojeného s konkrétním objektem – zde uživatelskou skupinou. Uživatel, který je vlastníkem takové licence, může po dobu její platnosti rozhodovat o přijímání nebo vylučování členů v rámci dané uživatelské skupiny.

Struktura databázové tabulky licencí (powers_licences)

	Jméno atributu	Význam atributu a případné poznámky
0	`li_licence`	Identifikační číslo licence.
1	`lx_section`	Kód sekce.
2	`lx_role`	Kód typu licence v rámci dané sekce.
3	`lx_item`	Identifikačního číslo referenčního objektu. Tento údaj je prázdný u univerzálních licencí.
4	`li_user`	Číslo vlastníka.
5	`lt_from`	Datum začátku platnosti.
6	`lt_until`	Datum konce platnosti. Údaje může být prázdný u dosud platných licencí.

Tab. 3.4.E: Struktura databázové tabulky existujících licencí

3.5 Relace a přihlašování do systému

Pro práci v systému není nutné, aby uživatel byl přihlášen, nicméně v tomto případě bude mít značně omezené možnosti své práce. Předpokládá se proto, že registrovaný uživatel bude v systému pracovat přihlášený, a tudíž pod svým jménem se všemi důsledky, které z toho vyplývají.

V okamžiku, kdy konkrétní návštěvník poprvé vstoupí na stránky aplikace, je započata z pohledu systému jedna přístupová relace. Aby systém vůbec mohl rozumět pojmu relace, používá modelu Session, který je standardní součástí PHP. Nicméně s tím, že v jádře byly implementovány vlastní session handlery, které oproti běžnému řešení v PHP využívají pro ukládání dat databázi.

Zároveň s novou relací Session je vytvořen trvalý záznam o návštěvě také v samostatné databázové tabulce obsahující stručné informace o všech provedených přístupech na stránky systému. Mimo jiné se také při vytváření nové relace analyzují a zaznamenávají údaje jako přístupová IP a doménová adresa návštěvníka, jeho prohlížeč nebo preferovaný jazyk.

Jelikož se v průběhu jedné relace může do systému přihlásit více uživatelů, je nutné přihlášení uživatele chápat pouze jako stav aktuálně existující relace. Z pohledu systému je potřeba rozlišovat pouze 4 různé stavy relace:

- Anonymní nepřihlášený návštěvník, které zahájil novou relaci
- Přihlášený registrovaný uživatel s ověřeným stavem přihlášení
- Přihlášený registrovaný uživatel vyzvaný k ověření identity
- Anonymní návštěvník po odhlášení předchozího uživatele ze systému

Pojem relace proto nelze slučovat s přihlášenými uživateli. Jestliže v průběhu jedné relace dojde k přihlášení více uživatelů, bude každá tato návštěva reprezentována vlastním záznamem v tabulce návštěv a v okamžicích přihlášení resp. odhlášení

uživatele do resp. ze systému je nutné patřičným způsobem ošetřit, jaká data relace se mají uchovat a která je nutné odstranit.

3.5.1 Použité databázové tabulky

Pro ukládání dat relací je určena tabulka `logs_sessions`. Její důležitou součástí je také atribut `sc_matrix`, jehož funkce je detailněji vysvětlena v následující kapitole, v části týkající se novinek resp. panelů. Jde o bitovou mapu, jejíž jednotlivé pozice znázorňují aktualizaci obsahu příslušený sekcí, čímž umožňují takové chování, kdy systém velké množství dat relace nemusí znovu a znovu při každém zpracování požadavku klienta od začátku vyhodnocovat, ale může si řadu výsledků uložit do dat session – a právě pomocí tohoto atributu zjistí, jestli jsou všechna takto vygenerovaná data stále aktuální.

Jednotlivé záznamy o návštěvách jsou trvale ukládány do tabulky `logs_visits`. Za zmínku ještě stojí tabulka `logs_attempts`, ve které jsou logovány všechny pokusy o neúspěšná přihlášení do systému.

Čistě pro úplnost zmiňme ještě existenci tabulek `logs_domains`, `logs_agents` a `logs_screens`, které po řadě slouží na uchovávání záznamů o kombinacích přístupových IP a doménových adres, názvů klientských prohlížečů a v případě poslední tabulky rozlišení klientovy plochy.

Tabulka dat relací (logs_sessions)

	Jméno atributu	Význam atributu a případné poznámky
0	`si_session`	Identifikační číslo relace.
1	`si_visit`	Číslo odpovídající návštěvy.
2	`si_user`	Číslo přihlášeného uživatele. Hodnota je prázdná, pokud uživatel není přihlášen.
3	`so_active`	Stav relace.
4	`st_touched`	Čas posledního přístupu.
5	`sc_matrix`	Aktualizační bitová mapa.
6	`sn_data`	Vlastní data session.

Tab. 3.5.A: Struktura databázové tabulky přístupových relací

Tabulka záznamů o návštěvách (logs_visits)

	Jméno atributu	Význam atributu a případné poznámky
0	`vi_visit`	Identifikační číslo návštěvy.
1	`vs_type`	Typ přístupu.
2	`vi_user`	Číslo přihlášeného uživatele.
3	`vt_start`	Datum a čas začátku návštěvy.
4	`vt_end`	Datum a čas ukončení návštěvy.
5	`vc_length`	Efektivní délka návštěvy v sekundách.
6	`vi_provider`	Identifikace přístupové adresy.
7	`vi_from`	Identifikace předchozí adresy návštěvníka.
8	`vi_agent`	Identifikace klientova prohlížeče.
9	`vi_screen`	Klientovo rozlišení.
10	`vi_lang`	Preferovaný jazyk.

Tab. 3.5.B: Struktura databázové tabulky záznamů o návštěvách

Tabulka pokusů o narušení (logs_attempts)

	Jméno atributu	Význam atributu a případné poznámky
0	`ti_attempt`	Identifikační číslo záznamu.
1	`ts_type`	Typ přístupu.
2	`ti_visit`	Identifikační číslo návštěvy.
3	`ti_user`	Zjištěný uživatel.
4	`tt_stamp`	Datum a čas uskutečnění.

Tab. 3.5.C: Struktura databázové tabulky o neúspěšných pokusech přihlášení do systému

4 Základní prostředky jádra

4.1 Formulace MySQL dotazů

Nejenom z důvodů ladění při vývoji systému, ale především kvůli jednotnému a částečně automatizovanému vyhodnocování výsledků databázových dotazů je součástí jádra systému speciální třída, jejíž instance jsou používány pro pokládání dotazů na databázi MySQL.

První fází při pokládání každého dotazu je nejprve jeho samotná formulace. Ta je součástí konstrukce každého objektu rozšířeného dotazu zmíněné třídy. Kromě tohoto příkazu obsahuje každý rozšířený databázový dotaz identifikační údaje, které jsou složeny z kódu sekce a jednoznačného čísla v rámci dané sekce.

Kromě ladících účelů je význam těchto identifikátorů především v tom, že pokud je zapnuta příslušná monitorovací direktiva v konfiguračním souboru systému, každý provedený dotaz je zaznamenán do zvláštní databázové tabulky. Přesněji řečeno se nezaznamenává konkrétní formulace dotazu, ale identifikační údaje a údaj o délce vyhodnocování dotazu.

Díky tomu je možné získat v průběhu reálného nasazení systému velmi přesnou představu o tom, jak jsou jednotlivé části systému vytíženy, stejně jako zda některé části systému nejsou navrženy nevhodným způsobem.

Typy dotazů a jejich automatické zpracování

Posledním parametrem při konstrukci rozšířeného dotazu, ne však méně podstatným, je uvedení typu dotazu z databázového hlediska. Tento údaj by pochopitelně mohl být získán analýzou samotného dotazu, nicméně to by představovalo zcela zbytečnou zátěž v době běhu systému.

Podle uvedeného typu dotazu (tím může být například `MYSQL_SELECT`, `MYSQL_SELECT_CALC`, `MYSQL_INSERT`, `MYSQL_INSERT_UPDATE` a řada dalších) se po jeho vykonání automaticky do vnitřních struktur příslušného objektu rozšířeného dotazu uloží nejenom samotný výsledek dotazu, ale i údaje o nalezených či ovlivněných řádcích, informace o chybách, nebo například hodnota posledního vloženého klíče v případě dotazů ze skupiny INSERT.

Toto automatické zpracování přináší značné zjednodušení na straně používání výsledků položených dotazů, ať už jsou jakéhokoli typu. Zejména proto, že informace o chybě nebo hodnotě posledního vloženého klíče standardní databázové API poskytuje pouze a jenom k poslednímu provedenému dotazu. Pokud by se tedy tyto údaje nezpracovaly včas, mohly by být přepsány jinými dotazy. Díky tomu uživatel získává mnohem větší volnost v tom, kdy bude dotazy klást a kdy je bude zpracovávat.

Přístup k databázovému spojení

Na závěr ještě uveďme, že pro vyhodnocování samotných dotazů slouží spojení, které je obaleno další funkcionalitou a pro všechny sekce je přístupné v podobě globální proměnné. K vytvoření tohoto spojení jsou využity přihlašovací údaje, které jsou

uvedeny v konfiguračním souboru systému. V principu navíc nic nebrání tomu, aby byly od třídy rozšířeného spojení vytvořeny jiné instance, které reprezentují další potřebná databázová spojení.

Samotné spojení s databází je vytvořeno v průběhu inicializační části zpracování hlavního indexového souboru systému, stejně jako je ukončeno až na samotném konci. V souvislosti s tím je nutné uvést, že handlery session popsané na konci předchozí kapitoly samy využívají přístup k databázi, a proto musí relace session skončit dříve, než je spojení s databází ukončeno. Situaci navíc komplikuje využívání objektového modelu, a tudíž je nutné příslušné procedury volat ručně, aby bylo zaručeno jejich správné pořadí provedení.

4.2 Koncepce editačních procesů

Editačním procesem se rozumí jakýkoli modul, jehož účelem je vytvářet v systému nové aplikační objekty, editovat existující objekty, nebo obecně získávat od uživatele jakékoli vstupy a ty pak nějakým způsobem zpracovávat. Postavení takových modulů je poměrně odlišné vůči těm, které pouze vygenerují jednorázový výstup.

Z pohledu různých schémat vykonávání skriptů, které byly diskutovány v modelech návrhu systému, jsou v drtivé většině tyto moduly zpracovány v takovém režimu, že se na vrcholu zásobníku plánovače skriptů neuchovávají. To má praktický důsledek v jejich chování. Běžné editační procesy totiž jsou realizovány ve více krocích, a tudíž by nedávalo dobrý smysl využívat vlastnosti kotvy ve výchozím schematu vykonávání modulů. Uživatel toto chování pozná tak, že po dokončení celého editačního procesu se vrátí na takovou stránku, ze které proces zahájil.

4.2.1 Serializace a perzistence editačních procesů

Další charakteristickou vlastností editačních procesů, která již byla naznačena, je nutnost realizovat celý proces ve více krocích. V některých případech jde pouze o zpřehlednění procesu jeho rozdělením na více částí, někdy je ale takovýto model přímo vyžadován, protože podoba některého kroku je přímo závislá na kroku předchozím.

Z těchto důvodů je nutné zařídit, aby součástí vnitřních dat relace se staly i poměrně rozsáhlé informace o stavu těchto editačních procesů – a díky tomu mohl editační proces jakožto modul přežít i do dalšího kroku. Pro zajištění této funkcionality byl vytvořen poměrně silný model, který nejenom že splňuje potřebné vlastnosti, ale navíc umožňuje, aby uživatel v jednom okamžiku měl otevřeno libovolné množství editačních procesů libovolných typů.

Na implementační úrovni je takového chování dosaženo následující cestou. Předpokládá se, že jednotlivé moduly jsou představovány samostatnými třídami, v tomto případě odvozenými od společného předka editačních procesů, jejichž statická metoda `main()` je vykonávána plánovačem skriptů.

V průběhu této metody dojde k vytvoření instance dané třídy a ta jako zcela konkrétní objekt obsahuje všechny potřebné informace, které jsou nutné pro přežití celého editačního procesu do dalšího kroku. Vlastní logika rozhodnutí, jestli je potřeba navázat na existující proces, nebo začít nový, je začleněna právě do zmíněné metody `main()`, protože tyto podmínky se proces od procesu liší.

Rovněž uložení svých dat do relace si modul musí zajistit sám. K tomu je využit mechanismus standardní serializace objektu celého editačního procesu. Jen pro úplnost je potřeba zmínit, že některé procesy obsahují přechodné proměnné, které se nemohou nebo není vhodné je serializovat (v jiných jazycích obvykle uvozené modifikátorem `transient`). V tomto případě musí daný modul obsahovat magickou metodu `__sleep()`, která provede případné filtrování atributů.

4.2.2 Zamykání aplikačních objektů

Pokud je v systému vytvářen nový aplikační objekt (jako například plánovaná akce, diskuzní kniha apod.), neexistuje riziko, že by daným editačním procesem byli ovlivněni i případní další uživatelé, kteří jsou ve stejnou dobu v systému přihlášeni.

V případě editace existujících aplikačních objektů se však objevuje poměrně zásadní problém. Jeho příčina existuje proto, že editovat jeden konkrétní objekt obvykle může celá řada uživatelů. Nelze proto vyloučit, že se najdou alespoň dva uživatelé, kteří by přibližně ve stejnou dobu dostali nápad editovat tentýž objekt. Pokud by tato situace nebyla ošetřena, hrozilo by, že se systém nebude chovat podle představ ani jednoho z nich.

Je zřejmé, že tento problém zde existuje bez ohledu na možnost současného otevření více různých editačních procesů najednou. Pouze lze říci, že riziko výskytu nevyžádaných situacích se tím zvyšuje – zvláště v případě, kdy nějaký uživatel zapomene dokončit nějakou svou otevřenou editaci. Toto chování sice lze považovat za extrémní, nicméně má dobrý smysl ho řešit.

Požadovaného chování systému je dosaženo zamykáním daných aplikačních objektů. Už na samotném počátku je nutné říci, že toto řešení není z exaktního pohledu atomickou záležitostí, a tudíž není stoprocentně korektní. Riziko selhání je ale v tomto případě už tak extrémně nízké, že nemá smysl se jím zabývat. Navíc je otázkou, do jaké míry má systém na této úrovni vůbec možnost něco řešit atomicky. Jiná řešení využívající například uzamykání tabulek pomocí prostředků samotné databáze zase nepřicházejí v úvahu z jiných důvodů.

Z pohledu naplnění cílů projektu je proto řešení založené na zamykání aplikačních objektů zcela dostatečné. Uživatel, který chce zahájit editační proces nad daným aplikačním objektem, musí nejprve ověřit, jestli tento objekt není zamknut platným zámkem. Pokud je, editaci zahájit nemůže. V opačném případě získá vlastnictví potřebného zámku a po jistou dobu, která je nastavitelná v konfiguračním souboru, má zaručeno, že žádný jiný uživatel nebude moci zahájit editaci téhož objektu.

Časové omezení je zde kvůli již rozebírané možnosti, kdy uživatel zapomene dokončit otevřenou editaci. V tomto případě po vypršení daného limitu ztrácí onu garanci exkluzivity ve vlastnictví zámku. Pokud by se objevil jakýkoli jiný uživatel, který by chtěl také zahájit editaci téhož objektu, bude mu to povoleno a bude mu přidělen nový exkluzivní zámek. Původní uživatel už nebude mít oprávnění svou editaci dokončit – a to ani v případě, pokud by se o to pokusil až v okamžiku, kdy druhý uživatel už svůj proces řádně dokončil. Neúspěšný uživatel je o vzniklé situaci korektně informován, navíc jsou mu vypsány hodnoty formulářových polí a on si je může vlastními prostředky zaznamenat.

Pro úplnost dodejme, že pokud prvnímu uživateli vyprší platnost zámku, nemusí to nutně znamenat, že o svou rozpracovanou editaci automaticky přijde. K tomu dojde pouze tehdy, když se objeví jiný zájemce. Jestliže se neobjeví, může první uživatel bez problémů v editaci pokračovat a platnost jeho zámku je opět prodloužena.

Tabulka aplikačních zámků (system_locks)

	Jméno atributu	Význam atributu a případné poznámky
0	`ci_lock`	Identifikační číslo zámku.
1	`co_status`	Stav platnosti zámku.
2	`cx_section`	Kód sekce.
3	`cx_type`	Kód typu aplikačního objektu.
4	`cx_object`	Identifikace konkrétního objektu.
5	`ci_user`	Číslo vlastnícího uživatele.
6	`ct_start`	Datum a čas začátku platnosti.
7	`ct_touched`	Datum a čas poslední obnovy platnosti.

Tab. 4.2.A: Struktura databázové tabulky aplikačních zámků

4.2.3 Vnitřní mechanismy editačních procesů

Další zmíněnou vlastností obvyklých editačních procesů je jejich vícekrokový průběh. Typicky můžeme jmenovat například kroky výběru objektu, který má být editován, krok výběru složky pro umístění nově vytvářeného aplikačního objektu, krok vstupních formulářových polí nebo krok úspěšného ukončení celého editačního procesu.

Třída realizující předka editačních procesů nabízí základní funkce na vypisování chyb v přístupových oprávněních, neúspěšném pokusu o získání nového zámku, vypršení platnosti dříve získaného zámku, chybách v realizaci editačního procesu nebo naopak informaci o jeho úspěšném zakončení. Samotnou logiku využívání těchto funkcí a zejména pak vlastní logiku jednotlivých kroků procesu si vždy daný editační modul zařizuje zcela sám.

Typický modul proto začíná velmi komplikovanou kontrolou uživatelských oprávnění, existence požadovaných objektů, stejně jako získáváním potřebných zámků. Každý formulářový vstup od uživatele je vždy pečlivě kontrolován a pokud nevyhovuje, je uživatel opakovaně vyzván k jeho korektnímu zadání. Teprve v okamžiku, kdy je vše bez chyb připraveno, dojde k samotné realizaci procesu (rozumí se tím například vytvoření záznamu v databázi o právě vytvořené nové akci). Pokud během této realizace dojde k nějaké chybě, je o ní uživatel informován a v některých případech se může pokusit krok realizace znovu opakovat.

Důležité vzhledem k bezpečnosti je také to, že uživatel sám neurčuje ve svých požadavcích, jakým způsobem vypadá tok řízení editačního procesu. Zjednodušeně řečeno pouze říká, jaké vstupy právě vyplnil apod., ale vlastní logika vyhodnocení je na straně serveru, a tudíž není možné například kontroly obejít a požadovat rovnou vykonání neplatných dat.

4.3 Objekty editačních formulářů

Získávání a zpracovávání nejrůznějších vstupních údajů od uživatelů je bezpochyby základním kamenem pro naplnění cílů tohoto projektu. Bylo proto potřeba vymyslet takový model zpracovávání těchto údajů, který by vůči uživateli vystupoval dostatečně přátelsky a jednotně i přes rozmanitost vstupních prvků.

Na implementační úrovni je zřejmé, že větší univerzálnosti těchto vstupních prvků se dá dosáhnout jedině složitějším kódem, nicméně o to snazší používání těchto editačních objektů je pak na straně jednotlivých editačních procesů.

Z těchto důvodů byla přímo v jádře začleněna celá hierarchie takovýchto editačních objektů. Pokrývají nejrůznější druhy vstupních údajů, jejich specifické způsoby kontroly korektnosti včetně rozsáhlé možnosti parametrizace a v neposlední řadě také schopnost vygenerovat samotný HTML kód daného formulářového prvku.

Každý editační proces při vzniku své instance (jak bylo diskutováno v předchozí podkapitole) kromě jiných potřebných údajů rovněž ve své vnitřní struktuře vytvoří instance všech editačních objektů, které mají představovat očekávané vstupní údaje, a inicializuje je výchozími hodnotami.

Po odeslání vyplněného formuláře editační proces pouze zavolá příslušné metody editačních objektů a ty samy zajistí zpracování vstupní hodnoty od uživatele, zkontrolují ji a pomocí návratové hodnoty editačnímu procesu oznámí případnou nekorektnost údaje. Jestliže editační proces nezjistí žádné chyby, přejde ke kroku realizace.

4.3.1 Struktura editačních objektů

Ačkoli tyto objekty obecně zajišťují zcela rozmanité vstupy, jejich vnitřní mechanismy chování a například logika zpracovávání nekorektních stavů je stejná. To na straně editačního procesu představuje jejich elegantní používání. Až na výjimky se stejným způsobem konstruují, aktualizují, kontrolují i vypisují. Samozřejmě vždy s příslušnými parametry, které jsou pro každý typ objektu specifické.

Struktura typického editačního objektu předpokládá, že během jeho konstrukce je kromě jiných parametrů uvedena jeho výchozí hodnota. Ta může být prázdná (např. v případě vytváření nového aplikačního objektu) nebo nikoli (při editacích skutečných objektů). Důležité také je, že většina editačních objektů rozlišuje mezi výchozí hodnotou prvku a jeho aktuální platnou hodnotou.

To přináší možnost zajímavého chování, kdy po zjištění fatálních chyb v průběhu kontroly korektnosti nově zadaného údaje je možné jej nahradit jeho původní korektní hodnotou. Například když uživatel vymaže dříve vyplněný povinný údaj, a tudíž odešle prázdnou hodnotu, kontrolní mechanismy vygenerují příslušná hlášení a aktuální hodnota prvku je opět nahrazena jeho původní.

Velmi častou vlastností editačních objektů textového charakteru je také omezení maximální délky vstupního řetězce. S využitím java scriptové podpory na straně klienta je pak možné informovat ho o počtu znaků, které má ještě k dispozici. Jiným hojně využívaným parametrem je rozlišení mezi prvky, které musí být povinně vyplněny a těmi nepovinnými. V případě vícerozměrných editačních objektů pak tento parametr přechází v minimální a maximální počet vyplněných údajů z určitého nabízeného množství.

Na okraj poznamenejme, že každá instance editačního objektu je opatřena jednoznačným identifikátorem, který slouží nejenom k pojmenovávání elementárních prvků HTML formulářů a následnému přístupu k jejich hodnotám v PHP, ale také ke zmíněnému univerzálnímu způsobu oznamování chybových hlášení či k zajištění částečné interaktivity těchto formulářů.

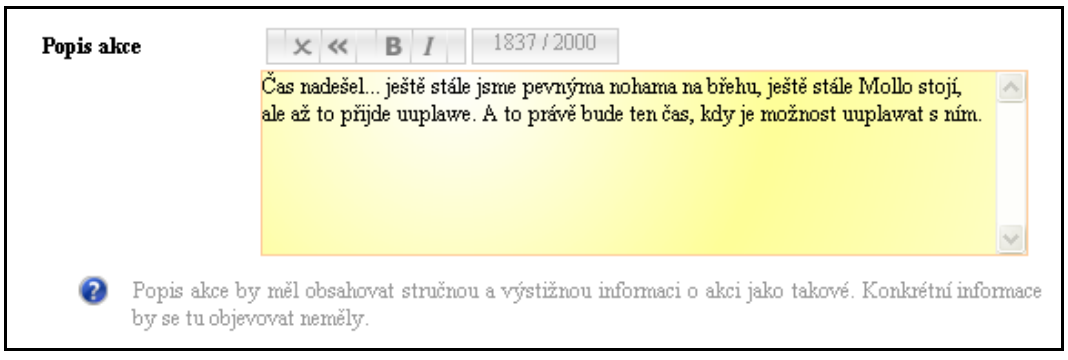
4.3.2 Přehled editačních objektů

U velkého množství typů editačních objektů bylo od počátku zřejmé, že budou pouze v malých variacích velmi často jednotlivými sekcemi využívány. Implementace hierarchie takovýchto objektů přímo v jádře proto měla smysl.

Jestliže daný editační proces potřebuje ke své činnosti nějaký editační objekt, který není součástí jádra, musí si ho nadefinovat sám, nebo může sáhnout k jinému nestandardnímu řešení. Konkrétním příkladem takovéhoto specifického řešení je pole odpovědí anketní otázky spolu s barvami jejich ukazatelů.

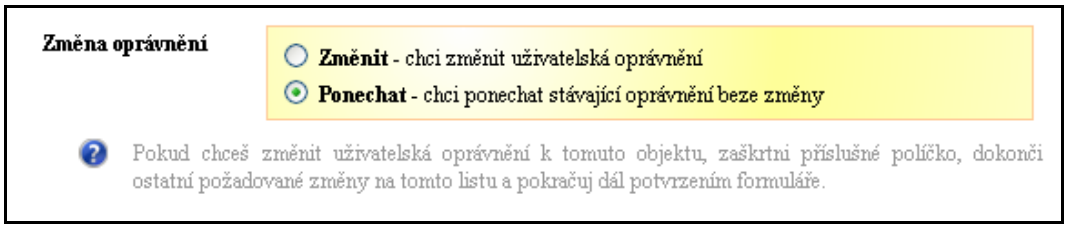
Následující přehled obsahuje všechny konkrétní typy editačních objektů, které nabízí jádro. Vynechány jsou pomocné abstraktní třídy, které vytváří společné rozhraní podobných typů objektů.

- *Textový řetězec* – očekává se krátký textový údaj, například název aplikačního objektu, emailová adresa atp. Údaje může být povinný nebo nikoli. Kontrola probíhá vůči předepsanému regulárnímu výrazu a rovněž se kontroluje maximální povolená délka řetězce.
- *Celé číslo* – editační objekt očekávající celé číslo v předem daném povoleném rozsahu. Číslo může nebo nemusí být povinně vyplněno.
- *Reálné číslo* – obdobný objekt jako předchozí celé číslo včetně kontroly povoleného rozsahu.
- *Datum ve formátu „j. n. Y“* – očekáváno je například datum „14. 3. 2005“. Kontroluje se celková platnost data. Objekt nezahrnuje kontroly data z minulosti apod.
- *Datum ve formátu „j. n.“* – datum bez roku. Kontrolována je smysluplnost zadaného data. Předpokládané využití je např. jako datum svátku při registraci uživatele.
- *Čas ve formátu „H:i:s“* – například „16:45:52“. Kontrolována je smysluplnost zadaného času, stejně jako je možné volit mezi povinným vyplněním nebo nikoli.
- *Přihlašovací jméno* – jde o jednoúčelový editační objekt login jména při registraci nového uživatele nebo následné editaci tohoto jména. Kromě nepovolených znaků kontroluje především unikátnost takovéhoto jména v rámci celého systému.
- *Rozsáhlý text* – očekává rozsáhlý textový údaj, který může a nemusí být formátovaný, stejně jako může nebo nemusí být povinně vyplněný. Kontrolováno je překročení maximální povolené délky, absence nepovolených znaků, použití nedovolených HTML tagů a jejich správné napárování.



Obr. 4.3.A: Formulářový prvek na vstup delšího textového údaje

- *Nabídka přepínačů* – jde o realizaci klasického „radio“ elementu HTML formuláře. Kontroluje tedy korektnost výběr právě jedné možnosti z nabízených.



Obr. 4.3.B: Formulářový prvek obsahující „radio“ elementy

- *Zaškrtávací prvek* – obdobně předchozímu objektu jde o jeden klasický „checkbox“ element.
- *Uživatelské heslo* – speciální editační objekt na uživatelské heslo. Obsahuje dvojici vstupních polí, z nichž obě musí být vyplněny stejnou hodnotou hesla. Kontrolována je dále minimální délka hesla.
- *Obrázek captcha* – jde o speciální generovaný obrázek obsahující různé znaky, jejichž správné interhpretování klientem a následné opsání do vstupního pole má zabránit automatizovanému vyplnění formuláře počítačem.

Obr. 4.3.C: Formulářový prvek na opsání kontrolního řetězce CAPTCHA

- *Pole textových údajů* – jde o vícerozměrný editační objekt, který obsahuje celé pole vstupních textových údajů. Objekt je schopen u každé jednotlivé položky kontrolovat platnost vůči regulárnímu výrazu a dále maximální povolenou délku. Celkově je navíc požadováno určité minimální množství správně vyplněných údajů z daného nabízeného množství. Zvláštní funkcí je možnost serializace všech vyplnených hodnot pomocí předem daného spojovacího řetězce. V tomto případě se nekontroluje maximální délka jednotlivých položek, ale délka takto vzniklého serializovaného pole.
- *Intervalové datum* – tento editační objekt představuje dvě vstupní data ve formátu „j. n. Y“, která navíc mají tvořit smysluplné období. Z tohoto důvodu je kontrolováno, jestli první datum je dřívější než druhé. Dále je možné zakázat, aby první nebo obě data mohla být v minulosti, a také je možné editaci toho či onoho data zakázat. Při korektním zadání období jsou automaticky spočítány například hodnoty krajních dat v jiných formátech nebo délka období ve dnech.
- *Pole checkboxů* – představuje výběr z velkého množství zaškrtávacích „checkbox“ elementů. Lze nastavit, aby bylo vybráno určité minimální množství položek. Typickým použitím tohoto objektu je například výběr klíčových slov u různých aplikačních objektů.

- *Výběr uživatelů* – jde o univerzální objekt na výběr libovolného většího množství registrovaných uživatelů. Konkrétní nabídka je automaticky upravena podle toho, jaké uživatele daný přihlášený uživatel vidí. Více informací o viditelných uživatelích je uvedeno v příslušné části kapitoly o návrhu uživatelských skupin.

4.3.3 Editační objekt uživatelských oprávnění

Uživatelská oprávnění představují v porovnání vůči ostatním dosud zmíněným editačním objektům velmi komplexní a složitý mechanismus. Nejenom kvůli složitosti samotného návrhu hierarchie oprávnění, ale také kvůli netriviálnímu začlenění do editačních procesů.

Editační objekt uživatelských oprávnění totiž nepředstavuje pouze složitější formulářový prvek, ale obsahuje také kompletní logiku na načtení existujících oprávnění, jejich následné kontroly vůči neopominutelným nebo naopak zakázaným výběrům a v neposlední řadě uložení upravených oprávnění do databáze.

Uživatel si v případě vytváření nového aplikačního objektu může vybrat, jestli si přeje zdědit oprávnění od vybrané složky, do které chce nový aplikační objekt umístit. Naopak při editaci existujícího objektu má uživatel možnost rozhodnout se, zda si v průběhu editačního procesu vůbec oprávnění přeje upravovat. Díky tomuto modelu je nutné mírně měnit chování na základě aktuálních požadavků klienta a typu editačního procesu. Formulářové prvky, které se dotazují na výše uvedené varianty, jsou rovněž přímou součástí tohoto editačního objektu.

Pokud uživatel vytváří nový aplikační objekt a nechce nebo nemůže dědit oprávnění od složky, inicializuje se vnitřní struktura prázdná. Pokud se oprávnění od složky dědit mají, je objekt inicializován oprávněními složky. Není však využito referenčního období dané složky, ale období nově vytvářeného objektu, které je už v tento okamžik známé. To může v případě oprávnění přidělených uživatelským skupinám způsobit, že dědění nebude ve smyslu uživatelů 1:1. Ačkoli je možné se tomuto chování vyhnout a vynutit si použití referenčního období složky, vybraný způsob chování je jako jediný smysluplný.

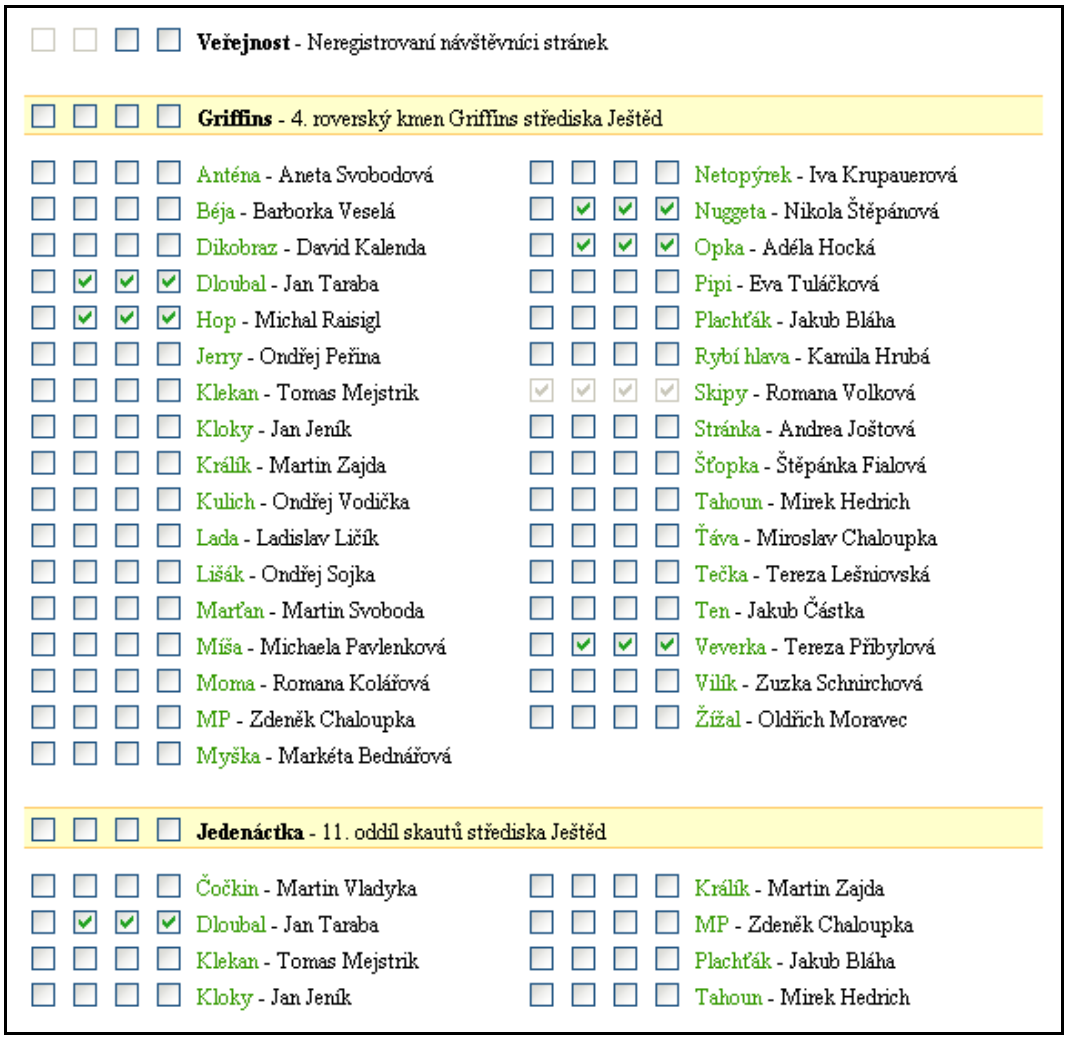
Dále je nutné si uvědomit, že stejně jako editovat daný aplikační objekt i delegovat oprávnění k danému objektu může v principu více uživatelů. Obecně tito uživatelé nemusí být v daném referenčním období členy týchž uživatelských skupin, a tudíž nemusí mít totožnou znalost o ostatních uživatelích systému. To jinými slovy znamená, že samotná nabídka uživatelů, která je součástí tohoto editačního objektu, může u každého takovéhoto správce vypadat odlišně.

Pokud k danému objektu už existují nějaká oprávnění u uživatelů nebo skupin, která jsou pro daného uživatele neviditelná, jsou tato automaticky ponechána a takovýto uživatel je nemá šanci ovlivnit. Pouze se dozví, že nějaká taková oprávnění existují. Pokud je potřeba, aby byla pozměněna, musí tak učinit ten, kdo je přidělil. Právě popsané řešení sice může vyústit v existenci nezrušitelných oprávnění, avšak vzhledem k nízké pravděpodobnosti výskytu této situace je zvolené řešení dostatečně dobré.

Při inicializaci editačního objektu oprávnění lez rovněž parametrizovat, jaké položky jsou implicitně vybrány a jaké naopak zakázány. Díky tomu lze například zařídit neopominutelnost oprávnění původního autora aplikačního objektu, nebo vyloučit přidělení některých úrovní oprávnění skupině veřejných návštěvníků.

Samotný formulář na přidělování oprávnění je členěn podle uživatelských skupin a v rámci každé skupiny jsou vypsáni všichni uživatelé, kteří jsou členy takové skupiny

(podle aktuálního referenčního období!). To typicky může znamenat, že někteří uživatelé budou na formuláři vypsání vícekrát. Pro snazší práci s takovýmto formulářem proto bylo nutné dodělat java scriptovou podporu, která navíc automaticky kontroluje závislosti jednotlivých úrovní oprávnění.



Obr. 4.3.D: Formulářový prvek na delegování uživatelských oprávnění

4.3.4 Editační objekt na upload souborů

Druhým editačním objektem, který obsahuje komplexní vnitřní mechanismy chování a vymyká se tak obyčejným editačním objektům, je pole formulářových prvků na uploadování souborů skrze standardní formulář.

Důležitou vlastností tohoto formulářového prvku je to, že není určen pouze na upload jednoho souboru, ale obecně většího množství souborů. Při inicializaci se určí, kolik polí bude daný vstupní formulář obsahovat, a dále, jaké je minimální požadované množství souborů, které musí být korektně nahrány.

Kdykoli je zavolána standardní kontrolní metoda tohoto objektu, dojde k vyhledání všech úspěšně uploadovaných souborů a ty se nechají projít důkladnými kontrolami. Pokud jsou úspěšné, uchová se jak samotný soubor tak i formace o něm. V opačném případě je uživatel standardními chybovými hlášeními upozorněn na konkrétní chyby a zároveň je mu vypsán detailní seznam, který obsahuje informace o všech odmítnutých souborech.

Kontrola nahraných souborů

U každého nahraného souboru jsou nejprve analyzovány chybové příznaky, které automaticky zpracovává PHP. Díky tomu je možné vyloučit chybně nahrané soubory nebo zjistit odmítnutí souboru kvůli překročení maximální povolené velikosti. Dále je soubor kontrolován z hlediska jeho MIME typu. Editační proces tak v parametrech může přesně vyjmenovat, jaké typy souborů jsou pro něj akceptovatelné. Jestliže soubor není požadovaného typu, je odmítnut.

Dále je kontrolována velikost souboru podle požadavků editačního procesu. Tato kontrola může buď zpracovávat každý soubor odděleně a zkoumat překročení jeho velikosti, nebo může porovnávat celkový součet velikostí všech nahraných souborů. Výběr tohoto chování je opět na editačním procesu.

Velmi silným mechanismem specifické kontroly uploadovaného souboru je možnost zavolání speciální funkce, která sama je parametrem celého editačního objektu. Každá takováto funkce musí jako svůj první parametr povinně akceptovat úplné jméno zkoumaného souboru. Jsou-li vyžadovány další parametry, pak jsou rovněž předány kontrolní metodě editačního objektu a ta už jejich předání dál zařídí sama. Příkladem konkrétní implementace takovéto specifické kontrolní funkce je kontrola rozměrů při nahrávání náhledového obrázku fotografického alba.

Pokud jakákoli část kontrol selže, je soubor odmítnut a uživatel získá o důvodech odmítnutí poměrně podrobná chybová hlášení. V opačném případě je soubor přesunut z dočasného úložiště, které má k dispozici PHP, do jiného dočasného úložiště, které si určí sám editační proces. Tento postup je nutný vzhledem k tomu, že editační procesy jsou obecně vícekrokové, stejně jako jeden konkrétní objekt na nahrávání souborů může být využíván opakovaně.

Konečné zpracování nebo umístění souboru na definitivní místo je čistě v kompetenci editačního procesu. Pro tyto účely má daný editační proces částečný přístup do vnitřní struktury tohoto editačního objektu – tak, aby měl k dispozici podrobný seznam všech úspěšně nahraných souborů.

Zpracovávání ZIP archivů

Poslední důležitou schopností, kterou nelze opomenout, je možnost zpracovávat obsah nahraných archivů formátu ZIP. Pokud je toto chování editačním procesem vyžadováno, kontrolní metoda automaticky nahrané archivy rozbalí a jednotlivé soubory extrahuje. Ostatními kontrolními mechanismy pak už každý takovýto soubor prochází odděleně a informace o původním umístění v archivu už není podstatná.

Analýza MIME typu souborů

Jak již bylo řečeno, jedním z kroků kontroly korektnosti uploadovaných souborů je i kontrola požadovaného typu souboru. Každý editační proces určuje, jaké typy souborů akceptuje. Rovněž nemusí omezovat nijak.

Standardně je informace o typu nahraného souboru k dispozici, aniž by musela být analyzována. To však nemusí být pravidlem a u souborů, které byly extrahovány z nahraných archivů, tento typ není znám určitě.

Z tohoto důvodu se využívá funkcí z rozšíření PHP, které jsou schopny na základě své vnitřní „magic.mime“ databáze určit množinu nejpravděpodobnějších typů zkoumaného souboru. Tato množina typicky obsahuje více než jeden typ, proto následuje heuristický odhad. Ten jednoduše nepředpokládá problémy, a tudíž z množiny nalezených typů preferuje ty, které jsou požadovány.

4.4 Jiné podpůrné funkce jádra

Jádro systému obsahuje i celou řadu méně významných mechanismů nebo funkcí, které jsou určeny na podporu rutinní činnosti v jednotlivých sekcích. Jmenovat lze například funkce na konverze z/do nejrůznějších formátů dat a časů.

Některé jiné mechanismy, jako například podpora více jazykových mutací nebo určování a vyhledávání novinek, jsou sice zásadního návrhového významu v rámci celého systému, mají dopad na celou řadu implementačních detailů v rámci všech sekcí a jsou velmi často využívány, ale neskrývají v sobě žádnou vnitřní složitost.

Proto je pojednání o nich pouze informativního charakteru.

4.4.1 Informační a chybové zprávy

Mechanismu informačních zpráv a chybových hlášení je určen pro jednotný způsob předávání informací takovéhoto charakteru směrem k uživateli. Tyto zprávy nejčastěji vznikají při úvodních kontrolách uživatelských oprávnění při ověřování možnosti zpracovat daný modul, ale zejména pak při kontrolách uživatelských vstupů v editačních procesech.

Zásadní vlastností návrhu tohoto mechanismu je skutečnost, že tyto zprávy vznikají v okamžiku, kdy ještě nemohou být přímo vypisovány. V případě editačních procesů pak jednotlivá hlášení obvykle přísluší zcela konkrétním editačním objektům, u kterých došlo k příčině vygenerování daného chybového hlášení. Z tohoto důvodu je nutné dokázat jednotlivé skupiny existujících informačních zpráv od sebe pomocí vhodné identifikace oddělit. K tomuto účelu slouží stejný způsob identifikování, jaký používají samotné editační objekty.

Každá informační zpráva je dále opatřena typem, který udává, jaká informační ikonka bude při vypisování použita. Uživatel pak snadno rozezná fatální chyby či varování na straně jedné a naopak položky nápovědy či tipy na straně druhé.

Kromě toho jsou v rámci každé skupiny jednotlivé záznamy řazeny pomocí číselného parametru, který je s nimi také spojen. Díky tomu lze při jejich vypisování dosáhnout takového pořadí, jaké je obvyklé nebo požadované. Typicky se nejprve vypisují chybová hlášení, aby je uživatel považoval za důležitější.

4.4.2 Podpora jazykových mutací

Celý systém už od samotného návrhu počítá s tím, aby mohl být nasazen s podporou více jazykových mutací. Přesněji řečeno, aby konkrétní lokalizace proběhla až za běhu na základě analýzy klientem preferovaného jazyka.

Každá sekce proto obsahuje zvláštní hlavičkové soubory, které obsahují definice všech textů v sekci použitých. Konvence na pojmenovávání těchto souborů je `_text_$lang.php`, kde `$lang` je 2-písmenný kód jazyka dané lokalizace.

Systém pro každou relaci vyhodnotí preferovaný jazyk a pokud jsou k dispozici definice textů daného jazyka, používá je. V principu není nutné, aby všechny sekce systému byly lokalizovány daným jazykem. Pokud nějaká sekce daný jazyk nepodporuje (ačkoli ostatní ano), pak je u této sekce vybrán její výchozí jazyk.

4.4.3 Evidence systémových událostí

Databázové tabulky určené na ukládání typických aplikačních objektů podporují pouze uchování informace o autorovi, časového razítka okamžiku vytvoření daného objektu a obvykle také informaci o poslední provedené změně (uživatel i čas provedení).

Tyto informace pochopitelně nemohou poskytnout kompletní představu o celé historii daného objektu. Přitom v některých situacích by takováto evidence mohla přinést důležité poznatky. Ponechat řešení na jednotlivých sekcích by bylo zbytečně nekoncepční, a proto byl vytvořen společný model, který je všem sekcím libovolně k použití.

Kdykoli nějaká sekce potřebuje zaznamenat informaci o provedení nějaké události s vybraným objektem, může k tomu využít právě funkcí, které nabízí přímo jádro. Konkrétně je k ukládání těchto záznamů používána tabulka `system_events` a sekce systému rovněž nabízí univerzální funkci na vypisování historie příslušných objektů.

Tabulka historie událostí (system_events)

	Jméno atributu	Význam atributu a případné poznámky
0	`ei_event`	Identifikační číslo události.
1	`ex_section`	Kód sekce.
2	`ex_type`	Kód typu objektu.
3	`ex_object`	Identifikace konkrétního objektu.
4	`es_code`	Kód typu události.
5	`ei_user`	Původce události.
6	`et_stamp`	Datum a čas provedení události.
7	`en_value`	Položka volného použití.

Tab. 4.4.A: Struktura databázové tabulky systémových událostí

4.4.4 Monitorování činnosti systému

Vzhledem k většímu rozsahu celého systému byly do jádra přidány i funkce, které jsou schopny určitým základním způsobem monitorovat činnost systému jako takového nebo činnost přihlášených uživatelů v něm.

První z nabízených mechanismů spočívá v možnosti podrobné evidence všech provedených databázových dotazů. V tomto případě se nepředpokládá, že by tento mechanismus byl za ostrého běhu systému stále aktivní. Je určen pouze na vyhodnocení zátěže jednotlivých částí systému či sekcí a díky tomu je možné nalézt chyby v návrhu databázových tabulek, které se mohou projevovat například jen při určitém charakteristickém způsobu chování konkrétních uživatelů.

Druhým nabízeným mechanismem je možnost evidence, jaké konkrétní moduly přihlášení uživatelé používají. Oproti předchozímu uvedenému mechanismu není tento nijak časově náročný, a tudíž může být stále aktivní. Navíc je implementován hned ve třech různých úrovní hloubky monitorování.

Nejjednodušší úroveň pouze zkoumá, kolik modulů během návštěvy daného uživatele bylo realizováno a kolik efektivního času v nich uživatel strávil. Rozlišujícím údajem na této úrovni je tedy pouze pojem návštěvy. Druhá úroveň rozlišuje údaje navíc podle jednotlivých sekcí a třetí nejvyšší úroveň poskytuje nejjemnější rozlišení až do hloubky každého jednotlivého modulu.

Pojem efektivního času je zaveden zejména proto, že uživatel může zahájit svou relaci, pak být několik hodin neaktivní a následně svou činnost opět obnovit. Z tohoto důvodu je na implementační úrovni omezení maximální velikosti časových rozestupů mezi požadavky.

Aktivace všech monitorovacích mechanismů obecně podléhá nastavení v konfiguračním souboru systému.

Tabulka monitorování databázových dotazů (watches_queries)

	Jméno atributu	Význam atributu a případné poznámky
0	`qx_section`	Kód sekce.
1	`qx_query`	Kód databázového dotazu.
2	`qt_time`	Datum a čas provedení.
3	`qc_length`	Délka provádění v mikrosekundách.

Tab. 4.4.B: Struktura databázové tabulky na monitorování provedených databázových dotazů

Tabulka monitorování činnosti uživatelů (watches_modules)

	Jméno atributu	Význam atributu a případné poznámky
0	`px_visit`	Identifikační číslo návštěvy.
1	`px_section`	Kód sekce.
2	`px_module`	Kód modulu v rámci sekce.
3	`pc_count`	Počet načtení modulu.
4	`pc_length`	Efektivní čas uživatele v sekundách.

Tab. 4.4.C: Struktura databázové tabulky na monitorování činnosti uživatelů

4.4.5 Mechanismy na určování novinek

Jedním ze stěžejních prostředků, jak dosáhnout kvalitní práce uživatele v systému, je umožnit mu velmi snadným způsobem vyhledávat novinky a změny, které se staly od okamžiku jeho poslední návštěvy. Pokud tyto mechanismy jsou dostatečně kvalitní, uživatel má velmi dobrý přehled o tom, co se v systému obecně děje.

Jádro systému proto poskytuje informace o období, ve kterém mají jednotlivé sekce novinky vyhledávat. Kromě toho, standardní plocha aplikace obsahuje panely, které tyto informace koncentrují pro celý systém na jedno místo.

Z praktického hlediska se navíc ukázalo, že je vhodné, aby sám uživatel měl možnost rozsah tohoto období upravit.

4.5 Kontextová menu a panely

Standardní pracovní plocha, ve které má klient možnost prostřednictvím svého webového prohlížeče se systémem pracovat, obsahuje kromě titulního menu a prostoru hlavního těla stránky také celou řadu panelů nejrůznějšího charakteru a obsahu.

Všechny tyto panely jsou určeny k tomu, aby výrazným způsobem usnadnily činnost uživatele – a to tak, že mu dají k dispozici informace či odkazy, které by ho mohly zajímat. V krajním případě jde o hlavní menu sekcí nebo kontextová menu aplikačních objektů, jiné panely informují o aktuálních informacích a jiné zase obsahují seznamy vyhledaných novinek.

Z hlediska zdrojového kódu je každý panel realizován jako konkrétní instance třídy, která implementuje magickou metodu `__toString()`. Ta je také automaticky volána v okamžiku, kdy je příslušný panel vypisován. Po vytvoření instance nového panelu dojde k registraci ke správci všech panelů. Při této registraci se určí, na jaké pozici má být panel umístěn.

Všechny panely jsou vypisovány až v okamžiku, kdy už zcela skončilo zpracovávání naplánovaných modulů. Pokud v nich došlo k nějakým důležitým změnám (viz níže), panely tyto změny mohou zaregistrovat. Každý panel může navíc implementovat svou inicializační metodu (nikoli konstruktor), která je automaticky zavolána správcem ještě dříve, než dojde ke zmíněnému vypisování panelů. Díky tomu se mohou jednotlivé panely ovlivňovat i navzájem.

4.5.1 Aktualizační matice a uchovávání dat panelů

Některé z těchto panelů jsou velmi náročné na vygenerování jejich obsahu. Skupina panelů, kterých se tento problém týká, může vygenerovat desítky databázových dotazů, jejichž vyhodnocení sice využívá existujících databázových indexů, ale jejich opakované vyhodnocování by bylo značně na úkor efektivnosti běhu celého systému a zátěže serveru.

Proto bylo nutné vyvinout mechanismus, který umožní jednou vygenerované výsledky těchto dotazů uložit do dat relace. Problém ovšem spočívá v tom, že s činností uživatelů může obecně docházet k tomu, že některá takto uložená data by už nebyla aktuální.

Aby bylo implementované řešení korektní za všech okolností, muselo by využívat složitějších operací, které by ale musely být provedeny atomicky. Na této úrovni něčeho takového není možné dosáhnout, a proto vybrané řešení může někdy selhat. Pravděpodobnost tohoto selhání je ale značně nízká a je v souladu s vymezeným účelem celého projektu.

Součástí přístupové relace je tzv. aktualizační matice. Jde o bitovou mapu, kde každá bitová pozice je vyhrazena jednotlivým sekcím systému. Není však uložena přímo ve vnitřních datech relace, ale musí být realizována samostatným atributem v databázové tabulce `logs_sessions`. To je nutné proto, aby údaj byl přístupný všem aktuálním relacím, jak bude hned vysvětleno.

V okamžiku, kdy se načítají data samotné session (v příslušném handleru), se rovněž načte aktuální podoba zmíněné aktualizační matice. Systém si tuto hodnotu ponechá a okamžitě svůj údaj v databázové tabulce vynuluje. Z uložené hodnoty matice je pak možné interpretovat, ve kterých sekcích došlo ke změně, a tudíž je nutné selektivně aktualizovat veškerá data, která se těchto sekcí týkají.

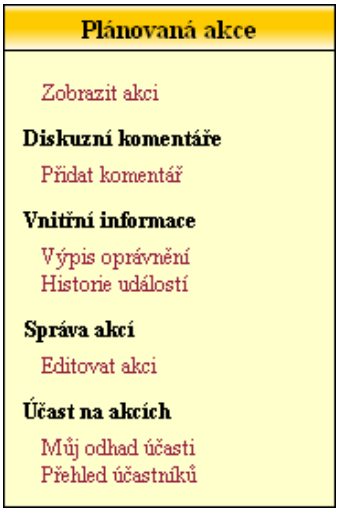
Pokud uživatel sám způsobí změnu obsahu nějaké sekce, která může ovlivnit aktuálnost takto generovaných dat, zneplatní si příslušnou pozici ve své aktualizační matici (veškerá činnost modulů je skončena dříve, než se kontextové panely vypisují – z tohoto důvodu tato změna matice je provedena včas, aby měla smysl a byla zaregistrována) a především obdobným způsobem upraví aktualizační matice všech ostatních aktuálně platných relací ostatních návštěvníků.

4.5.2 Přehled základních panelů

Následující přehled obsahuje seznam a popis většiny základních panelů, které jsou v jádře systému implementovány. Ty méně významné byly vynechány.

Hlavní a kontextová menu

Panel obsahující položky menu. Buď může jít o menu celé sekce, nebo o kontextové menu určitého aplikačního objektu. Jmenovat můžeme například kontextová menu diskuzní knihy a diskuzní kapitoly. Kontextová menu obecně obsahují odkazy, které nějakým způsobem zobrazují nebo manipulují s daným aplikačním objektem.



Obr. 4.5.A: Kontextový panel lokálního menu akce

Jistou specialitou, kterou provádějí hlavní menu sekcí, je generování odkazů na roční výpisy příslušných aplikačních objektů. Obecně ale hrozí, že takovéto seznamy by po několika letech funkčnosti systému byly příliš dlouhé. Proto byl vytvořen mechanismus, který vypisuje pouze období blízké současnému roku s větším záběrem do minulosti a dále menší okolí roku, který byl naposledy zobrazen. Díky tomuto dynamickému seznamu je možné komprimovat celkovou velikost menu, ale není tím zabráněna možnost dohledání jakéhokoli požadovaného roku.

Panel aktuálního kalendáře

Jde o informační panel, který zobrazuje standardní kalendářovou tabulku – jak pro celý aktuální měsíc, tak i část předcházejícího a následujícího měsíce. Přesah do vedlejších měsíců poskytuje větší přehled a tím větší orientaci uživatele.

Obr. 4.5.B: Kontextový panel aktuálního kalendáře

Smyslem panelu samozřejmě není pouze zobrazení kalendáře, ale především jeho naplnění daty. Řada aplikačních objektů na stránkách má vymezeno období, které je pro ně určitým způsobem určující. U plánovaných akcí jde o celé období konání, u položky nástěnky datum vyvěšení a například u anketní otázky jde o datum konce hlasování.

Pokud na dané datum připadá ve výše uvedeném smyslu nějaký aplikační objekt, je místo pouhého čísla dne zobrazen odkaz. Uživatelům, jejichž prohlížeč navíc poskytuje dostatečnou java scriptovou podporu, je interaktivně po najetí myši na konkrétní datum zobrazen přímý výpis takto nalezených objektů.

Panel kalendáře je jedním ze skupiny panelů, která využívá dříve popsanou aktualizační matici.

Panel novinek a změn

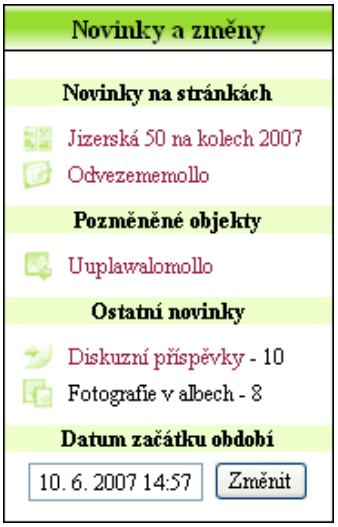
Tento panel obsahuje celkem tři informační části. První obsahuje konkrétní seznam novinek aplikačních objektů, které jsou vzhledem k nastavenému období považovány za novinky. Druhá část obsahuje stejně vypadající seznam položek, které

byly změněny ve stejném období (kvůli duplikací jsou zde vynechány všechny objekty, které jsou už obsaženy v předchozí části).

Namátkou můžeme mluvit o aplikačních objektech plánovaných akcí, diskuzních knihách, fotografických albech nebo položkách nástěnky.

Třetí část obsahuje také přehled novinek, ale už ne konkrétních aplikačních objektů, ale pouze souhrnné vyjádření počtu takových položek. Tím se rozumí například informace o počtu nových příspěvků v diskuzních knihách, počtu nově vložených fotografií do fotografických alb či počtu nově zaregistrovaných uživatelů v systému.

Tyto souhrnné novinky mohou obsahovat konkrétní odkaz, pokud v příslušné sekci existuje modul, který tyto novinky dokáže detailněji vypsat. Pokud takový modul neexistuje (v některých případech by to ani nedávalo dobrý smysl), příslušná položka nemá charakter odkazu a má pouze informační význam.



Obr. 4.5.C: Kontextový panel nalezených novinek a změn

Panel aktuálních témat

Panel aktuálních témat obsahuje seznam odkazů na konkrétní aplikační objekty, které jsou v blízkém období k aktuálnímu dni v určitém smyslu aktuální. Rozsah tohoto období je předmětem nastavení v konfiguračním souboru systému a jeho výchozí hodnota je plus/mínus 2 dny. Význam aktuálnosti je pak odlišný u každého zkoumaného objektu.

Seznam bude obsahovat například všechny akce, které v tuto dobu probíhají, anketní otázky, kterým končí období hlasování, nebo také uživatelé, kteří v tomto období slaví narozeniny nebo svátek.

Navigace stránkovaných výpisů

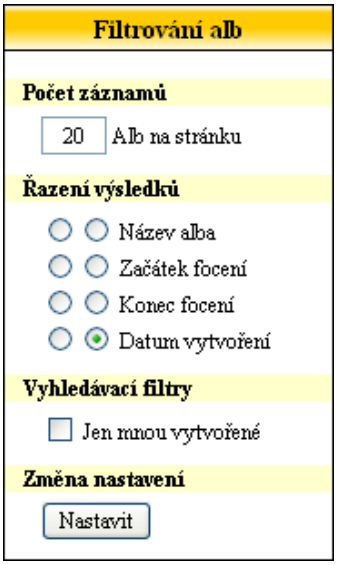
Jak bude detailněji rozebráno v následující podkapitole, jádro systému obsahuje univerzální podporu pro stránkované výpisy nejrůznějších položek. Tento panel slouží pro zjednodušení navigace mez jednotlivými stranami takovýchto výpisů.

Vlastní navigace spočívá v možnosti přímého přechodu na předchozí stránku či následující stránku výpisu, dále první a poslední – a především na libovolnou požadovanou, určenou zadaným pořadovým číslem.

Nastavení vyhledávacích kritérií

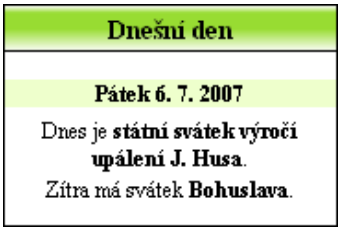
V případě panelu vyhledávacích kritérií nejde o jeden konkrétní panel, ale celou řadu odvozených implementací. Většinu stránkovaných výpisů je možné nejrůznějšími způsoby parametrizovat a omezovat tak výsledky hledání. Právě tento panel je určen jako rozhraní, kde uživatel tato nastavení provádí.

Ačkoli každý stránkovaný výpis může implementovat libovolně vypadající panel kritérií, obvykle tyto panely obsahují volby na změnu vzhledu vypisovaného seznamu z nabízené nabídky, pole udávající



Obr. 4.5.D: Kontextový panel vyhledávacích kritérií

počet vypisovaných záznamů na jednu stranu, volby na různé způsoby řazení nalezených výsledků a dále volby na různá filtrování těchto seznamů.



Obr. 4.5.E: Kontextový panel informací o aktuálním dni

Panel aktuálních informací

Jde o velmi jednoduchý panel, který zobrazuje informace o aktuálním dni a o tom, kdo v daný a následující den slaví podle kalendáře svátek, resp. jestli v tyto dny nejsou nějaké státem uznávané svátky.

4.6 Stránkované a parametrizované seznamy

Generování seznamů nejrůznějších položek je v celém systému velmi častou záležitostí. Z tohoto důvodu má smysl poskytnout univerzální podporu nejrůznějším stránkovaným výpisům přímo v jádře systému.

Implementovány jsou celkem dvě abstraktní třídy, které tuto funkcionalitu nabízí. Druhá z nich je odvozena od té první a nabízí řadu rozšířených možností – zejména možnost stránkování velkých seznamů, možnost zobrazení informačních hlaviček vyhledávání a dále automatizované zpracovávání vyhledávacích kritérií.

Samotné funkce na zpracovávání těchto kritérií jsou umístěny v prvním jednodušším předkovi. Rozdíl je v tom, že třída prvního jednoduchého seznamu tyto metody jenom implementuje, zatímco druhá třída rozšířeného seznamu je sama v průběhu inicializace volá.

4.6.1 Jednoduchý vs. rozšířený seznam

Konkrétní implementace seznamů položek daného typu, jsou odvozeny právě od jedné z těchto základních abstraktních tříd. Často se ale stává, že jeden konkrétní typ položky má více konkrétních implementací. Například v sekci diskuzních knih existuje hned několik konkrétních implementací seznamů diskuzních kapitol.

Tyto třídy vykazují natolik odlišené chování, že se vyplatí realizovat je v oddělených třídách, zároveň ale mají hodně společného. Konkrétně metody, které generují části databázových dotazů, které jsou závislé na aktuálním nastavení vyhledávacích kritérií, by ve všech těchto třídách byly totožné.

Z tohoto důvodu je oběma těmto konkrétním třídám nadřazena ještě jedna společná abstraktní třída, která implementuje právě zmíněné společné metody, které se týkají jakýchkoli seznamů diskuzních kapitol. Problém je ovšem v tom, že rozšířený seznam má zcela odlišný způsob konstrukce, inicializace i chování než jednoduchý seznam.

V rozebíraném případě jedna konkrétní implementace seznamu kapitol požaduje chování jednoduchého seznamu, druhá rozšířeného. Z tohoto důvodu musí společný předek obecného seznamu kapitol být odvozen od rozšířeného seznamu a dále musí existovat mechanismus, jak mohou být jeho rozšířené vlastnosti odmítnuty na úkor využití chování jednoduchého seznamu.

4.6.2 Funkce jednoduchého seznamu

Jak již bylo řečeno, předek jednoduchých seznamů implementuje pouze metody, pomocí nichž se dají zpracovávat změny v aktuálním nastavení vyhledávacích kritérií, a dále metodu, která je schopna automaticky zaregistrovat příslušný panel vyhledávacích kritérií.

Veškeré ostatní chování si každá konkrétní implementace jednoduchého seznamu musí zařídit sama. Důvodem existence této třídy je tedy to, aby funkcionalita nastavení vyhledávacích kritérií mohla být přístupná i seznamům, které nestojí o chování, které poskytují rozšířené seznamy rozebírané dále.

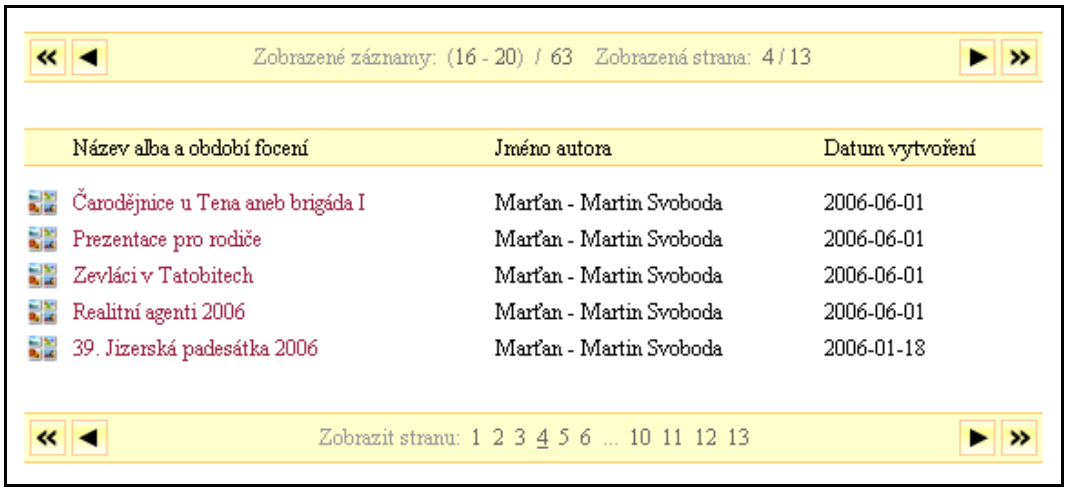
4.6.3 Funkce rozšířeného seznamu

Hlavním smyslem rozšířených seznamů je jejich dělení na stránky o předem dané velikosti. Komplikovanost konstrukce takovýchto seznamů spočívá v tom, že dokáží pracovat se dvěma druhy databázových dotazů, jejichž položky výsledku mají danou instanci seznamu tvořit.

První z těchto typů je klasický dotaz obsahující konstrukci `SELECT … FROM …` Takovýto dotaz najde úplně všechny vyhovující výsledky. Samotný proces stránkování je v tomto případě realizován až po vyhodnocení databázového dotazu a může tudíž konstrukce příslušného objektu seznamu může být provedena v jednom kroku.

Problém přichází s druhým typem dotazu, který umožňuje vybrat pouze předem známou část ze všech odpovídajících výsledků, ale zároveň vrátit i informaci o tom, kolik by těchto záznamů celkově bylo, pokud by dotaz nebyl omezen formulí `LIMIT`. Konkrétně jde o dotaz tvaru `SELECT SQL_CALC_FOUND_ROWS … FROM … LIMIT …`

V tomto případě je část konstrukce seznamu nutné provést před provedením databázového dotazu a naopak zbylou část až poté. Díky tomu je konstrukce rozšířených seznamů složitější a pokud nějaký konkrétní seznam je odvozen od rozšířeného, ačkoli nechce využívat jeho služeb, musí tuto inicializaci obejít.



Obr. 4.6.A: Příklad rozšířeného seznamu spolu s informačními hlavičkami

Vyhledávací hlavičky

Abstraktní předek rozšířených seznamů dále nabízí funkce, které dokáží vypsat univerzální informační hlavičky, které buď obsahují stručné vyhodnocení výsledku

vyhledávání a informace o právě zobrazené části celého seznamu, nebo v druhém případě navigační čísla stránek seznamu.

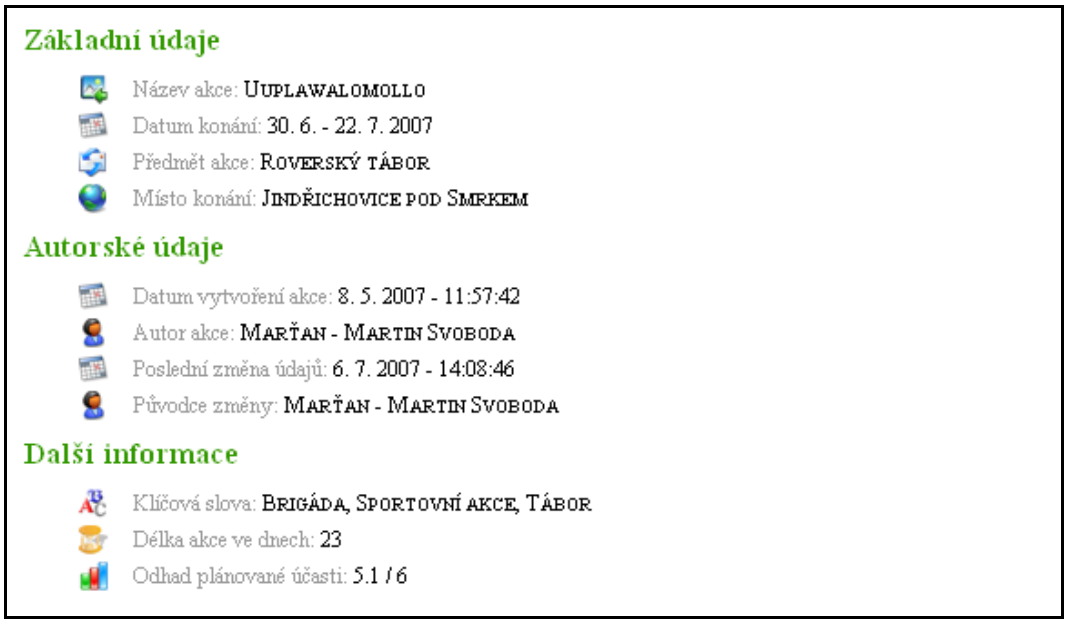
V obou případech dále obsahují navigační tlačítka, která umožňují přejít na první, předcházející, následující a poslední stranu celého seznamu.

5 Popis implementovaných sekcí

5.1 Plánování akcí

Sekce je určena na plánování schůzek, různých jiných setkání či obecně jakýchkoli akcí. Umožňuje zobrazovat roční přehledy konaných akcí, seznamy aktuálních, plánovaných nebo nedávno konaných akcí, stejně jako umožňuje samotné akce vypisovat, vytvářet a editovat.

Další zajímavou funkcí je potvrzování účasti na daných akcích registrovanými uživateli, kteří se mají dané akce zúčastnit. Organizátoři tak mohou mít přehled o tom, jaká bude na dané akci odhadovaná účast.



Obr. 5.1.A: Zobrazení základních údajů o naplánované akci

5.1.1 Vytváření a editace akcí

Vytvářet nové akce mohou všichni registrovaní uživatelé, kterým nebyla odebrána licence na vytváření objektů. V souladu se základní koncepcí hierarchie uživatelských oprávnění mohou následně akci editovat všichni uživatelé, kterým toto oprávnění bylo přiděleno autorem akce, nebo zprostředkovaně od někoho, kdo získal od autora akce oprávnění delegovat.

Základními charakteristikami akce jsou její název, předmět, místo a datum konání. Systém smysluplnost zadaných dat automaticky kontroluje, ale připouští vytváření akcí zpět do minulosti. Toto chování je z jistých důvodů vhodné, stejně jako z opačného pohledu nevhodné.

Pro získání detailnějšího (ale stále přehledného) pohledu na náplň, účel nebo plánovaný program akce, je vhodné dobře zvolit klíčová slova, která akci charakterizují, stejně jako je vhodné zvolit krátký informační popis, který neobsahuje žádné detailní a konkrétní sdělení, ale pouze slouží pro stručné uvedení k akci. Podobně jako u jiných objektů systému se tento popis používá i v nejrůznějších přehledech.

5.1.2 Potvrzování účasti na akcích

Při vytváření objektu akce její autor krom jiného určuje uživatele, pro které je daná akce určena. Všichni tito potenciální účastníci mají možnost před začátkem konání akce (přesněji nejpozději před jejím skončením) odhadnout nejpravděpodobnější míru své plánované účasti.

Přehledy takto potvrzené účasti jsou k dispozici všem, kteří jsou registrovanými uživateli systému a mají oprávnění údaje o dané akci zobrazit. Pokud akce ještě neskončila, objevuje se přehled potvrzených účastníků automaticky přímo v údajích akce, v opačném případě se k seznamu lze také nepřímo dostat.

Samotný odhad účasti se vyjadřuje v krocích od 0%, 10%, … až pod 100% a dále má každý účastník možnost připojit krátkou poznámku, týkající se například informace o omezené účasti apod. Jak odhad tak zmíněnou poznámku je možné opakovaně měnit a zpřesňovat tak odhad.

Obr. 5.1.B: Přehled potvrzených účastníků na vybrané akci

5.1.3 Struktura databázových tabulek

Veškeré objekty plánovaných akcí jsou ukládány do tabulky `actions_actions`. Ta umožňuje uchovat veškeré informace, které se akce přímo týkají – tedy popisné údaje jako název, datum nebo místo konání, dále informace o plánované účasti a nakonec i autorské údaje.

Pro uchovávání záznamů o odhadech potvrzené účasti na akcích slouží tabulka `actions_interests`.

Tabulka položek nástěnky (actions_actions)

	Jméno atributu	Význam atributu a případné poznámky
0	`ai_action`	Identifikační číslo plánované akce.
1	`ao_active`	Příznak platnosti nebo zrušení akce.
2	`ao_keywords`	Popisná klíčová slova přibližující náplň akce.
3	`an_name`	Název samotné akce.
4	`an_info`	Krátký informační popis.
5	`an_subject`	Předmět akce.

6	an_place	Místo konání.
7	an_text	Rozsáhlé textové informace libovolného obsahu.
8	at_start	Datum začátku akce.
9	at_end	Datum konce akce.
10	at_interval	Intervalové datum konání.
11	ac_length	Délka akce ve dnech.
12	ac_interest_sum	Počet uživatelů, kteří odhadli svou účast.
13	ac_interest_max	Očekávaná účast na akci podle odhadů.
14	ax_author	Autor údajů o akci.
15	ax_reader	Uživatel, který údaje naposledy změnil.
16	at_created	Datum a čas vytvoření záznamu akce.
17	at_changed	Datum a čas případné změny údajů.

Tab. 5.1.A: Struktura databázové tabulky plánovaných akcí

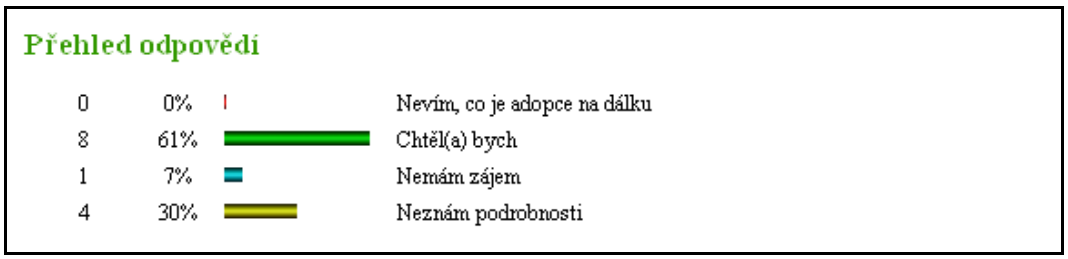
Tabulka položek nástěnky (actions_interests)

	Jméno atributu	Význam atributu a případné poznámky
0	ri_interest	Identifikátor odhadu účasti.
1	rx_action	Plánovaná akce.
2	rx_user	Uživatel, kterého se odhad účasti týká.
3	rc_rating	Odhadovaná míra účasti (od 0.0 do 1.0 v krocích).
4	rn_comment	Doplňující komentář.
5	rt_changed	Datum a čas vytvoření nebo poslední změny.

Tab. 5.1.B: Struktura databázové tabulky potvrzené účasti na jednotlivých akcích

5.2 Anketní otázky

Sekce anketních otázek poskytuje funkcionalitu na vyhledávání anketních otázek, jejich zobrazování, správu anket a rovněž i moduly na samotné hlasování. Oproti běžnému chápaní anketní otázky, jak ji lze vídat na typických internetových stránkách, je tato obohacena o celou řadu rozšiřujících možností.



Obr. 5.2.A: Přehled odpovědí vybrané anketní otázky

Za zmínku stojí především v principu neomezené množství variant jednotlivých odpovědí v anketě, možnost volby barev jejich ukazatelů, možnost výběru konkrétních skupin uživatelů nebo jednotlivých uživatelů, kteří mohou v anketě hlasovat, ale také možnost přidělit každému hlasujícímu předem daný počet hlasů, které má v dané anketě k dispozici, ale také možnost limitovat počty hlasů věnovaných na jednotlivé varianty odpovědí.

Další důležitou vlastností je, že lze vytvářet anketní otázky plně anonymní (určené pro neregistrované návštěvníky), ale i ankety pro registrované uživatele, které se rovněž

mohou chovat jako anonymní. To znamená, že se v databázových tabulkách neuchovávají informace, ze kterých by bylo možné vystopovat původce hlasu.

5.2.1 Druhy anketních otázek

Celkově je možné vybrat si z pěti druhů anketních otázek, z nichž každý má poněkud odlišné vlastnosti, a hodí se tudíž pro mírně odlišný účel použití. Celkový charakter anketní otázky je dán kombinací následujících vlastností. Ne všechny jsou ovšem povoleny. Jednak kvůli smysluplnosti, tak i kvůli implementačním důvodům.

Veřejná anketa / privátní anketa

Pokud je anketa *veřejná*, mohou v ní hlasovat libovolní návštěvníci stránek, aniž by museli být registrováni a přihlášeni. Hlasování je tak v každém případě spojeno jen s konkrétní relací a v žádném případě ne s konkrétním uživatelem (pokud by byl přihlášen). To umožňuje volnější pravidla hlasování, které může být omezováno jen základními prostředky.

V opačném případě *privátní* ankety mohou hlasovat pouze registrovaní a přihlášení uživatelé, kteří jsou navíc standardním způsobem určeni v hierarchii oprávnění k dané anketní otázce. V tomto typu anketní otázky nemohou hlasovat veřejní anonymní návštěvníci.

Viditelná anketa / skrytá anketa

Skrytá anketa má takovou vlastnost, že dokud v ní příslušný uživatel neodhlasuje svůj příděl hlasů nebo dokud se jich nevzdá, neuvidí to, jak hlasovali ostatní uživatelé.

Naopak *viditelná* anketa dosavadní průběh hlasování zobrazuje, takže je nutné počítat s tím, že hlasující uživatel pak může být dosavadními výsledky ovlivněn.

Anonymní anketa / podepisovaná anketa

Jestliže je potřeba mít k dispozici seznamy hlasujících uživatelů včetně počtu a typu vybraných odpovědí, je nutné vybrat si *podepisovaný* typ ankety. V tomto případě jsou tedy jednotlivé hlasy přímo a prokazatelně svázány s konkrétními uživateli a navíc je díky tomu možné omezovat počty hlasů i na úrovni jednotlivých odpovědí.

Anonymní anketa na druhé straně umožňuje provádět anonymní hlasování, ačkoli ho provádějí registrovaní uživatelé. Zmíněná anonymita spočívá v tom, že systém uchovává informace o tom, že daný uživatel hlasoval, ale už si nepamatuje, jakou si vybral variantu odpovědi.

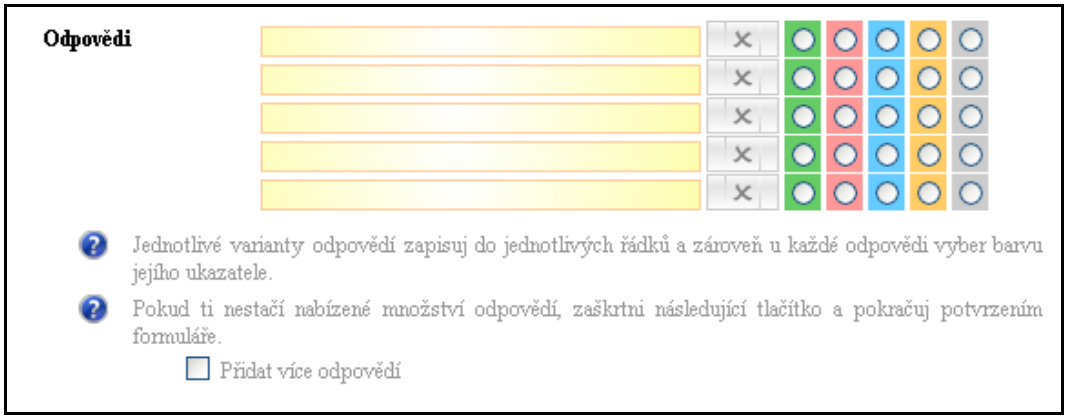
5.2.2 Vytváření a editace anket

Prvním krokem při vytváření nové anketní otázky je výběr jejího typu. V zásadě totiž lze říci, že konkrétní typ do značné míry ovlivňuje nejenom chování a zamýšlený účel použití anketní otázky, ale také i prvky editačního formuláře. Jen některé typy anketních otázek totiž umožňují přímo omezovat počet hlasů na konkrétní odpověď.

Stejně jako i další podobné objekty v systému, lze anketní otázky při jejich vytváření umístit do konkrétního adresáře, včetně možnosti zdědění oprávnění od dané složky.

Anketní otázky je možné plně editovat pouze v době před začátkem jejího aktivního období (viz dále). Toto omezení zabraňuje zpětně změnit koncepci ankety, a tudíž není ohrožena vypovídací hodnota ankety jako takové.

Po začátku hlasování v anketě je už možné pouze měnit konec zmíněného aktivního období. A to jak prodlužovat, tak i zkracovat. Po uzavření anketní otázky, tedy po skončení období určeného na hlasování, už nejsou dovoleny žádné změny.



Obr. 5.2.B: Vytváření nebo editace odpovědí na anketní otázku

5.2.3 Hlasování v anketách

V principu mohou v anketních otázkách hlasovat jak registrovaní uživatelé tak i veřejní návštěvníci. Konkrétní možnosti závisí na vybraném typu anketní otázky (viz výše). Ať už je okruh hlasujících jakýkoli, hlasovat se může pouze v průběhu aktivního období ankety – tedy období určeného na hlasování.

V případě skrytých typů anketních otázek uživatel nevidí dosavadní průběh hlasování, dokud nevyčerpá všechny své přidělené hlasy. V některých případech autor ankety nutně nepředpokládá vyčerpání všech přidělených hlasů, a proto má v těchto případech hlasující možnost svých hlasů se vzdát. Tento proces je nevratný.

5.2.4 Struktura databázových tabulek

Objekty anketních otázek jsou uchovávány v tabulce `inquiry_queries`. Ta obsahuje základní informace o anketě jako takové, informace o období hlasování, základní statistické údaje o hlasování, limitující vlastnosti anketní otázky a autorské údaje.

Texty jednotlivých variant odpovědí na anketní otázky jsou uloženy v tabulce `inquiry_answers`. Důležitou vlastností každé varianty odpovědi je barva ukazatele, který je použit při vypisování anketní otázky.

Samotné hlasy uživatelů jsou uloženy v tabulce `inquiry_votes`. Ta umožňuje zaznamenat nejenom skutečně provedené hlasy uživatelů, ale je použita i pro uchovávání informace o tom, že se daný uživatel v dané anketní otázce dobrovolně vzdal zbylých hlasů ze svého přídělu.

Tabulka anketních otázek (inquiry_queries)

	Jméno atributu	Význam atributu a případné poznámky
0	`qi_inquiry`	Identifikační číslo anketní otázky.
1	`qn_name`	Název anketní otázky.
2	`qn_question`	Vlastní text otázky.
3	`qo_type`	Typ ankety. Jde o třímístnou bitovou mapu určující jednotlivé vlastnosti ankety (a tudíž její typ).
4	`qt_start`	Datum začátku období hlasování.
5	`qt_end`	Datum uzavření ankety.
6	`qt_interval`	Intervalové datum období hlasování.
7	`qt_length`	Délka období hlasování ve dnech.
8	`qc_quota`	Příděl hlasů na každého hlasujícího.
9	`qc_limit`	Omezení počtu hlasů na jednu odpověď. Týká se jen některých typů anketních otázek.
10	`qc_answers`	Počet variant odpovědí.
11	`qc_votes`	Celkový počet odvolených hlasů.
12	`qx_author`	Autor anketní otázky.
13	`qx_reader`	Uživatel, který provedl poslední změnu ankety.
14	`qt_created`	Datum a čas vytvoření ankety.
15	`qt_changed`	Datum změny ankety.

Tab. 5.2.A: Struktura databázové tabulky anketních otázek

Tabulka variant odpovědí (inquiry_answers)

	Jméno atributu	Význam atributu a případné poznámky
0	`ai_inquiry`	Číslo ankety, k níž daná odpověď patří.
1	`ai_answer`	Pořadové číslo odpovědi v rámci ankety.
2	`an_text`	Text odpovědi.
3	`ac_color`	Barva ukazatele.
4	`ac_votes`	Informativní počet existujících hlasů.

Tab. 5.2.B: Struktura databázové tabulky variant odpovědí anketních otázek

Tabulka existujících hlasů (inquiry_votes)

	Jméno atributu	Význam atributu a případné poznámky
0	`vi_inquiry`	Číslo anketní otázky.
1	`vi_answer`	Číslo varianty odpovědi. Pokud jde o anonymní anketu, údaj je prázdný. Pokud jde o záznam o vzdání se hlasů, obsahuje `0`.
2	`vx_voter`	Identifikace hlasujícího uživatele. Pokud jde o veřejnou anketu, údaj je prázdný.
3	`vx_visit`	Identifikace relace. Nutné pro veřejné typy anket.
4	`vt_created`	Datum a čas provedeného hlasování.

Tab. 5.2.C: Struktura databázové tabulky hlasů v anketních otázkách

5.3 Fotografická alba

Sekce fotografických alb slouží především k přehlednému seskupování fotografií do jednotlivých alb. Typicky se předpokládají opravdu fotografie, ale obdobně mohou být vkládány i obrázky jiného charakteru, samozřejmě se ztrátou některé specifické funkčnosti (především automatického popisování fotografií pomocí údajů EXIF přidávaných digitálními fotoaparáty).

Kromě standardních přehledů fotografických alb podle roků, přehledů aktuálních alb, modulů na vytváření a editaci alb, je velká pozornost věnována procesu vkládání jednotlivých fotografií do alb a také na charakterizaci obsahu samotných fotografií.

Po vytvoření nového alba tak mají všichni uživatelé s oprávněním album editovat možnost přímo vkládat nebo mazat fotografie v albu obsažené. Systém automaticky nahrané fotografie zpracuje, v případě potřeby zmenší jejich rozměry a také vygeneruje náhledový obrázek – miniaturu, která se zobrazuje v přehledu fotografií alba.

Obr. 5.3.A: Výpis fotografií vybraného alba

5.3.1 Vytváření a editace alb

Prvním a nepovinným krokem při vytváření nového alba fotografií je jeho umístění do vybrané složky. Stejně jako u ostatních obdobných objektů systému není povinné album do nějaké složky umístit, stejně jako je možné jej umístit do více složek najednou.

Kromě základních popisných údajů alba – jako název a krátký informační popis – je ještě nutné zadat období focení fotografií. Toto období je nutné při změně obsahu alba ručně kontrolovat vůči případné změně skutečného období, protože v některých

případech by automatická kontrola například vůči údajům z EXIF nemusela být požadovaným chováním.

Dalším krokem ve vytváření nového alba nebo editaci existujícího je výběr autorů fotografií, které jsou v albu umístěny. V seznamu jsou obsaženi všichni uživatelé, kteří jsou v danou chvíli viditelní z hlediska zařazení uživatelů do jednotlivých skupin. Všichni takto vybraní autoři fotografií jsou uvedeni jako autoři alba bez spojitosti s konkrétními fotografiemi. Autorství každé jednotlivé fotografie lze také nastavit (ve vlastnostech fotografie) – a k tomu slouží právě zde vytvoření seznam.

Povinnou částí procesu vytváření nového alba je nahrání malého náhledového obrázku celého alba. Tento obrázek musí splňovat požadované vlastnosti a používá se v jednotlivých přehledech alb.

Stejně jako všechny ostatní položky i obrázek je možné kdykoli později znovu editovat.

5.3.2 Nahrávání a mazání fotografií

Vkládání fotografií do alba se děje přímo přes standardní editační formuláře a může jej provádět každý uživatel, který má oprávnění dané album editovat.

Uživatel ve standardním dialogu na upload souborů vybere všechny fotografie, které chce do alba vložit. Pokud mu nestačí nabízené množství vstupních polí, může v rámci jednoho editačního procesu nahrávání opakovat. Daleko důležitější je ovšem možnost nahrávat fotografie z archivů ZIP. Systém automaticky takovýto archiv rozpozná, zpracuje ho a jednotlivé fotografie z něj extrahuje.

Obecně se předpokládá, že budou vkládány pouze smysluplné soubory fotografií, a to ještě navíc v podobě, která jednotlivé soubory fotografií postupně čísluje (souvislost se nepožaduje). Tento požadavek je smysluplný hned ze dvou důvodů. Jednak digitální fotoaparáty fotografie číslují, v druhém případě pak určitá prvotní úprava fotografie uživatelem je stejně nutná, protože nemá smysl na server nahrávat násobně velké fotografie, když budou stejně zmenšeny.

Pokud budou nahrány soubory, které nejsou platnými fotografiemi, budou odmítnuty. Špatně pojmenované soubory budou automaticky přečíslovány a fotografie, které mají větší rozměry než nastavené maximální, budou automaticky zmenšeny.

Náhledový obrázek každé fotografie je také vytvořen automaticky. Dalším z kroků přidávání fotografií do alba je automatická detekce údajů EXIF a jejich uložení do databáze spolu s dalšími informacemi o fotografii.

Kromě modulu na hromadné nahrávání fotografií do alba je dále možné přímo nahrazovat soubor konkrétní fotografie, která už v albu je. I v tomto případě systém automaticky zmenší velkou fotografii, vytvoří náhled a načte případné EXIF informace.

Fotografie lze z alba také jednoduchým způsobem mazat. Takovéto vymazání je nevratné, protože všechny soubory a záznamy jsou opravdu vymazány.

5.3.3 Vlastnosti fotografií a popisky

Pokud soubor fotografie obsahuje informace EXIF přidávané automaticky digitálními fotoaparáty, jsou tyto informace v průběhu přidávání fotografie do alba automaticky načteny a zpracovány. Jde zejména o datum a čas vyfocení, výrobce a model použitého digitálního fotoaparátu, ale i ostatní informace jako režim, ISO, clonové číslo či délka expozice.

Pokud se nepodařilo tyto údaje načíst, je možné datum a čas v modulu na editaci vlastností fotografie vyplnit ručně. To je vhodné zejména kvůli dobrému uspořádání fotografií při vypisování obsahu alba. Dále je u každé fotografie možné vybrat jejího autora. Vybírá se ovšem pouze z omezeného seznamu, který byl vytvořen ve vlastnostech samotného alba (viz výše).

Dále je možné u každé fotografie vybrat klíčová slova, která charakterizují její obsah. Můžeme mluvit například o portrétu, krajině, sportu apod. Rovněž je možné zaznamenat, kteří konkrétní uživatelé jsou na dané fotografii vyfoceni.

Všechny tyto údaje má oprávnění editovat pouze ten uživatel, který má oprávnění editace daného alba. Obecně širší skupina uživatelů, kteří mají právo reflexe, mohou dále vytvářet popisky k fotografiím. Každý uživatel v tomto případě vytváří svůj vlastní individuální popisek a při zobrazování fotografie se vypisují všechny tyto příspěvky.

Obr. 5.3.B: Výpis popisných informací připojených k fotografii alba

5.3.4 Struktura databázových tabulek

Na uchovávání záznamů o fotografických albech slouží tabulka `photos_albums`. Kromě základních popisných a autorských údajů obsahuje ještě informační počet fotografií v albu obsažených a celkový počet popisků k fotografiím. Seznam uživatelů, kteří jsou vybráni jako autoři fotografií, je uložen v tabulce `photos_authors`.

Záznamy o samotných fotografiích jsou v tabulce `photos_photos`. Obsahuje všechny identifikační údaje, technické (jako velikosti fotografie a náhledu) i popisné údaje včetně hodnot převzatých z EXIF. Kvůli komprimaci místa je zavedena speciální tabulka `photos_cameras`, která identifikuje jednotlivé modely používaných fotoaparátů. Seznam uživatelů, kteří jsou vyfoceni na fotografiích, je uložen v tabulce `photos_snapped` a nakonec pro uchovávání popisků k fotografiím slouží tabulka `photos_captions`.

Tabulka fotografických alb (photos_albums)

	Jméno atributu	Význam atributu a případné poznámky
0	`ai_album`	Identifikační číslo fotografického alba.
1	`an_name`	Název alba.
2	`an_info`	Krátký informační popis alba.

3	at_start	Datum začátku období focení fotografií.
4	at_end	Datum konce tohoto období.
5	at_interval	Intervalové datum focení.
6	ac_photos	Počet fotografií umístěných v album.
7	ac_captions	Počet popisků k fotografiím daného alba.
8	ax_author	Autor informací o albu.
9	ax_reader	Uživatel, který informace naposledy změnil.
10	at_created	Datum a čas vytvoření alba.
11	at_changed	Datum a čas případné poslední změny.

Tab. 5.3.A: Struktura databázové tabulky fotografických alb

Tabulka autorů fotografií (photos_authors)

	Jméno atributu	Význam atributu a případné poznámky
0	ax_album	Číslo fotografického alba.
1	ax_author	Záznam o vybraném autoru fotografií v albu.

Tab. 5.3.B: Struktura databázové tabulky autorů fotografií

Tabulka fotografií (photos_photos)

	Jméno atributu	Význam atributu a případné poznámky
0	pi_album	Číslo fotografického alba.
1	pi_photo	Identifikační číslo konkrétní fotografie. Toto číslo odpovídá názvu souboru bez přípony.
2	ps_order	Výchozí řazení fotografií v albu.
3	pt_date	Datum digitalizace.
4	pt_time	Čas digitalizace.
5	px_author	Konkrétní uživatel, který je autorem fotografie.
6	po_keywords	Klíčová slova.
7	pc_snapped	Počet vyfocených uživatelů na dané fotografii.
8	pk_photo_x	Šířka fotografie v pixelech.
9	pk_photo_y	Výška fotografie.
10	pk_thumb_x	Šířka náhledového obrázku.
11	pk_thumb_y	Výška náhledu.
12	pk_camera	EXIF – identifikátor fotoaparátu.
13	pk_program	EXIF – režim focení.
14	pk_iso	EXIF – citlivost ISO.
15	pk_exposure	EXIF – délka expozice.
16	pk_fnumber	EXIF – clonové číslo
17	pk_focus	EXIF – ohnisko
18	pc_opened	Počet zobrazení fotografie.
19	pt_created	Datum a čas vložení fotografie do alba.

Tab. 5.3.C: Struktura databázové tabulky položek fotografií

Tabulka digitálních fotoaparátů (photos_cameras)

	Jméno atributu	Význam atributu a případné poznámky
0	mi_camera	Identifikační číslo fotoaparátu.
1	mn_make	EXIF – název výrobce.
2	mn_model	EXIF – model fotoaparátu.

Tab. 5.3.D: Struktura databázové tabulky digitálních fotoaparátů

Tabulka vyfocených uživatelů (photos_snapped)

	Jméno atributu	Význam atributu a případné poznámky
0	`sx_album`	Číslo fotografického alba.
1	`sx_photo`	Číslo fotografie v albu.
2	`sx_user`	Záznam o vyfoceném uživateli na fotografii.

Tab. 5.3.E: Struktura databázové tabulky vyfocených uživatelů na fotografiích

Tabulka popisků fotografií (photos_captions)

	Jméno atributu	Význam atributu a případné poznámky
0	`ci_caption`	Identifikační číslo popisku.
1	`cx_album`	Číslo fotografického alba.
2	`cx_photo`	Číslo fotografie v albu.
3	`cn_text`	Vlastní text popisku.
4	`cx_author`	Identifikátor autora popisku.
5	`ct_created`	Datum a čas vytvoření.
6	`ct_changed`	Datum a čas poslední změny.

Tab. 5.3.F: Struktura databázové tabulky popisků jednotlivých fotografií

5.3.5 Ukládání souborů fotografií

Všechny soubory fotografií, náhledových obrázků jednotlivých fotografií a náhledový obrázek alba jsou ukládány v datovém adresáři sekce fotografií. Poslední zmiňovaný obrázek je uložen jako `data/photos/albums/$album.jpg`, kde `$album` je číslo daného fotografického alba. Obdobně soubory fotografií resp. náhledů jsou uloženy jako `data/photos/{photos|thumbs}/$album/$photo.jpg`. V tomto případě je `$photo` identifikačním číslem fotografie.

Při vytváření nového alba resp. při vkládání fotografií do alba se potřebná adresářová struktura vytvoří automaticky.

5.4 Diskuzní knihy

Sekce diskuzí nabízí dva prostředky, které umožňují diskutování uživatelů nad různými tématy na různých místech v systému.

První standardní řešení vytváří model diskutování v samostatných knihách. Knihou můžeme rozumět ucelený soubor diskuzních témat, které spolu nějakým způsobem podle autorova záměru souvisí. Vlastní témata diskuze jsou realizována pomocí diskuzních knih. Teprve do nich uživatelé nebo návštěvníci své příspěvky zapisují.

Druhé řešení představuje distribuovanou diskuzi k ostatním objektům systému, které to umožňují. Jde například o plánované akce nebo položky nástěnky. Uživatelé tak díky funkcím diskuzních komentářů mají možnost psát příspěvky, které se přímo zobrazují u těchto objektů.

V prvním případě je diskuze strukturována a uživatelé mají možnost na sebe reagovat s přímými odkazy, v druhém případě nikoli.

Obr. 5.4.A: Výpis diskuzních kapitol vybrané knihy

5.4.1 Vytváření e editace knih a kapitol

Prvním krokem ve vytvoření nové diskuze je vytvoření nové diskuzní knihy. Stejně jako u jiných sekcí, vytvářet nové knihy mohou všichni registrovaní uživatelé, kterým toto oprávnění nebylo odebráno. Rovněž údaje zaznamenané v knize je možné později libovolně měnit.

Důležitým krokem při vytváření nebo následné editaci knihy je určení hierarchie oprávnění. Lze tak vymezit skupiny uživatelů, kteří mají oprávnění knihy procházet, v knihách diskutovat, spravovat knihy a kapitoly, nebo v nejvyšší úrovni oprávnění delegovat samotná oprávnění.

Po vytvoření nové knihy je nutné do ní nejprve přidat jednotlivé diskuzní kapitoly. Ty představují ucelená témata diskuze, která patří do dané knihy. Každá kapitola je tedy součástí právě jedné knihy. Obdobně jako u knih i kapitoly je možné libovolně editovat, pouze oprávnění jsou určena nadřazenou knihou a platí pro všechny její kapitoly.

5.4.2 Diskutování v kapitolách

Diskuze v kapitolách umožňují, aby diskutujícími byli jak registrovaní uživatelé, tak i veřejní návštěvníci. Jak již bylo řečeno výše, konkrétní skupinu oprávněných diskutujících vybírá autor v okamžiku vytváření knihy (resp. později při editacích).

Pokud kniha umožňuje diskutování veřejných návštěvníků, mají možnost se pod svůj příspěvek podepsat a zaznamenat i emailovou adresu. Tyto údaje se kromě diskusního příspěvku uchovávají i dočasně v datech relace, takže pokud daný návštěvník má v úmyslu napsat více příspěvků, nemusí tyto údaje opakovaně vypisovat a jen je potvrdí.

Samotná diskuze v kapitolách představuje strukturovanou formu diskuze, kdy každý diskutující má možnost buď vytvořit v dané kapitole nový kmenový příspěvek, nebo v druhém případě reaguje na jiný již existující příspěvek. Z matematického pohledu tedy můžeme o kapitole mluvit jako o standardním lese, kde každý strom je uvozen kmenovým příspěvkem.

Registrovaní uživatelé dále mají možnost krátce po vytvoření nového diskusního příspěvku ho ještě editovat. Mohou tak například opravovat překlepy a jiné chyby.

Obr. 5.4.B: Příklad strukturované diskuze v rámci diskuzní kapitoly

5.4.3 Diskuzní komenáře

Diskuzní komentáře představují odlišný typ diskutování. Především nejsou soustředěny v sekci diskuzí, ale jde o službu, kterou tato sekce poskytuje jiným. Konkrétně například akcím. Každý registrovaný uživatel s příslušnými oprávněními má tak možnost psát své diskuzní reakce přímo k dané akci.

V tomto typu diskuze se nevytvářejí žádné knihy ani kapitoly, diskuzní příspěvky jsou přímo propojeny s konkrétními objekty již zmíněného typu akce apod. Sekce diskuzí pouze všechny tyto komentáře zastřešuje. To znamená, že obsahuje vlastní moduly na vytváření a editaci takovýchto příspěvků, ale také poskytuje moduly na jejich vyhledávání.

Důležitým rozdílem je, že v tomto typu diskuze nemohou diskutovat neregistrovaní návštěvníci, a především tato diskuze není strukturovaná. Jednotlivé příspěvky se lineárně řadí za sebou podle toho, kdy byly vytvořeny. Klasické reakce na konkrétní příspěvky tudíž není možné psát.

Stejně jako u normálních příspěvků, i komentáře lze krátce po jejich vytvoření editovat a opravovat v nich chyby.

5.4.4 Přehledy nových příspěvků

Automatické přehledy a vyhledávání novinek je sice standardní součástí i ostatních sekcí, nicméně u diskuzí je tato potřeba možná o to více důležitější. V diskuzích je proto možné vyhledávat nejenom nové či pozměnění knihy a kapitoly, ale především nové diskuzní příspěvky a diskuzní komentáře.

Registrovaní uživatelé tak mohou mít velmi dobrý přehled o tom, kde se zrovna diskutuje, jaké příspěvky si už přečetli a jaké nikoli. Kvůli velkému objemu dat tyto záznamy samozřejmě nelze vést individuálně pro každého uživatele a každý záznam, takže se využívá standardního mechanismu jádra na vyhledávání novinek – a hledání probíhá pouze podle aktuálně nastaveného období.

Informace o nových příspěvcích (či dokonce o autorovi posledního příspěvku, byl-li jím registrovaný uživatel) jsou zobrazovány ve standardních přehledech knih, stejně jako přehledech kapitol v knihách. Pro snazší přístupnost slouží zvláštní modul, který automaticky vypisuje všechny kapitoly, které obsahují nové příspěvky.

Obdobný mechanismus vyhledávání novinek funguje i u diskuzních komentářů. Výsledky jsou seskupovány podle typů objektů (akce apod.) a obsahují přímé odkazy na tyto objekty.

5.4.5 Struktura databázových tabulek

Diskuzní knihy jsou ukládány v tabulce `debates_books`. Pro jednotlivé diskuzní kapitoly knihy slouží tabulka `debates_chapters`. Obě tyto tabulky obsahují popisné údaje, ze kterých lze určit především účel zřízení dané knihy nebo kapitoly. Dále obsahují informativní přehledy o probíhající diskuzi – např. počty příspěvků.

Pro ukládání samotných příspěvků je určena tabulka `debates_notes` a pro diskuzní komentáře `debates_comments`. Obě tyto tabulky jsou podobné, nicméně mají odlišný způsob identifikace příslušnosti záznamů a příspěvky navíc umožňují uchovávat informace o autorech příspěvků, kteří nejsou registrovanými uživateli.

Tabulka diskuzních knih (debates_books)

	Jméno atributu	Význam atributu a případné poznámky
0	`bi_book`	Identifikační číslo diskuzní knihy.
1	`bn_name`	Název knihy.
2	`bn_info`	Krátký informační popis.
3	`bn_text`	Doplňující podrobný text.
4	`bt_start`	Datum začátku probíhající diskuze. Pokud diskuze nezačala, je údaj prázdný.
5	`bt_end`	Obdobně konec období diskutování.
6	`bt_interval`	Intervalové datum tohoto období.
7	`bc_chapters`	Počet kapitol knihy.
8	`bc_notes`	Celkový počet příspěvků v knize.
9	`bx_author`	Autor diskuzní knihy.
10	`bx_reader`	Uživatel, který jako poslední provedl změny.
11	`bx_last`	Uživatel, který jako poslední napsal příspěvek. Pokud je to anonymní návštěvník, je údaj prázdný.
12	`bt_created`	Datum a čas vytvoření knihy.
13	`bt_changed`	Datum a čas případné poslední změny.

Tab. 5.4.A: Struktura databázové tabulky diskuzních knih

Tabulka kapitol knih (debates_chapters)

	Jméno atributu	Význam atributu a případné poznámky
0	ci_book	Číslo diskuzní knihy.
1	ci_chapter	Pořadové číslo kapitoly v rámci knihy.
2	cn_name	Název diskuzní knihy.
3	cn_info	Krátký informační popis knihy.
4	ct_start	Datum začátku probíhající diskuze. Údaj je prázdný, pokud diskuze nezačala.
5	ct_end	Konec tohoto období.
6	ct_interval	Intervalové datum tohoto období.
7	cc_notes	Celkový počet příspěvků v kapitole.
8	cx_author	Autor diskuzní kapitoly.
9	cx_reader	Uživatel, který jako poslední provedl změny.
10	cx_last	Uživatel, který jako poslední napsal příspěvek. Pokud je to anonymní návštěvník, je údaj prázdný.
11	ct_created	Datum a čas vytvoření kapitoly.
12	ct_changed	Datum a čas případné poslední změny.

Tab. 5.4.B: Struktura databázové tabulky kapitol diskuzních knih

Tabulka diskuzních příspěvků (debates_notes)

	Jméno atributu	Význam atributu a případné poznámky
0	ni_note	Identifikační číslo diskusního příspěvku.
1	ni_book	Číslo knihy.
2	ni_chapter	Číslo kapitoly.
3	no_status	Typ příspěvku. Vyhrazeno pro budoucí použití.
4	nn_name	Titulek příspěvku.
5	nn_text	Vlastní text příspěvku.
6	nc_level	Úroveň ve stromu. Kořen má hodnotu 0.
7	nx_parent	Nadřazený příspěvek ve stromu.
8	nx_author	Autor příspěvku, je-li jím registrovaný uživatel.
9	nn_anonym	Jméno autora, je-li jím anonymní návštěvník.
10	nn_email	Emailová adresa takovéhoto návštěvníka.
11	nt_created	Datum a čas vytvoření příspěvku.
12	nt_changed	Datum a čas případné poslední změny.

Tab. 5.4.C: Struktura databázové tabulky příspěvků v diskuzních kapitolách knih

Tabulka diskuzních komentářů (debates_comments)

	Jméno atributu	Význam atributu a případné poznámky
0	ri_note	Identifikační číslo diskusního komentáře.
1	rx_section	Kód sekce odpovídající nadřazenému objektu.
2	rx_type	Kód typu objektu.
3	rx_object	Identifikační číslo objektu.
4	rn_name	Titulek komentáře.
5	rn_text	Vlastní text komentáře.
6	rx_author	Autor komentáře.
7	rt_created	Datum a čas vytvoření komentáře.
8	rt_changed	Datum a čas případné poslední komentáře.

Tab. 5.4.D: Struktura databázové tabulky diskuzních komentářů

5.5 Položky nástěnky

Sekce nástěnky poskytuje univerzální prostředky na uchovávání textových informací více či méně libovolného charakteru. Každá položka, která je určena k vyvěšení na nástěnce, má určeno aktivní období, po které je aktuální a bude na nástěnce vyvěšena mezi aktuálními položkami.

To samozřejmě nezabraňuje standardním způsobem vyhledávat mezi všemi existujícími položkami nástěnky například pomocí ročních výpisů.

Jak již bylo řečeno, nepředpokládá se žádná hlubší strukturovanost takovýchto textů. Pouze každá položka má svůj název, předmět, krátký informační popis používaný v přehledech a pro ještě větší zvýšení možnosti jednotlivé položky elegantně charakterizovat, je možné využívat klíčová slova, obdobně jako u akcí nebo fotografií.

5.5.1 Struktura databázových tabulek

Jedinou databázovou tabulkou této sekce je `board_reports`. Ta obsahuje veškeré informace jednotlivých položek nástěnky – od popisných údajů, vlastního textu až po autorské údaje.

Tabulka položek nástěnky (board_reports)

	Jméno atributu	Význam atributu a případné poznámky
0	`bi_report`	Identifikační číslo položky nástěnky.
1	`bo_keywords`	Klíčová slova charakterizující obsah položky.
2	`bn_name`	Název položky.
3	`bn_info`	Krátký informační popis.
4	`bn_subject`	Předmět nebo věc položky.
5	`bn_text`	Vlastní text položky.
6	`bt_start`	Datum začátku období vyvěšení.
7	`bt_end`	Datum konce tohoto období.
8	`bt_interval`	Totéž období jako intervalové datum.
9	`bx_author`	Autor položky.
10	`bx_reader`	Uživatel, který provedl poslední změny.
11	`bt_created`	Datum a čas vytvoření.
12	`bt_change`	Datum a čas poslední případné změny.

Tab. 5.5.A: Struktura databázové tabulky položek nástěnky

5.6 Pracovní složky

Sekce složek poskytuje v zásadě dvě základní funkce. Tou první a nejdůležitější je vytváření virtuálních složek, do kterých lze umisťovat řadu objektů z ostatních sekcí systému. Účel takovýchto složek je jednoduchý – díky nim lze uživatelům poskytnout poměrně zásadní nástroj k tomu, aby mohli shlukovat informace, které spolu věcně souvisí, a díky tomu mohly být následně takto propojené objekty snadněji vyhledávány.

Objekty, které lze do složek vkládat, jsou například plánované akce, položky nástěnky, diskuzní knihy, fotografická alba, ale i uploadované soubory, o kterých bude řeč později právě s rámci této sekce.

5.6.1 Vkládání objektů do složek

Prvotním krokem při používání složek je samozřejmě vytvoření samotné složky. Ta obsahuje pouze základní popisné údaje, konkrétně název a krátký informační popis. Měly by být voleny tak vhodně, aby z nich uživatel získal jasnou představu o tom, čeho se objekty ve složce týkají.

Poměrně významným krokem při vytváření nové složky nebo její následné editace je určení uživatelských oprávnění. Ta se totiž týkají nejenom samotné složky, ale lze jich využít i v případě, kdy jsou vytvářeny nové objekty jako akce apod., které jsou přímo v procesu jejich vytváření do nějaké složky již umístěny.

Přesněji řečeno, uživatel vytvářející například novou diskuzní knihu ji může hned v počátku umístit do vybrané složky a následně má možnost pro novou knihu zdědit veškerá oprávnění, která se týkají samotné složky.

Význam takovéhoto dědění je snadný – účelem složek je seskupovat informace, které spolu nějakým způsobem souvisí – lze tedy předpokládat, že i nároky na delegaci oprávnění na různých vrstvách hierarchie budou podobné. V každém případě je toto dědění jen pomůckou pro uživatele a není tím nijak omezena možnost oprávnění pro nový objekt jakkoli později upravovat.

Důležité je také to, že takovýto proces dědění je možné provést pouze v okamžiku vytváření nových objektů, a dále, že samotné dědění je jednorázovou záležitostí, během které se nevytváří žádné pevné vazby mezi složkou a objektem. Pokud tedy budou někdy později upravena oprávnění ke složce, jednotlivé objekty tím nebudou automaticky dotčeny.

Pokud jde o samotná oprávnění složek vůči jím samotným, určují zejména to, kteří uživatelé mají oprávnění zobrazovat obsah složek, nebo vkládat do složek nové objekty.

5.6.2 Uploadování souborů

Druhou službou, kterou tato sekce nabízí, je uploadování souborů od klientů. Ty se po úspěšném procesu nahrání uloží na server a logicky se zařadí do složek, úplně stejně, jako ostatní objekty stránek, o kterých se mluvilo výše.

Nahrávání takovéhoto souboru se děje pomocí standardního editačního formuláře a soubor může být opakovaně přepisován jeho novými verzemi. Kromě souboru jako takového, obsahuje ještě logický záznam o soboru ještě řadu dalších informačních a popisných údajů.

Jde zejména o popisné jméno. Tedy nikoli systémové jméno souboru, ale název, který vystihuje obsah souboru a pod kterým daný soubor vystupuje na stránkách, obdobně jako všechny ostatní objekty. Dále soubor obsahuje krátký popis. Tyto údaje je možné libovolně editovat, nicméně tak může činit pouze takový uživatel, který daný soubor vytvořil.

Důležitým faktem je, že každý uploadovaný soubor musí mít určenu domovskou složku. Jde o složku, ve které je daný soubor fyzicky uložen. Platí tudíž, že v rámci jedné složky není možné, aby v ní bylo více souborů, které by měly stejná fyzická jména. Systém tuto věc automaticky kontroluje a v případě potřeby soubory přejmenuje tak, aby ke kolizi nedošlo.

Tím není narušena možnost, aby soubor byl logickou součástí i jiných složek. Fyzicky je nicméně uložen jako `data/folders/$folder/$file`, kde `$folder` je číslo domovské složky a `$file` je fyzické jméno souboru včetně přípony.

5.6.3 Struktura databázových tabulek

Sekce složek obsahuje celkem tři databázové tabulky. První z těchto tabulek je `fodlers_folders`, která slouží pro uchování informací o existujících logických složkách. Samotné informace o umístění jednotlivých objektů do těchto složek jsou uloženy v tabulce `folders_links`.

Tabulka `folders_files` je určena na uchovávání informací o uploadovaných souborech. Samotná data těchto souborů jsou uložena mimo databázi, jak již bylo vysvětleno výše.

Tabulka složek (folders_folders)

	Jméno atributu	Význam atributu a případné poznámky
0	`fi_folder`	Identifikační číslo složky
1	`fn_name`	Název složky.
2	`fn_info`	Krátký informační text.
3	`fa_author`	Autor složky.
4	`fc_items`	Počet položek umístěných ve složce.
5	`ft_created`	Datum a čas vytvoření složky.
6	`ft_touched`	Datum a čas poslední případné změny.

Tab. 5.6.A: Struktura databázové tabulky složek

Tabulka umístění objektů do složek (folders_links)

	Jméno atributu	Význam atributu a případné poznámky
0	`ri_folder`	Identifikační číslo vybrané složky.
1	`rx_type`	Kód typu objektu.
2	`rx_item`	Identifikační číslo umisťovaného objektu.

Tab. 5.6.B: Struktura databázové tabulky umístění jednotlivých objektů do složek

Tabulka uploadovaných souborů (folders_files)

	Jméno atributu	Význam atributu a případné poznámky
0	`si_file`	Identifikační číslo souboru.
1	`sn_file`	Původní název souboru.
2	`sx_folder`	Domovská složka, kde je soubor umístěn.
3	`sn_name`	Popisný název souboru.
4	`sn_info`	Krátký informační popis obsahu souboru.
5	`sx_author`	Autor přílohy.
6	`so_mime`	Rozpoznaný MIME typ souboru.
7	`sc_size`	Velikost souboru v bytech.
8	`sc_opened`	Počet downloadů souboru.
9	`st_created`	Datum a čas prvního uploadu.
12	`st_changed`	Datum a čas poslední případné změny údajů.

Tab. 5.6.C: Struktura databázové tabulky uploadovaných souborů

6 Závěr

Už od počátku vývoje tohoto systému bylo jisté, že bude mít své reálné nasazení. To se později ukázalo jako poměrně podstatné – nejenom kvůli zanedbatelnému přizpůsobení výchozích nastavení systému, ale zejména kvůli možnosti věnovat realizaci takto rozsáhlého projektu potřebné množství času.

Kvalitativní hledisko naplnění vytyčených cílů mohou ukázat až konkrétní uživatelé, tudíž hodnotit na tomto místě lze pouze kvantitativní rámec naplnění předem daných specifikací systému. V tomto smyslu lze říci, že se podařilo implementovat vše, co bylo původním záměrem.

To samozřejmě neznamená, že řešení úlohy je naplněno a není možné projekt dále rozšiřovat. Už v průběhu samotné práce na implementaci projektu se neustále objevovaly nové návrhy na další funkcionalitu, včetně návrhů celých sekcí, jak je tento pojem v systému používán.

Řadu z těchto nápadů se podařilo implementovat už v rámci této bakalářské práce, ostatní se možná realizace nedočkají nikdy. Jisté je ale to, že díky osobním zájmům bude práce na projektu pokračovat.

Použité informační zdroje

- PHP Group: *Online PHP Documentation*, http://www.php.net
- MySQL AB: *MySQL Documentation*, http://www.mysql.com
- World Wide Web Consortium: *XHTML 1.0 Specification*, http://www.w3.org
- World Wide Web Consortium: *CSS 2.1 Specification*, http://www.w3.org
- Eric Meyer: *Eric Meyer on CSS*, http://www.ericmeyeroncss.com

Obsah přiloženého CD

- Zdrojové soubory informačního systému
- Postup instalace na webovém serveru
- Elektronický text této bakalářské práce