Trojrozměrná správa přístupových práv pro webové systémy

Three-dimensional administration of access rights for web systems

Tomáš Marcaník

Bakalářská práce
2010

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2009/2010

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Tomáš MARCANÍK**
Osobní číslo:	**A07060**
Studijní program:	**B 3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Téma práce:	**Trojrozměrná správa přístupových práv pro webové systémy**

Zásady pro vypracování:

1. **Analyzujte stávající systémy správy přístupových práv pro webové aplikace, které umožňují vytvářet a upravovat hierarchii uživatelů a jejich skupin a takto definovaným uživatelským objektům pak dynamicky přidělovat obecná práva (tj. N různých práv) k hierarchii objektů aplikace.**
2. **V rámci analýzy proveďte srovnání paměťové a časové složitosti těchto systémů.**
3. **Vytvořte vzorovou GUI komponentu pro správu přístupových práv, která bude umět využít všech vlastností daného systému.**
4. **Výsledný systém bude zveřejněn pod open source licencí New BSD.**

Rozsah bakalářské práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **Nette Foundation. Příručka programátora [online]. 2008, 2010 [cit. 2010-02-05]. Dostupný z WWW: http://guides.nettephp.com/cs/.**
2. **Nette Foundation. API reference [online]. c2009 [cit. 2010-02-05]. Dostupný z WWW: http://api.nettephp.com/0.9.2/.**
3. **BENOIT, Mike, RUSSELL, James, DAMBEKALNS, Karsten. Generic Access Control Lists with PHP. PhpGACL [online]. 2002 - 2006, version 60 [cit. 2010-02-05]. Dostupný z WWW: http://phpgacl.sourceforge.net/manual.pdf.**
4. **Zend Technologies. Programmers Reference Guide : Zend_Acl [online]. c2006-2010 [cit. 2010-02-05]. Dostupný z WWW: http://framework.zend.com/manual/en/zend.acl.html.**
5. **Zend Technologies. Programmers Reference Guide : Zend_Auth [online]. c2006-2010 [cit. 2010-02-05]. Dostupný z WWW: http://framework.zend.com/manual/en/zend.auth.html.**
6. **CASTRO, Elizabeth. HTML, XHTML a CSS : Názorný průvodce tvorbou WWW stránek. [s.l.] : [s.n.], 2007. 440 s. ISBN 978-80-251-1531-2.**
7. **PHP5, MySQL, Apache : Vytváříme webové aplikace. [s.l.] : [s.n.], 2006. 816 s. ISBN 80-251-1073-7.**
8. **GUTMANS, Andi, SAETHER BAKKEN, Stig, RETHANS, Derick. Mistrovství v PHP 5. [s.l.] : [s.n.], 2007. 656 s. ISBN 978-80-251-1519-0.**

Vedoucí bakalářské práce: **Ing. Tomáš Dulík**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **5. března 2010**

Termín odevzdání bakalářské práce: **1. června 2010**

Ve Zlíně dne 5. března 2010

L.S.

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky

prof. Ing. Vladimír Vašek, CSc.
děkan

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

ABSTRAKT

Cílem bakalářské práce je analyzovat stávající systémy správy přístupových práv pro webové aplikace, které umožňují vytvářet a upravovat hierarchii uživatelů a jejich skupin a dynamicky jim pak přidělovat obecná práva. Provést jejich časové a paměťové srovnání a nakonec pro vybraný systém vytvořit vzorovou GUI komponentu pro správu přístupových práv, která bude umět využít všech vlastností daného systému.

Klíčová slova: GUI, ACL, Nette Framework, phpGACL, Zend Framewok, komponenta

ABSTRACT

The aim of this work is to analyze the existing systems of access rights management for Web applications that allow you to create and edit the hierarchy of users and groups and then dynamically allocate them to the general access rights. Carry out their time and memory comparison, and finally selected for the system to create a GUI component model for managing access rights to be able to use all the features of the system.

Keywords: GUI, ACL, Nette Framework, phpGACL, Zend Framework, component

Tímto bych chtěl poděkovat Ing. Tomáši Dulíkovi za odborné vedení, cenné rady, věcné připomínky a pomoc při zpracování bakalářské práce.

„Lidé nechtějí od života žádat mnoho, protože se bojí prohry. Kdo však touží svádět dobrý boj, musí pohlížet na svět jako na nesmírný poklad, který tu čeká, aby byl dobyt.“

Paulo Coelho

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně …….……

podpis diplomanta

OBSAH

ÚVOD

Pomineme-li na internetu webové stránky, které jsou pouze statické a jejich změna se provádí zásahem přímo do zdrojových kódů, tak prakticky na každé stránce je potřeba řešit přístupová práva a to z toho důvodu, že ne každý tyto změny smí provádět.

Je spousta možností, jak tato přístupová práva řešit. Jedním z nejjednodušším a hojně užívaným způsobem je prosté přihlášení. Pokud je uživatel přihlášen, je také oprávněn provádět dané operace.

Dalším hojně užívaným způsobem je rozlišovat oprávnění na základě rolí, jako je např. uživatel, moderátor, administrátor, atd. Každá z těchto rolí má pak přidělen vlastní soubor práv.

K jednotlivým rolím lze pak přidělit i privilegia, která daná role může provádět – přidávat články, mazat komentáře, hlasovat v anketě, atd.

Mezi nejkomplexnější a nejvíce využívané způsoby správy přístupových práv patří trojrozměrná správa přístupových práv, která kombinuje role, zdroje a jejich práva.

Já jsem se pokusil najít 3 nejvíce využívané systémy, které právě využívají trojrozměrnou správu přístupových práv.

Mezi prvními vybranými byl systém phpGACL, jenž je přímo určen pro správu přístupových práv. Další volba padla na Zend Framework, který je jakousi alfou a omegou mezi frameworky, neboť za jejím vývojem stojí tvůrci jádra PHP 4. Jako poslední byl vybrán Nette Framework, který pochází z dílny českého autora a mezi programátory se těší stále větší pozornosti. Navíc Nette Framework vychází z konceptu Zend Framework, a proto mě zajímalo, který z těchto frameworků vyjde v testu lépe.

Po uveřejnění vytvořené GUI komponenty pro správu přístupových práv v Nette Framework na fóru frameworku se komponenta setkala výhradně s kladnými ohlasy a již po pár dnech jsem byl požádán, abych tuto komponentu umístil mezi doplňky Nette Framework.

I. TEORETICKÁ ČÁST

1 PŘÍSTUPOVÁ PRÁVA

Přístupová práva jsou součástí každé pokročilejší aplikace. Rozhoduje o tom, kteří uživatelé mají přístup k jednotlivým zdrojům a které funkce dané aplikace mohou využívat. V tomto ohledu rozlišujeme pojmy „autentizace“ a „autorizace“. [1]

1.1 Autentizace

Autentizací se rozumí přihlašování uživatelů, tedy proces, při kterém se ověřuje, zda je uživatel opravdu tím, za koho se vydává. [2]

1.2 Autorizace

Při autorizaci se ověřuje, zda má uživatel dostatečná oprávnění pro přístup k určitému souboru či k provedení určité akce. Tato kontrola se provádí na základě členství uživatele v různých uživatelských skupinách, přístupových seznamech, apod. Nebo-li se předpokládá předchozí úspěšná autentizace, na jejíž spolehlivosti je autorizace plně závislá. [1]

1.2.1 Nejjednodušší způsob autorizace

Jako nejjednodušší autorizační mechanizmus by se dala použít samotná autentizace – tj. pouhé přihlášení uživatele. Takováto autorizace by se dala využít u jednodušších stránek, kde přihlášený uživatel má veškerá práva – přihlášením se uživatel autorizuje ke všem akcím a operacím.

1.2.2 Řízení přístupu založené na úrovních citlivosti

Každý uživatel je administrátorem zařazen na určitou úroveň citlivosti (důvěryhodnosti). Zdrojům a operacím se také přidělí různé úrovně citlivosti. Při přístupu uživatele k některému objektu se úroveň jeho citlivosti porovnává s úrovní citlivosti daného objektu. Povolení nebo zamítnutí je vyhodnoceno na základě tohoto porovnání. [1]

1.2.3 Autorizace na základě rolí

Často využívaným mechanizmem je autorizace na základě přidělených rolí (nebo též skupin). Každému uživateli hned při přihlášení (popř. přístupu na stránku) přiřadíme jednu či více rolí, ve kterých bude vystupovat. [3] Každá role je stanovena jako nezávislý profil, který má již nastavená patřičná pravidla ke zdrojům a operacím.

Pod pojmem role si můžeme představit například pozice ve firmě. Tj. ředitel, manažer, ekonom, IT technik, atd.

1.2.4 Autorizace na základě zdrojů

Zdroji jsou myšleny logické prvky na webu (v aplikaci). V případě běžné webové stránky tyto zdroje mohou reprezentovat například články, příspěvky, menu, ankety, reklamní bannery atp. Administrátor uživateli nastaví, do kterých zdrojů má přístup povolen a do kterých naopak odepřen.

1.2.5 Autorizace na základě práv

Pod pojmem práva (privilegia) si můžeme v oblasti webů/aplikací představit také operace, které uživatel provádí. Na běžném webu to mohou být práva typu editace, vytváření, mazání, hlasování atp.

1.2.6 Komplexní řešení autorizace

Komplexním a hojně užívaným řešením autorizace je tzv. trojrozměrná správa přístupových práv. V tomto případě jsou jako rozměry myšleny právě role, zdroje a práva.

Každé roli se nastaví zdroj, do kterého může role přistupovat a dané práva, které v tomto zdroji může role vykonávat.

Např. mějme uživatele *Tom*, který je členem role *Generální ředitel*. Tato role má povolený vstup do zdroje *Česká spořitelna* a jako právo má nastavené *Náhled*. Díky stromové struktuře má nyní *Generální ředitel* povolen *Náhled* nad všemi potomky zdroje *Česká spořitelna*.

Dalším příkladem může být role *IT Technik*, která má stejně nastavené právo *Náhled*, jako *Generální ředitel*. K tomuto právu mu ještě přibude právo *Správa*, které bude mít povolené nad zdroji *Pobočka Tř. T. Bati* a *Pobočka Zarámí*. *IT Technik* tedy bude mít právo *Náhled* nad všemi počítači v České spořitelně, ale spravovat (opravovat) je bude moci pouze na pobočkách Tř. T. Bati a Zarámí.

2 VYBRANÉ SYSTÉMY PRO SPRÁVU PŘISTUPOVÝCH PRÁV

Mezi systémy pro správu přístupový práv nesmí bezesporu chybět knihovna phpGACL (PHP Generic Access Control Lists), jejímž autorem je Mike Benoit a která je přímo

určena pro správu přístupových práv. Tuto knihovnu využívá, mimo jiné, např. aplikace dotProject nebo CMS Mambo. [4]

Knihovna phpGACL je určena výhradně pro správu přístupových práv a na nic dalšího se již využít nedá. Pokud bychom chtěli sáhnout po něčem, co by nám nabízelo jak správu přístupových práv, ale také jiné nástroje pro naší aplikaci, tak bychom sáhli nejspíš po nějakém frameworku.

Jedním z nejznámějších je právě Zend Framework. Firma Zend Technologiest Ltd., která tento fremework vyvíjí, stojí také za jádrem PHP od verze 4. [5]

Jako poslední systém, který nabízí správu přístupových práv, byl vybrán Nette Framawrok, který pochází z české dílny a jejím hlavním autorem je David Grudl.

2.1 phpGACL

2.1.1 Princip funkce

Knihovna je rozdělena na čtyři základní objekty – jsou jimi Access Control Objects (ACO), Access Request Objects (ARO), Access eXtension Objects (AXO) a Access Control Lists (ACL).

ACO jsou věci, které chceme kontrolovat. Např. webové články, ankety, příspěvky, atd. Tento objekt si můžeme představit jako soubor zdrojů.

ARO jsou věci, které žádají přístup. Např. lidé, uživatelé, vzdálený počítač, atd. Objekt ARO tedy reprezentuje role.

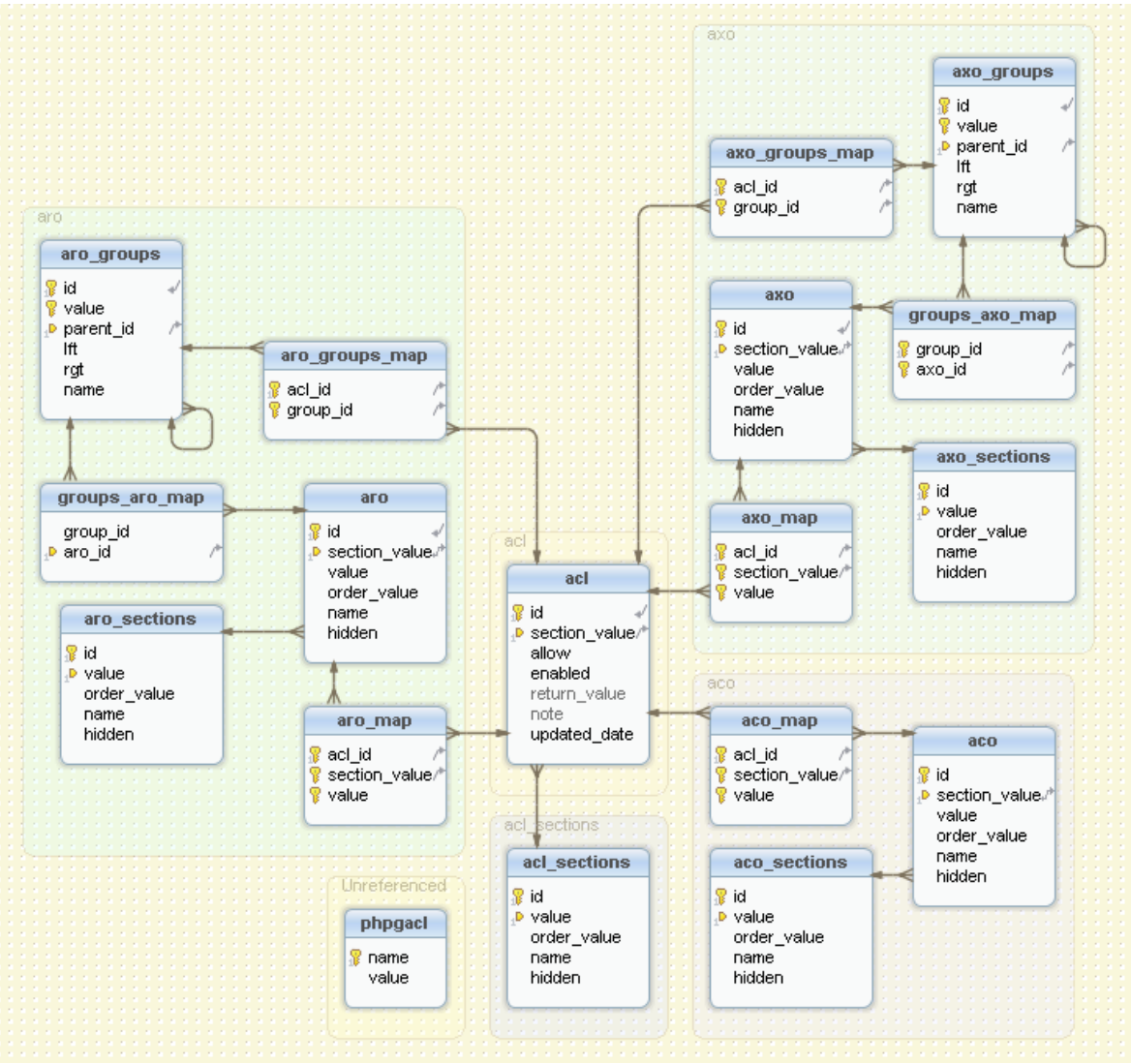
Objekt AXO může (není podmínkou) přidat třetí rozměr k oprávnění a reprezentuje akce, které může objekt ARO vykonávat nad objekty ACO.

Objekt ACL pak slouží k provázání výše zmíněných objektů. Obsahuje tedy veškeré pravidla přístupů. [6]

Knihovna phpGACL využívá celkem 18 tabulek.

- *acl* – obsahuje veškerá pravidla oprávnění a propojení jednotlivých objektů (AR, AXO, ACO)
- *acl_sections* – rozdělení pravidel na jednotlivé sekce
- *aco* – obsahuje všechny objekty ACO

- *aco_map* – spojovací tabulka propojující pravidla v tabulce *acl* s objekty ACO (realizace vazby M:N)
- *aco_sections* – sekce objektu ACO
- *aro* – obsahuje všechny objekty ARO
- *aro_groups* – skupiny objektů ARO. Objekty mohou zaujímat stromovou strukturu.
- *aro_groups_map* – spojovací tabulka propojující pravidla v tabulce *acl* se skupinami objektů ARO (realizace vazby M:N)
- *aro_map* – spojovací tabulka propojující pravidla v tabulce *acl* s objekty ARO (realizace vazby M:N)
- *aro_sections* – sekce objektu ARO
- *axo* – obsahuje všechny objekty AXO
- *axo_groups* – skupiny objektů AXO. Objekty mohou zaujímat stromovou strukturu.
- *axo_groups_map* – spojovací tabulka propojující pravidla v tabulce *acl* se skupinami objektů AXO (realizace vazby M:N)
- *axo_map* – spojovací tabulka propojující pravidla v tabulce *acl* s objekty AXO (realizace vazby M:N)
- *axo_sections* – sekce objektu AXO
- *groups_aro_map* – spojovací tabulka propojující objekty ARO s jejich příslušnými skupinami (realizace vazby M:N)
- *groups_axo_map* – spojovací tabulka propojující objekty AXO s jejich příslušnými skupinami (realizace vazby M:N)
- *phpgacl* – obsahuje informace o verzi schématu a knihovny



Obr. 1: Databázové schéma knihovny phpGACL

Objekty ARO mohou mít stromovou hierarchii a stejně tak objekty AXO. Objekty ACO již tuto podporu nemají.

Součástí knihovny jsou 2 hlavní třídy. Třída `Gacl`, která by měla být použita jen v aplikacích, kde se testuje oprávnění přístupu a všechna důležitá nastavení. V tomto případě jsou pravidla uložena v databázi. Třída `Gacl_Api` by měla být použita v aplikacích, kde jsou veškeré objekty a pravidla definována přímo ve scriptu.

K nejdůležitějším metodám pro práci s knihovnou patří metody `add_object_section`, `add_object`, `add_group`, `add_group_object`, `add_acl` a `acl_check`.

Metoda `add_object_section` vytváří jednotlivé sekce a `add_object` už konkrétní objekty.

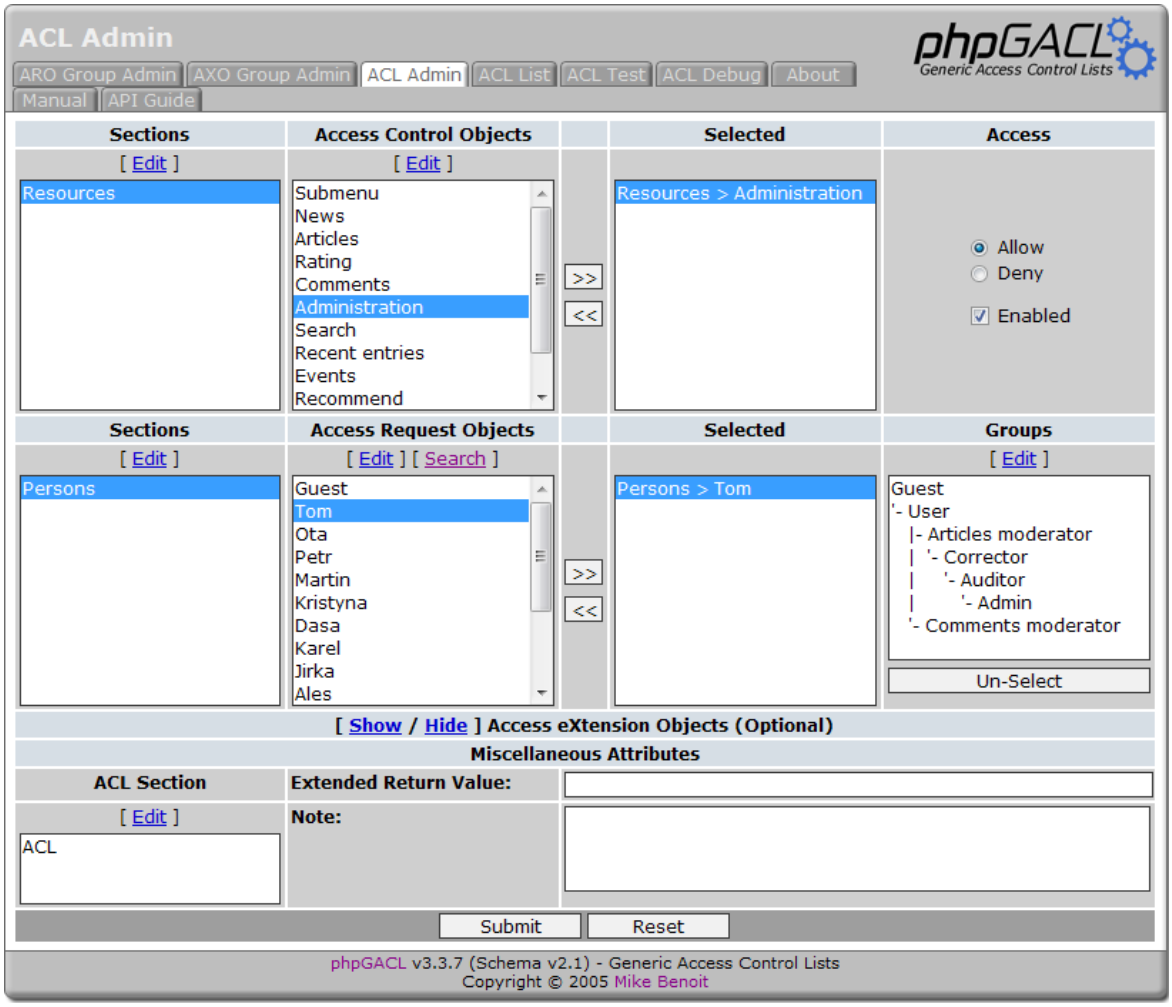
K vytváření skupin slouží metoda `add_group`. Následné přiřazení objektu do skupiny se provádí metodou `add_group_object`.

Pravidla přístupů a provázání jednotlivých objektů se děje za pomoci metody `add_acl`.

K ověření přístupu slouží metoda `acl_check`, která se nachází v obou třídách. Metodě jsou předány 4 povinné parametry a 4 nepovinné parametry. Jsou jimi název sekce objektů ACO, název objektu ACO, název sekce objektu ARO, název objektu ARO, název sekce objektu AXO, název objektu AXO, ID skupiny objektu ARO a ID skupiny objektu AXO. [7]

2.1.2 Základní administrační rozhraní

Spolu s knihovnou phpGACL je také distribuováno administrační rozhraní, které nabízí správu všech prvků knihovny phpGACL. Tvůrcem této administrace je autor knihovny Mike Benoit. [8]



Obr. 2: Vzhled administračního rozhraní knihovny phpGACL

I přes komfort, kterým je možnost nastavit celou knihovnu přes grafické rozhraní, čímž odpadá nutnost přímého zásahu do databáze, je grafické rozhraní velmi složité na ovládání a někdy i na pochopení.

2.1.3 Výhody systému

Jedna instalace knihovny se dá použít pro více aplikací na jednom serveru a to díky rozdělení objektů do jednotlivých sekcí (v tomto případě aplikací).

Komplexní knihovna nabízející spoustu uplatnění v rámci přístupových práv.

2.1.4 Nevýhody systému

Složité ovládání grafického rozhraní, které výrazně snižuje jeho použitelnost.

Metoda `acl_check`, která se využívá k testování přístupu, má celkem 4 povinné parametry, které nejsou ani kontrolovány, jestli opravdu existují. V případě překlepu to administrátor nemusí hned tak zjistit.

2.2 Zend Framework

Zend Framework obsahuje jak objekt obstarávající autentizaci (`Zend_Auth`), tak i objekt pro správu přístupových práv (`Zend_Acl`).

2.2.1 Princip funkce

Správa přístupových práv funguje v Zend Frameworku na principu rolí, zdrojů a práv.

Všechny role musí implementovat rozhraní `Zend_Acl_Role_Interface` a samotná role se pak vytváří pomocí objektu `Zend_Acl_Role`. Toto rozhraní se skládá pouze z jediné metody, kterou je metoda `getRoleId`. [9]

Vytváření zdrojů je velmi podobné jako vytváření rolí. Zase je nutná implementace rozhraní `Zend_Acl_Resource_Interface`, která obsahuje metodu `getResourceId`. Zdroje i role podporují hierarchickou strukturu. Objekt Zend_Acl také podporuje práva na zdrojích. [9]

Veškeré operace s přístupovými právy se provádí pomocí metod objektu `Zend_Acl`. Role se registrují pomocí metody `addRole` a zdroje analogicky pomocí `addResource`. Prvním parametrem je název a jako druhým nepovinným parametrem je název rodiče.

Jakmile máme vytvořeny všechny role a zdroje, tak už zbývá jen registrace pravidel mezi nimi. K tomu slouží metoda `allow` v případě povolení přístupu a metoda `deny` k odepření přístupu. Těmto metodám předáváme jako první parametr název role, jako druhý název zdroje a jako třetí název práva. Práva není nutné nijak registrovat. Místo prvního a druhého parametru je možné uvést hodnotu `null`, která říká, že pravidlo se má použít na všechny role nebo zdroje.

K ověření přístupu slouží metoda `isAllowed`, která má jako první parametr název role, jako druhý název zdroje a třetím parametrem je název práva.

2.2.2 Výhody

Metoda `isAllowed` kontroluje existenci vstupních parametrů.

Perfektně zpracovaná dokumentace.

Spousta možností využití.

Jednoduché vytváření rolí, zdrojů a následné vytváření pravidel mezi nimi.

2.2.3 Nevýhody

Nutnost uvádění názvu role v metodě `isAllowed`, čímž se velmi snižuje použitelnost tohoto systému ve větších aplikací. Při implementaci do stránky se musí vždy myslet na to, které všechny role budou moci k danému zdroji přistupovat. Např. pokud k danému zdroji mohou přistupovat celkem tři role, tak ve zdrojovém kódu musím metodu `isAllowed` uvést hned třikrát lišící se pouze v názvu role.

Při kešování je nutné objekt `Zend_Acl` kešovat pro každou roli zvlášť, protože chybí jednak podpora rolí v objektu `Zend_Auth` a pak také chybí jakákoliv provázanost mezi objektem `Zend_Acl` a `Zend_Auth`.

Veškeré případné navázání na databázi je nutné vyřešit dodatečně.

Absence výchozí role pro nepřihlášeného uživatele.

2.3 Nette Framework

2.3.1 Princip funkce

K řízení přístupových práv v Nette Framework slouží autorizační handler, který implementuje rozhraní `IAuthorizator`. Toto rozhraní má jedinou metodu a tou je metoda `isAllowed` s úkolem rozhodnout, zda má daná role povolení provést určitou operaci s určitým zdrojem. [3]

V tuto chvíli záleží čistě na programátorovi, jak implementuje autorizační handler. Nicméně Nette Framework disponuje i jednou, poměrně silnou předpřipravenou implementací a tou je třída `Permission` implementující model Access control List. Práce s ní spočívá v definici rolí, zdrojů a jednotlivých oprávnění, přičemž role a zdroje umožňují vytvářet stromové hierarchie. [3]

Samotný kód třídy `Permission` z velké části vychází z objektu `Zend_Acl`. Na rozdíl od `Zend_Acl` třída `Permission` používá pro zdroje a role textové identifikátory (namísto objektů `Zend_Acl_Role` nebo `Zend_Acl_Resource`), což je určitě krok správným směrem. [10]

Třída `Permission` má hlavní metody `addRole`, `addResource`, `allow` a `deny`, které jsou funkčností shodné s objektem `Zend_Acl`.

Velkou výhodou je uvedení pouze dvou parametrů u metody `isAllowed`, přičemž role se sama doplní pomocí metody `getRoles`. V cyklu se projdou všechny role a při prvním kladném vyhodnocení se kladně vyhodnotí i celá metoda. [10]

2.3.2 Výhody

Metoda `isAllowed` kontroluje existenci vstupních parametrů.

Automatické provázání s autentizačním handlerem – objekt `Permission` automaticky pracuje s rolemi, které jsou nastavené při autentizaci.

Při kešování stačí kešovat pouze objekt `Permission` bez jakýchkoliv výjimek.

Pro nepřihlášeného uživatele je výchozí rolí `guest`.

Spousta možností využití.

Velmi jednoduché vytváření rolí, zdrojů (využití textových identifikátorů) a následné vytváření pravidel mezi nimi.

2.3.3 Nevýhody

Veškeré případné navázání na databázi je nutné vyřešit dodatečně.

Dokumentace je vyvíjena spolu s vývojem frameworku, takže se může stát, že člověk při procházení dokumentace narazí na část, která platila u předchozí verze frameworku.

3 DATABÁZOVÁ VRSTVA

Pro přesnější porovnání systému Zend Framework a Nette Framework byla vybrána jednotná databázová vrstva. Případné rozdíly ve výsledcích těchto dvou systémů jsou zapříčiněny tedy pouze jejich implementací rozhodovací logiky modelu ACL.

V systému phpGACL byla použita jeho výchozí databázová vrstva, kterou je ADODB Library for PHP4. Přepsaní celého systému do jiné databázové vrstvy by bylo nejen velmi pracné, ale zejména také kontraproduktivní.

3.1 Dibi

Databázová vrstva dibi v aktuální verzi 1.2. podporuje mnoho ze současných významných databází. Mezi tyto databáze patří: MySQL, PostgreSQL, SQLite, MS SQL, Oracle, Access a generické PDO a ODBC. [11]

II. PRAKTICKÁ ČÁST

4 IMPLEMENTACE VYBRANÝCH SYSTÉMŮ

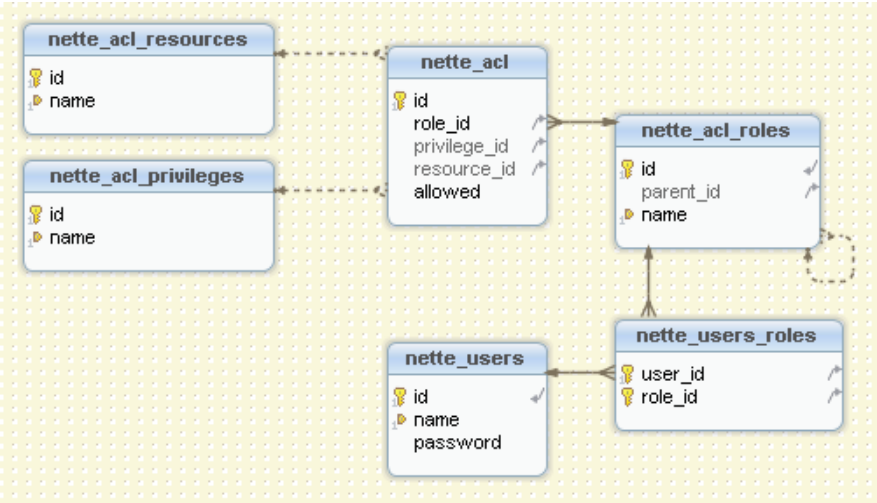
Vybrané systémy bylo nejprve nutné implementovat do testovací stránky, aby bylo možné využívat vlastností daného systému. Jako další následovalo vytvoření objektu, který bude zajišťovat autentizaci návštěvníka. U systémů Zend Framework a Nette Framework bylo ještě nutné vytvořit objekt, který zajistí naplnění objektu ACL patřičnými daty.

Na začátku testovací stránky se vždy nachází kód, který zjistí velikost aktuálně využité paměti, zapnutí bufferování výstupu a inicializace session.

Na konci stránky bufferování ukončíme, odešleme a následně zalogujeme hodnotu využité paměti.

Pro Zend Framework a Nette Framework bylo nutné vytvořit databázi, do které se uloží potřebná data. Protože Nette Framwork vychází z konceptu Zend Framework, byla pro oba systémy vytvořena pouze jedna databáze. Schéma obsahuje celkem 6 tabulek.

- *nette_acl* – obsahuje pravidla určující oprávnění. Propojuje tabulky obsahující role, zdroje a práva.
- *nette_acl_privileges* – soubor všech práv
- *nette_acl_resources* – soubor všech zdrojů
- *nette_acl_roles* – definice rolí. Role mohou zaujímat stromovou strukturu.
- *nette_users* – tabulka obsahující všechny uživatele
- *nette_users_roles* – spojovací tabulka propojující uživatele a role (realizace vazby M:N)



Obr. 3: Databázové schéma pro Zend Framework a Nette Framework

Systém phpGACL využívá svou výchozí databázovou strukturu (Obr. 1).

Vytvořené databázové schéma pro Zend Framework a Nette Framework obsahuje jen ty nejzákladnější tabulky, proto jich v porovnání s phpGACL (18 tabulek) obsahuje pouze 6.

Knihovna phpGACL, na rozdíl od mnou vytvořeného databázového schématu, obsahuje tabulky pro rozdělení jednotlivých objektů (ACL, ACO, ARO a AXO) do jednotlivých sekcí. Dále umožňuje objekty ARO a AXO rozčlenit do jednotlivých skupin. Jako poslední vylepšení, oproti mému schématu, je možnost navázání jednoho pravidla z tabulky *acl* na více objektů ARO a AXO.

Všechna tato rozšíření dělají z knihovny phpGACL jeden z nejvíce flexibilních systémů pro správu přístupových práv.

4.1 phpGACL

4.1.1 Implementace do testovací stránky

Nejprve je nutné vytvořit objekt *Gacl*, díky kterému budeme moci přistupovat k metodě *acl_chcek*, která slouží ke kontrole oprávnění.

Také si vytvoříme objekt *Login*, který nám bude zajišťovat veškeré potřebné akce spojené s uživatelem.

4.1.2 Objekt Login

Objekt *Login*, obstarávající akce spojené s uživatelem, má celkem 3 metody:

- *authenticate($username, $password)* – zajišťuje autentizaci uživatele
- *isAuthenticated()* - vrací *true*, pokud je uživatel přihlášen
- *signOut()* - provádí odhlášení uživatele

K uchovávání informací o uživateli využíváme sessions. Pokud uživatel není přihlášený, tak mu ručně nastavíme výchozí roli *guest*.

4.2 Zend Framework

4.2.1 Implementace do testovací stránky

Jako první získáme instanci uživatele: *$user = Zend_Auth::getInstance()*. Nyní nám objekt *$user* nabízí tyto metody:

- *getIdentity()* - získá identitu uživatele, pokud je dostupná ve skladišti. Jinak vrací *null*
- *hasIdentity()* - vrací *true*, pokud je identita ze skladiště dostupná
- *clearIdentity()* - vymaže identitu z trvalého úložiště (odhlášení)
- *authenticate($authAdapter)* – ověření autentizace. Jako parametr je předán objekt implementující rozhraní *Zend_Auth_Adapter_Interface*

Objekt *$user* vytváříme jako první, protože nejdřív je žádoucí zjistit role, do kterých uživatel patří a podle toho sestavit objekt *Acl*.

Pokud uživatel není přihlášený (*!$user->hasIdentity()*), musíme ho ručně přiřadit do výchozí role *guest*.

Jako poslední provedeme vytvoření objektu *Acl*, kterému jako parametr předáme pole rolí aktuálního návštěvníka. Objekt má metodu *isAllowed*, která slouží k ověření oprávnění.

4.2.2 Objekt Login

Implementuje rozhraní *Zend_Auth_Adapter_Interface* a slouží k autentizaci uživatele. Má jen jednu metodu a tou je metoda *authenticate()*.

Tato metoda vrací objekt *Zend_Auth_Result*, který má 3 parametry: kód určující výsledek autentizace, identitu a pole obsahující textové zprávy.

V případě úspěšné autentizace je návratový kód *Zend_Auth_Result::SUCCESS* a při neúspěšné autentizaci je jim *Zend_Auth_Result::FAILURE*.

4.2.3 Objekt Acl

Slouží k naplnění objektu *Zend_Acl* na základě předaných rolí, které jsou předány objektu při jeho vytvoření.

4.3 Nette Framework

4.3.1 Implementace do testovací stránky

V Nette Framework nemusíme mít jeden objekt na akce spojené s uživatelem a druhý objekt pro práci s přístupovými právy. Na to všechno nám stačí pouze jeden objekt, který vytvoříme tímto způsobem: *$user = Environment::getUser()*.

O přihlášení uživatele se stará autorizační handler. Tento autorizační handler je nejprve nutné zaregistrovat *$user->setAuthenticationHandler(new Login)*. Jako parametr je mu předán objekt, který vykoná samotné přihlášení.

V tomto stavu objekt $user obsahuje, mimo jiné, tyto metody:

- *isAuthenticated()* - test na autentizaci uživatele
- *getIdentity()* - vrací aktuální identitu uživatele, pokud existuje
- *signOut()* - odhlásí uživatele z aktuální session
- *authenticate($username, $password)* – provádí autentizaci uživatele
- *isAllowed($resource, $privilege)* – slouží k ověření oprávnění

Samotná metoda *isAllowed* by nefungovala správně, kdybychom nezaregistrovali autorizační handler *$user->setAuthorizationHandler(new Acl())*, kterému předáme objekt, který třídu *Permission* naplní patřičnými daty.

4.3.2 Objekt Login

Má jedinou metodu *authenticate*, jejíž úkolem je provést autentizaci uživatele. Jako parametry jsou ji předány přihlašovací jméno a heslo.

Pokud přihlášení není úspěšné, je vyhozena výjimka *AuthenticationException*. V případě úspěšného přihlášení metoda vrací objekt *Identity*, který má jako první parametr jméno uživatele, druhým parametrem je pole rolí a jako poslední jsou jakákoliv data spojená s uživatelem – tj. např. přezdívka, e-mail, věk, atd.

4.3.3 Objekt Acl

Objekt *Acl* má za úkol naplnit objekt *Permission*, který rozšiřuje, patřičnými daty.

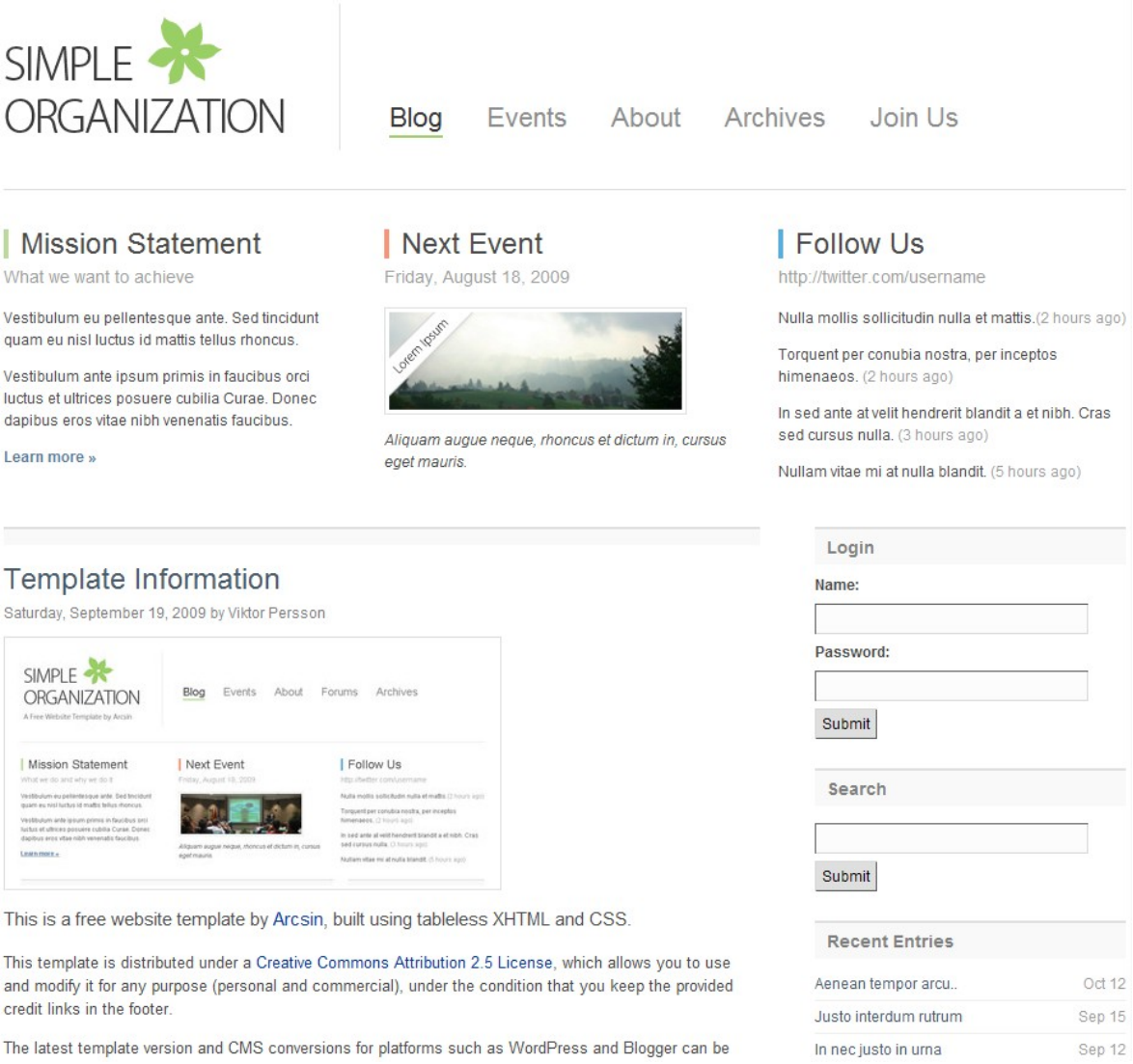
Na rozdíl od Zend Framework není nutné nijak zohledňovat členství v jakékoliv z rolí. Objekt *Permission* se naplní veškerými daty a rozhodovací logika uvnitř frameworku si již sama „sebere“ patřičná data.

5 SROVNÁNÍ VYBRANÝCH SYSTÉMŮ

Srovnání vybraných systému se provádělo simulací přístupů uživatelů na testovací stránku, kde se střídavě vždy přihlásili a odhlásili.

5.1 Testovací stránka

Jako testovací stránka byla vybrána šablona Simple Organization, jejímž autorem je designér a kodér Viktor Persson. [12]



Obr. 4: Testovací stránka

5.1.1 Rozdělení stránky

Výsledná stránka, na které bylo prováděno testování, vznikla spojením dílčích stránek distribuovaných spolu se šablonou a to tak, aby se co nejvíce přiblížila obsahu

nejběžnějších stránek internetu.

Stránka byla rozdělena na pomyslných 12 zdrojů. Jsou jimi: *Submenu*, *News*, *Articles*, *Rating*, *Comments*, *Administration*, *Search*, *Recent entries*, *Events*, *Recommend*, *Board of members* a *Topics*.

5.1.2 Role

Aby se stránka přiblížila nějakému redakčnímu systému, na který se trojrozměrná správa přístupových práv asi nejvíc využívá, bylo vytvořeno celkem 7 rolí s různě nastavenými právy.

Role samotné byly umístěny do stromové hierarchie, aby nebylo nutné nastavovat každé roli práva zvlášť, ale mohlo se využít dědění.

Tab. 1: Hierarchie rolí

Guest
L User
L Comments moderator
L Articles moderator
L Corrector
L Auditor
L Admin

Pokud uživatel není přihlášen, je automaticky zařazen do výchozí role *guest*.

Každé roli bylo nastaveno unikátní oprávnění s přístupem do daných zdrojů s určitými právy.

Tab. 2: Přehled nastavených oprávnění u jednotlivých rolí

Zdroje	Práva	Role						
		Guest	User	Comments moderator	Articles moderator	Corrector	Auditor	Admin
Submenu	Show							
	Edit							
News	Edit							
	Delete							
Articles	Edit							
	Delete							
Rating	Rate							
Comments	Add							
	Edit							
	Delete							
Administration	Show							
Search	Show							
Recent entries	Edit							
Events	Add							
Recommend	Edit							
Board of Members	Edit							
Topics	Edit							

Červené pole značí situaci, kdy role nemá povolený přístup do daného zdroje s konkrétním právem a zelené pole značí analogicky přístup povolen.

5.1.3 Uživatelé

Vytvořené role využívá celkem 15 uživatelů, z nichž každý je členem pouze jedné role.

Tab. 3: Přehled všech uživatelů a jejich nastavených rolí

	Uživatel	Role
1	Tom	Admin
2	Ota	Auditor
3	Petr	Auditor
4	Martin	Corrector
5	Kristyna	Articles moderator
6	Dasa	Articles moderator
7	Karel	Comments moderator
8	Jirka	Comments moderator
9	Ales	User
10	Roman	User
11	Jakub	User
12	Ivo	User
13	Honza	User
14	Marek	User
15	Jana	User

5.2 JMeter

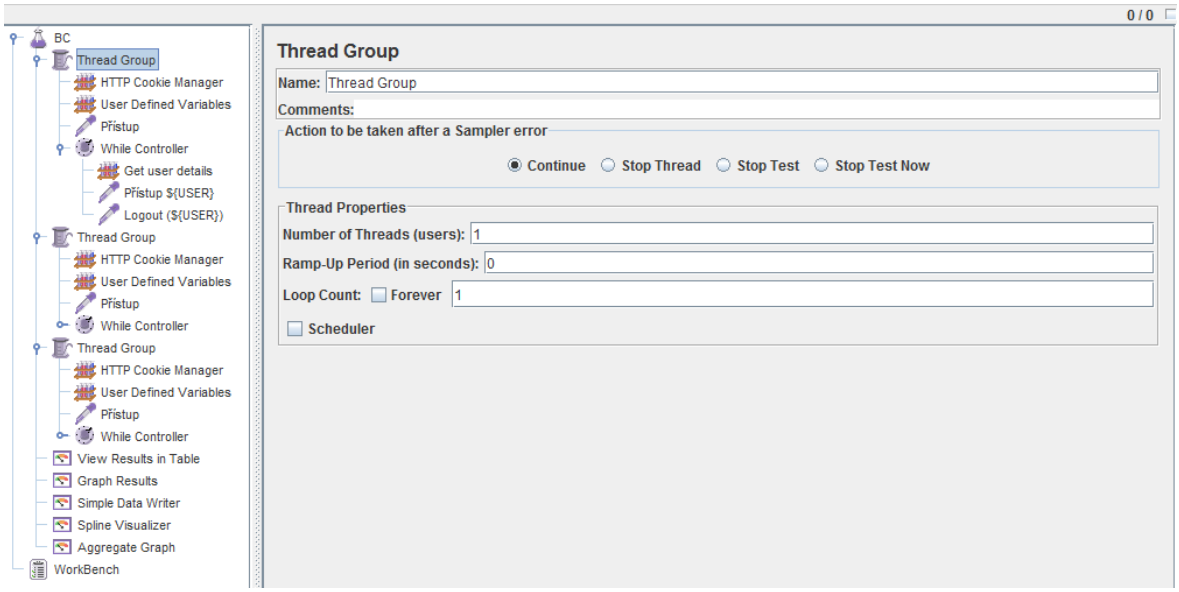
Důležitá vlastnost každé aplikace je její výkon. Ten ovlivňuje několik faktorů, které se nedají jednoduše odhadnout, ale musí se změřit. JMeter pomáhá tento výkon měřit, a to nejen u webových aplikací (HTTP/HTTPS), ale také u FTP serverů a relačních databází. [13]

Pomocí JMeteru se pro vaši aplikaci vytvoří specifický test, který simuluje reálnou zátěž během provozu. Můžete tak specifikovat, kolik klientů současně bude posílat požadavky, na jaké soubory je bude posílat a s jakými parametry. [13]

Open Source JMeter je součást projektu Jakarta od firmy Apache. Je napsán kompletně v jazyce Java a nabízí grafické rozhraní v podobě swing komponent, ale využít můžete také non-GUI módu, kdy není na počítači, na kterém se testuje, nutnost žádného grafického prostředí. Další mód je pro tzv. Remote testing a funguje tak, že na testovaném serveru se spustí pouze serverová část, které odesílá požadavky a výsledky posílá klientovi na jiném stroji. To umožňuje obejít místa mezi klientem a serverem, která zpomalují test. [13]

5.2.1 Nastavení

K nastavení testu se používá strom uzlů a elementů na levé straně okna. Každý uzel pak specifikuje jistá nastavení pro test. Nejvyšší uzel je Test Plan (plán testu), kde není nutné nic nastavovat, ale je možné zde definovat uživatelské proměnné. Důležitějším elementem testu je Thread Group (skupina procesů). V něm určujeme množství virtuálních uživatelů (procesů), kteří server navštíví, čas jejich "příchodu" a počet opakování. [14]



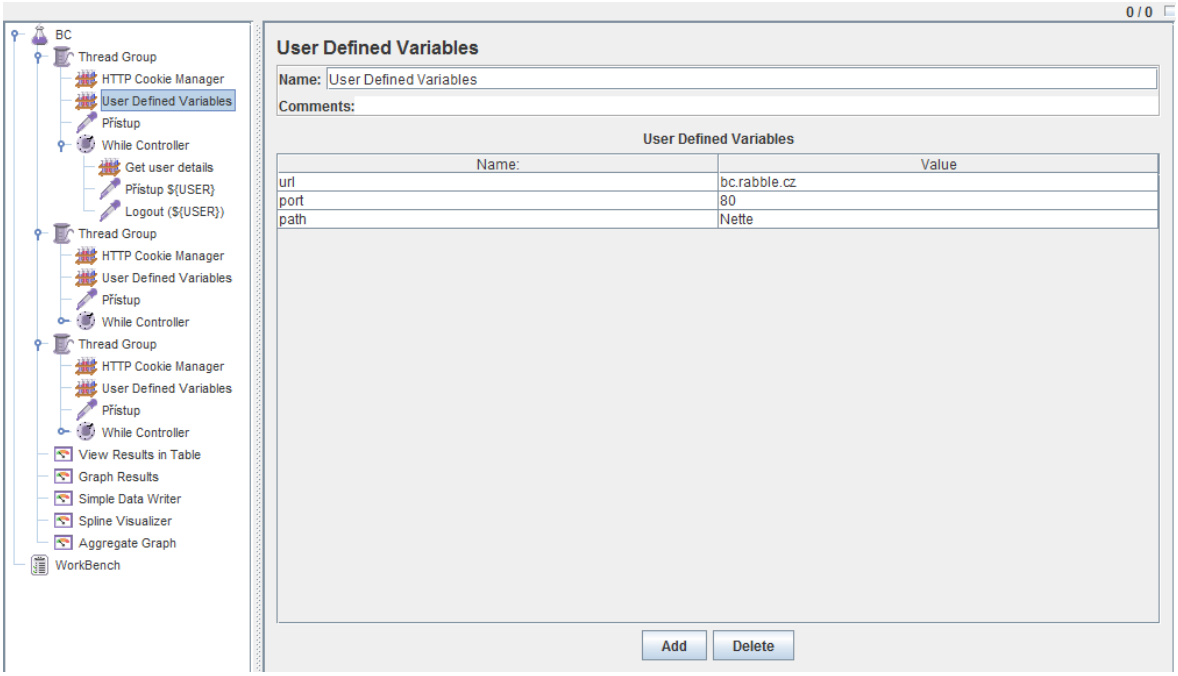
Obr. 5: Nastavení Thread Group v programu JMeter

V každém Thread Group jsou 2 Config Elementy. Jedním z nich je http Cookie Manager, který nedělá nic jiného, než že spravuje cookie a druhým je User Defined Variables, kde jsou nastavené proměnné pro daný Thread Group.

V tomto Config Elementu jsou pouze 3 proměnné:

- adresa serveru, na kterém se nachází testovací stránka
- port, na kterém stránky běží
- cesta ke konkrétnímu systému

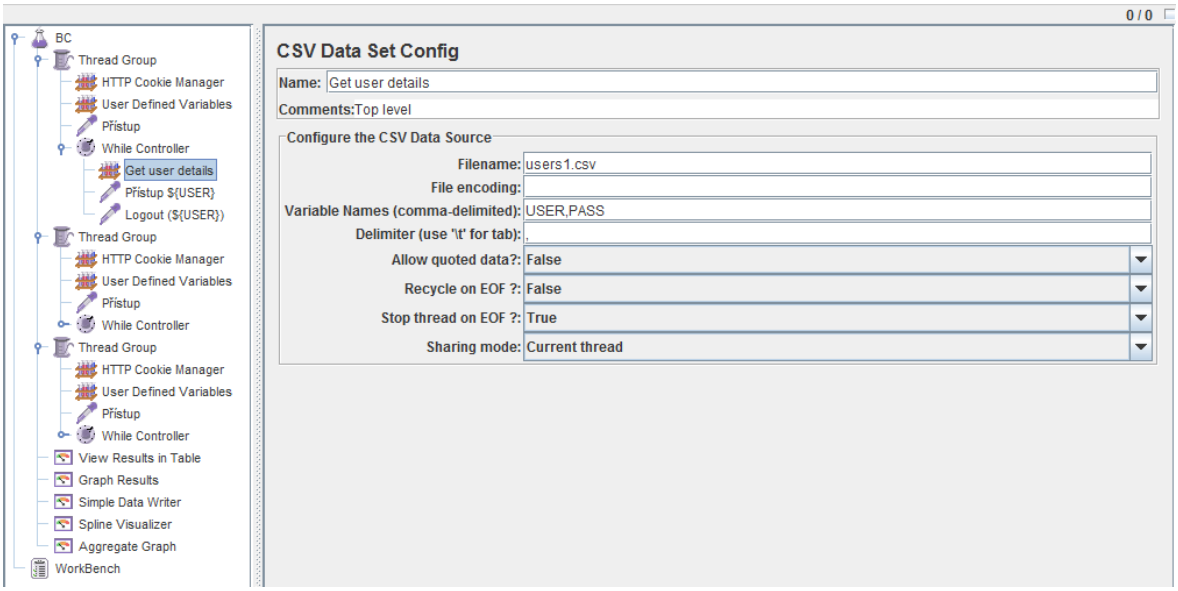
Adresa, na kterou se pak přistupuje, vypadá takto: *http://${url}/${path}/index.php*



Obr. 6: Nastavení proměnných pro konkrétní Thread Group

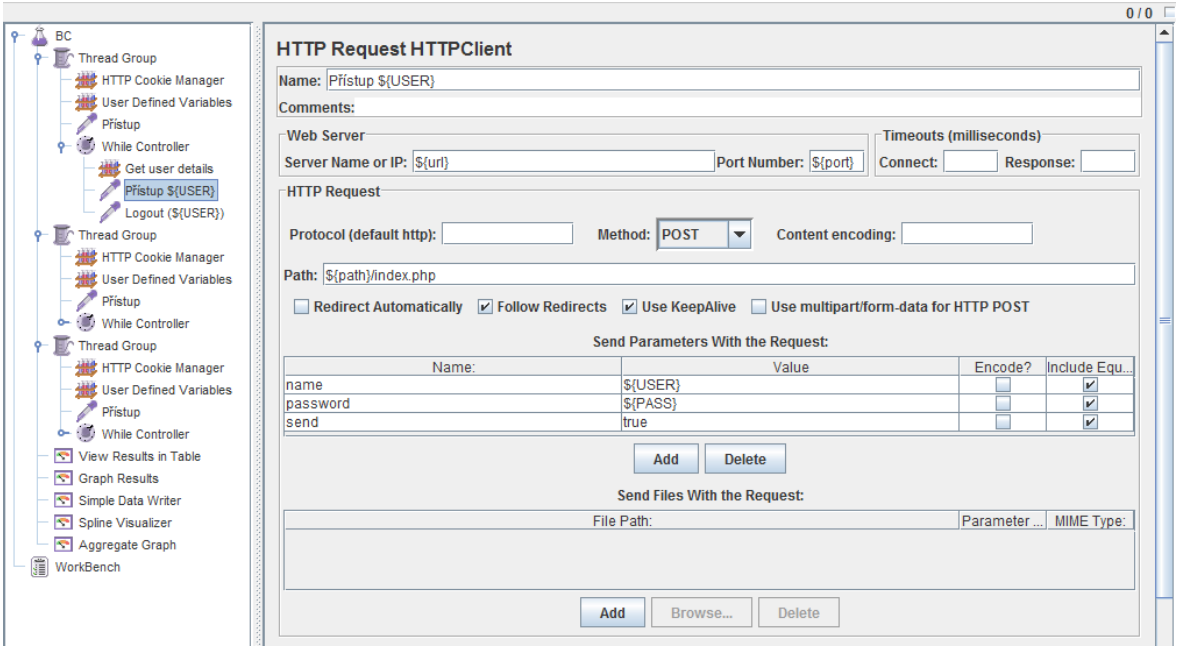
Přímým potomkem Thread Group je dále jeden http Request HTTPClient, který simuluje prvotní přístup nepřihlášeného návštěvníka a While Controller, který simuluje přihlášení a odhlášení všech 15 uživatelů.

Přihlašovací jména a hesla všech uživatelů jsou uložena ve 3 CSV souborech, ze kterých je čte a následně ukládá do proměnných *${USER}* a *${PASS}* Config Element zvaný CSV Data Set Config, který je součástí While Controlleru.



Obr. 7: Nastavení Config Elementu CSV Data Set Config

Součástí While Controlleru je pak už jen 2x Sampler HTTP Request HTTPClient, z nichž první simuluje přístup uživatele a druhý jeho následné odhlášení.



Obr. 8: Nastavení Sampleru HTTP Request HTTPClient simulující přihlášení uživatele

5.2.2 Testování vybraných systémů

Náš Test Plan se skládá celkem ze 3 Thread Group, z nich každý bude simulovat přístup všech 15 uživatelů na stránku. Tyto 3 Thread Groupy se spustí ve stejnou chvíli a poběží paralelně. Na testovací stránce se tedy provede celkem 45 přihlášení a 45 odhlášení pro každý vybraný systém.

Vybrané systémy jsou testovány zvlášť se zapnutým a vypnutým kešováním.

5.3 Logování událostí

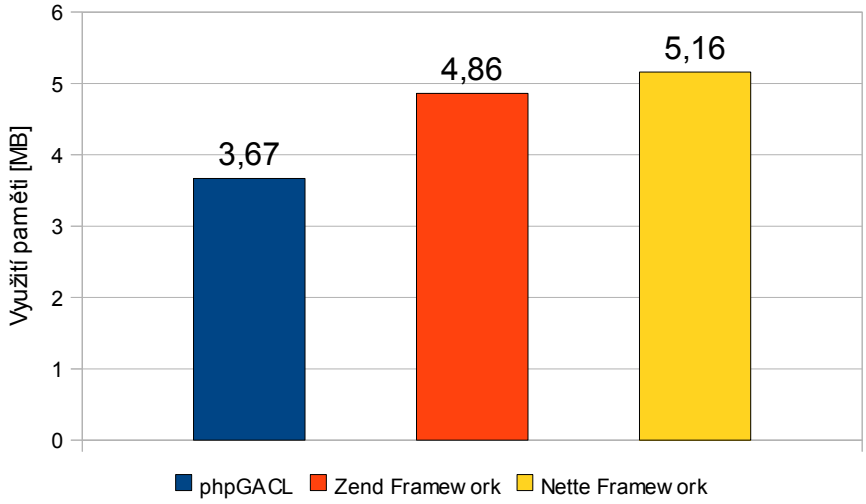
Součástí testovací stránky je jednoduchá PHP funkce, která provádí logování všech akcí, které se na stránce vykonávají. Logovanými akcemi je odhlášení uživatele, přihlášení uživatele a přístup uživatele (děje se při každém načtení stránky). Součástí logování je také informace o využití paměti.

Toto logování bylo během vytváření testu a samotném průběhu testu velmi nápomocné, neboť díky němu bylo možné ověřit všechny potřebné akce a odladit tak celé nastavení testu.

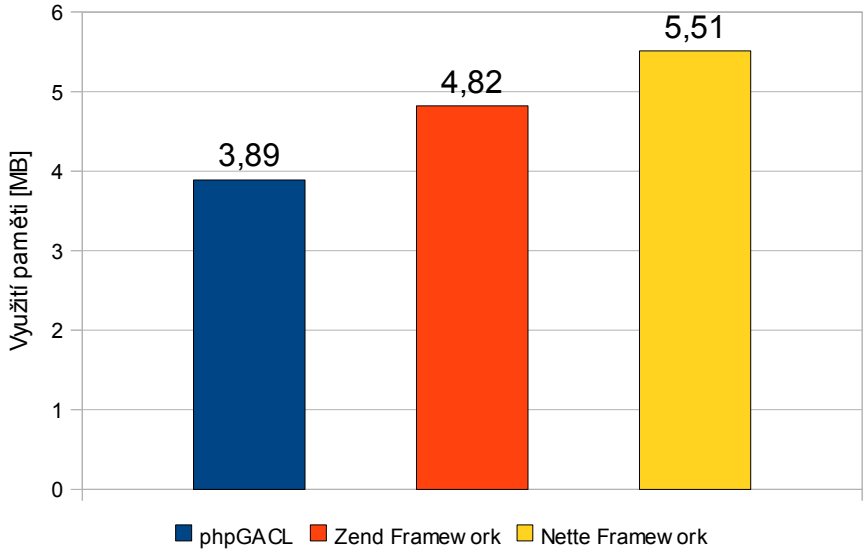
5.4 Celkové srovnání

Data, z nichž se sestavovaly výsledky testů, jsou kombinací výstupu z programu JMeter a zalogovaných dat. Hodnoty uváděné v grafech jsou průměrem naměřených hodnot.

Jako první porovnáme paměťovou náročnost vybraných systémů.

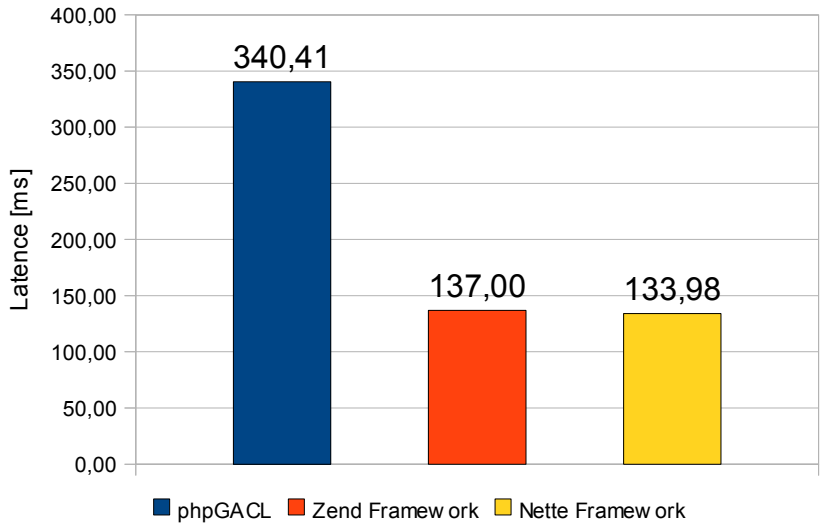


Graf 1: Srovnání paměťové náročnosti bez využití cache

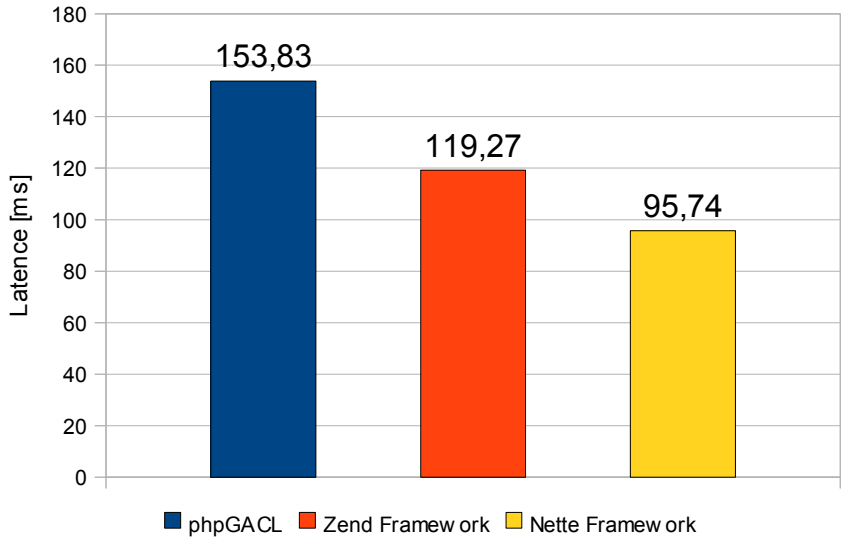


Graf 2: Srovnání paměťové náročnosti s využitím cache

V případě časové náročnosti byla porovnávána latence, což je zpoždění, které vznikne než dorazí data z bodu A do bodu B. [15]



Graf 3: Časová náročnost bez využití cache



Graf 4: Časová náročnost s využitím cache

5.5 Vyhodnocení testování

Z grafů je patrné, jak který systém obstál.

Nejnižší paměťové nároky měl systém phpGACL a to 3,67 MB bez využití cache a 3,89 MB s využitím cache. Využití cache vedlo u systému phpGACL tedy ke zhoršení. Stejně tak na tom byl Nette Framework. U jediného Zend Framework došlo ke zlepšení, a to hodnotou 0,04 MB, která je prakticky zanedbatelná.

Tab. 4: Přehled paměťové náročnosti vybraných systémů

	Paměťová náročnost		
	phpGACL	Zend Framework	Nette Framework
Bez cache	3,67	4,86	5,16
S cache	3,89	4,82	5,51

V oblasti časové náročnosti byly rozdíly přece jen markantnější. Nejhorší a tedy nejpomalejší odezvu měl systém phpGACL s hodnotou 340,41 ms, která je cca 2,5x pomalejší než to bylo u zbylých dvou systémů. Nejlépe dopadl Nette Framework s odezvou 133,98 ms. V těsném závěru je Zend Framework s odezvou 137 ms a rozdílem pouhých 3,02 ms.

Použití cache s pořadím nijak nezamíchalo. Nejrychlejší odezvu měl Nette Framework s hodnotou 95,74 ms (zlepšení 38,24 ms). Zend Framework si polepšil o 17,73 ms na průměrnou odezvu 119,27 ms. Největší zlepšení měl systém phpGACL, který si polepšil o celých 186,58 ms na konečnou odezvu 153,83 ms.

Tab. 5: Přehled časové náročnosti vybraných systémů

	Časová náročnost		
	phpGACL	Zend Framework	Nette Framework
Bez cache	340,41	137,00	133,98
S cache	153,83	119,27	95,74

6 NOVÁ KOMPONENTA "GUI FOR ACL" PRO NETTE FRAMEWORK

K vytvoření GUI komponenty pro správu přístupových práv jsem si vybral Nette Framework a to hned z několika důvodů.

Tím prvním důvodem byl asi fakt, že jsem s ním již nějakou dobu pracoval, takže jsem v něm jednak uměl, rozuměl mu a měl k němu jistý „vztah“. Tento vztah jsem si vybudoval hlavně díky faktu, že celý framework pochází z dílny českého autora.

Dalším důležitým faktem je stále se rozrůstající komunita, která se stará u rozsáhlou dokumentaci. Na fóru frameworku se vám ze strany komunity dostane vždy pomoci.

Při práci člověk jistě ocení velké množství doplňků, které jsou v frameworku k dispozici.

V porovnání s Zend Framework můžeme jistě mluvit o jednodušší a intuitivnější syntaxi.

Jako poslední důvod, proč jsem si zvolil právě Nette Framework, bych rád řekl, že tomuto frameworku předpovídám velkou budoucnost. Nejeden framework bude jistě v budoucnu

právě tímto frameworkem zcela zastíněn.

6.1 MVC(P) architektura

Model MVC je softwarová architektura, která rozděluje celou aplikaci na datový model, řídící logiku a uživatelské rozhraní. Zkratky MVC reprezentují M*odel-View-Controller*. Nette Framework využívá modelový vzor MVP (*Model-View-Presenter*). Zjednodušeně by se dalo říct, že *Presenter* v Nette Framework je totéž, co *Controller* v jiných frameworcích. [16]

6.2 GUI komponenta jako modul

Protože je GUI komponenta distribuována i s ukázkovou stránkou, byla celá aplikace psána formou modulů - v jednom modulu se nachází ukázková stránka a ve druhém je pak GUI komponenta. Díky této struktuře je snadnější i implementace do již existující aplikace.

6.3 Testovací stránka

Testovací stránka je dodávána spolu s GUI komponentou, aby si uživatel, který si tuto komponentu stáhne, mohl ihned vyzkoušet její funkčnost a až podle toho, jak ho tato komponenta zaujme a jak ji shledá pro svůj účel užitečnou, ji mohl teprve implementovat do své aplikace.

6.3.1 Role

Testovací stránka využívá celkem 15 rolí. Výchozí role jsou *Acl Admin*, který má přístup do administrace přístupových práv a role *guest*, jenž je výchozí rolí pro nepřihlášeného uživatele.

Dále je 13 rolí, které jsou rozděleny do 2 hierarchických stromů. První strom má na nejvyšší pozici role, která má nejvyšší práva a na nejnižší pozici je naopak role s nejnižšími právy.

Druhý hierarchický strom je opakem prvního – na nejvyšší pozici je role s nejnižšími právy a na nejnižší pozici je zase role s nejvyššími právy.

Výhody a nevýhody těchto způsobů dědění práv jsou popsány právě na testovací stránce.

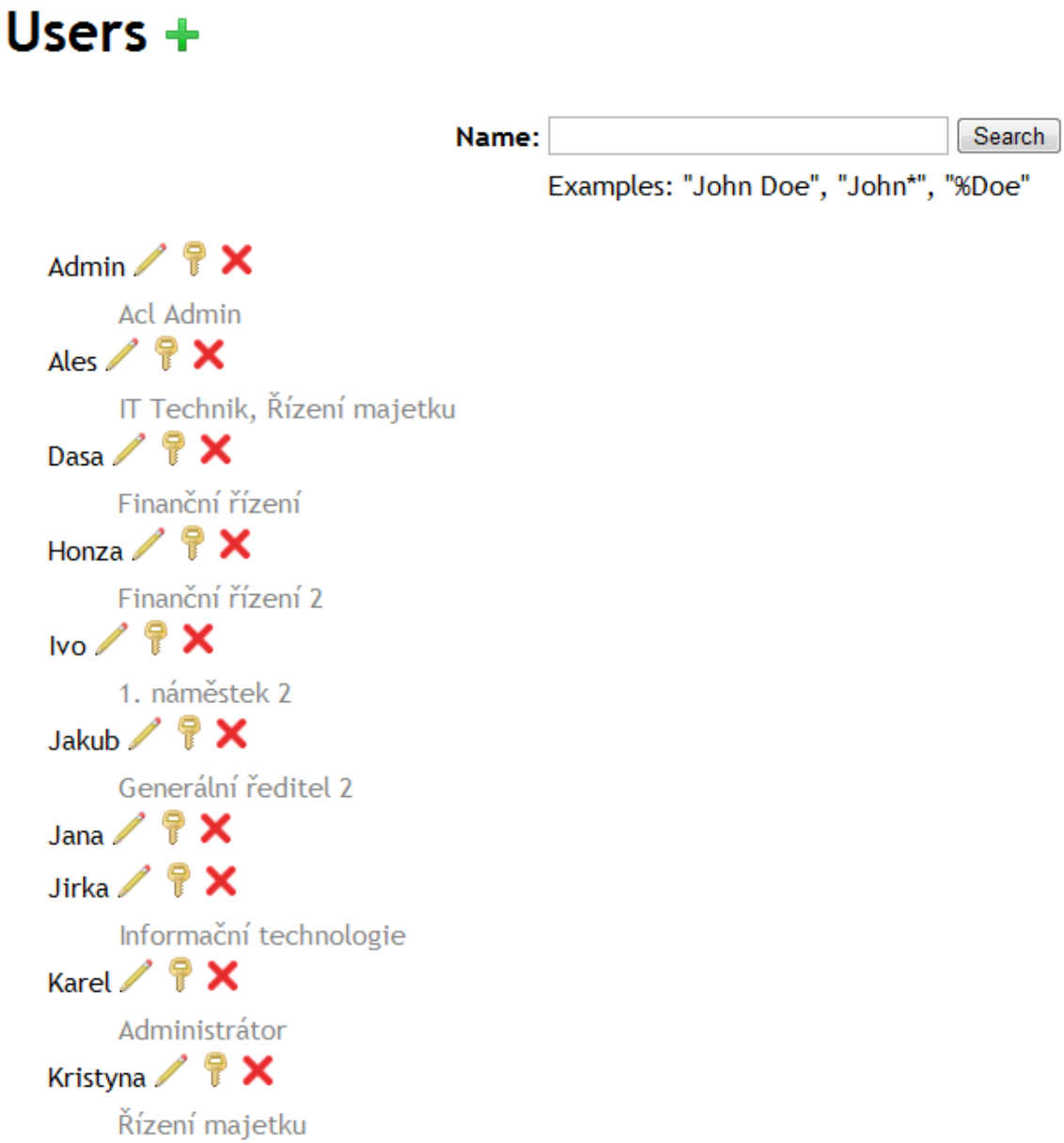
6.3.2 Uživatelé

Uživatelů je celkem 16, aby si zájemce o komponentu mohl vyzkoušet různé kombinace nastavených rolí a jejich oprávnění.

6.4 Popis GUI komponenty

6.4.1 Záložka Users

Slouží k administraci uživatelů. U každého uživatele jsou zobrazeny všechny role, jejichž je členem. Uživatele můžeme prostřednictvím této záložky vytvářet, upravovat, mazat a prohlížet oprávnění, která má daný uživatel nastavené. Záložka Users také nabízí vyhledávání mezi uživateli.

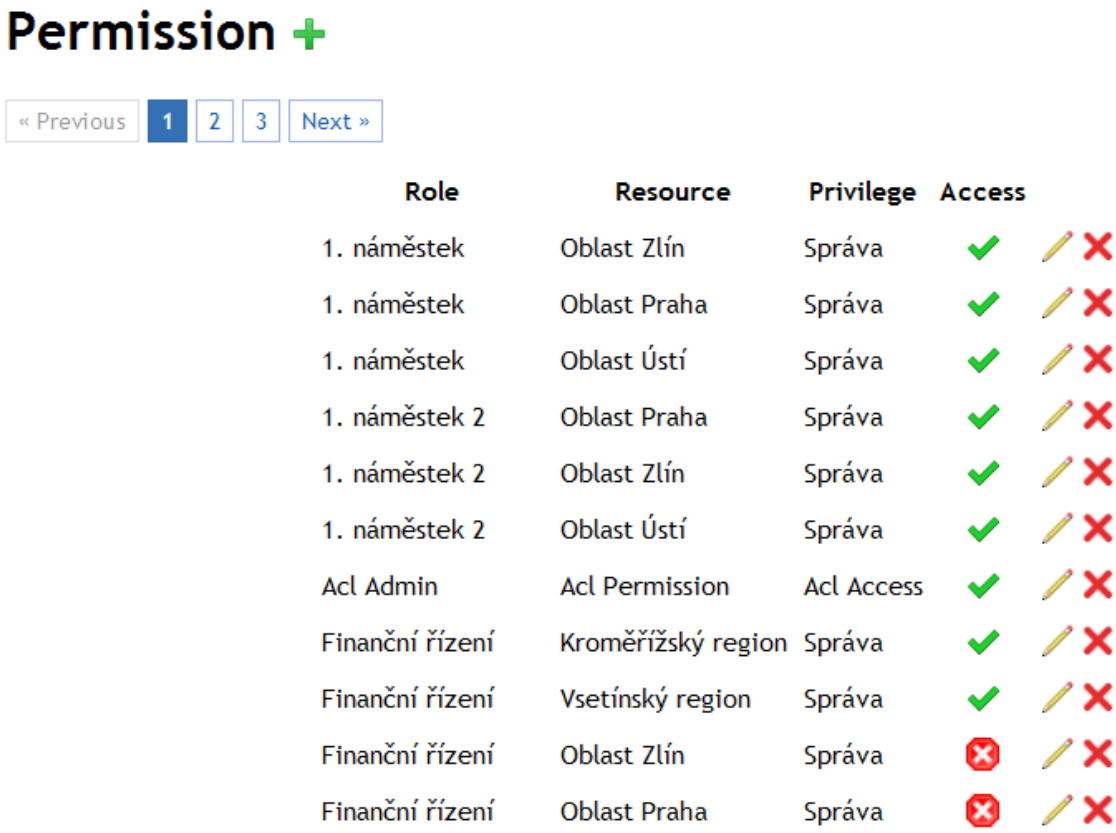


Obr. 9: Záložka Users

- Vytvoření nového uživatele
- Úprava již existujícího uživatele
- Zobrazení všech zdrojů s příslušnými právy, do kterých má daný uživatel povolen vstup
- Smazání uživatele

6.4.2 Záložka Permission

Zobrazuje veškerá vytvořená pravidla pro oprávnění. Pravidla lze vytvářet, upravovat a mazat.

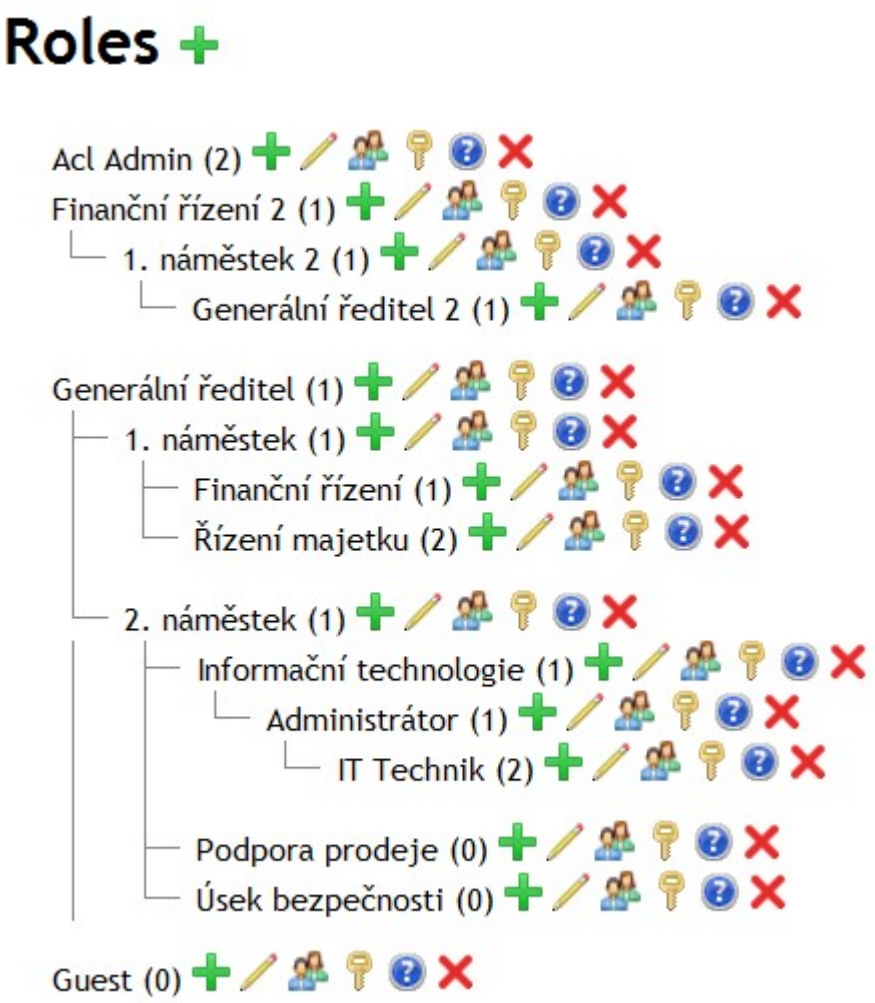


Obr. 10: Záložka Permission

- Vytvoření nového pravidla
- Úprava již existujícího pravidla
- Smazání pravidla

6.4.3 Záložka Roles

Zobrazuje všechny role a to v jejich případných stromových strukturách. Role lze vytvářet, upravovat, zobrazit uživatele patřící do dané role, zobrazit nastavené oprávnění a mazat role.



Obr. 11: Záložka Roles

- Vytvoření nové role
- Úprava již existující role
- Zobrazení všech uživatelů patřící do dané role
- Zobrazení všech zdrojů s příslušnými právy, kam má daná role povolen vstup
- Po najetí myši se zobrazí poznámka k dané roli
- Smazání role

6.4.4 Záložka Resources

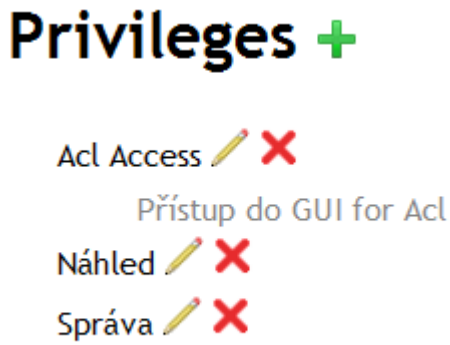
Slouží k zobrazení všech zdrojů a to včetně jejich případných stromových struktur. Zdroje lze vytvářet, upravovat a mazat.

Obr. 12: Záložka Resources

- Vytvoření nového zdroje
- Úprava již existujícího zdroje
- Po najetí myši se zobrazí poznámka k danému zdroji
- Smazání zdroje

6.4.5 Záložka Privileges

Zobrazuje všechna dostupná práva a umožňuje vytvářet nová, upravovat stávající a mazat je.



Obr. 13: Záložka Privileges

- Vytvoření nového práva
- Úprava již existujícího práva
- Smazání práva

6.5 Programátorský mód

Programátorský mód má za úkol ulehčit práci programátora při instalaci (popř. správě) GUI komponenty.

Pokud je tento mód vypnutý a my chceme vytvořit nové zdroje nebo práva, budeme muset to provést přímým zásahem do databáze, neboť tato funkce nebude v GUI dostupná a to z prostého důvodu. Zdroje a práva jsou přímo závislá na zdrojovém kódu (klíčový název se používá v metodě *isAllowed)*. Pokud tedy chceme změnit zdroje nebo práva, tak musíme změnit i zdrojový kód a to může zpravidla pouze programátor. Proto by tato funkce neměla být dostupná běžnému uživateli.

Programátorský mód zpřístupňuje vytváření zdrojů a práv a také úpravu jejich klíčových názvů.

Aby byl programátorský mód dostupný, že potřeba jej zapnout v konfiguračním souboru config.ini - jedná se o položku *acl.programmer_mode = true*.

6.6 Instalace

Popis instalace je popsán v souboru *install.cs.txt*, který je distribuován spolu s GUI komponentou.

ZÁVĚR

V prováděném testu zvítězil, co se velikosti využité paměti týče, jednoznačně systém phpGACL. Toto vítězství je nejspíš z velké části ovlivněno faktem, že phpGACL slouží výhradně ke správě přístupových práv, a proto také neobsahuje nic navíc. Kdežto Zend Framework a Nette Framework jsou komplexní systémy, které nabízí nepřeberné možnosti v oblasti vytváření webových aplikací.

Při porovnávání časové náročnosti byl jasným vítězem Nette Framework až za použití cache. Bez cache vyšly oba frameworky bez větších rozdílů stejně.

Vzorová GUI komponenta byla vytvořena v Nette Framework. Tento systém byl vybrán hned z několika důvodů – jedná o český framework, jednodušší syntaxe oproti Zend Framework, dále stále se rozrůstající komunita kolem Nette Framwork a velké množství doplňků, které usnadňují práci při vytváření webových aplikací.

GUI komponenta „spatřila světlo světa“ dne 19.4.2010, kdy byla představena na fóru Nette Frameworku. O 11 dní později byla komponenta nahrána na Google Code (http://code.google.com/p/gui-for-acl/), kde je v současné době také ke stažení.

Po překonání prvotních problémů s instalací, kdy jsem do distribuovaného archivu zapomněl přidat také exporty databáze, se celá komponenta setkala jen s kladnými ohlasy a již po 5 dnech od zveřejnění jsem byl požádán, abych tuto komponentu umístil mezi doplňky Nette Framework – což se také stalo http://addons.nettephp.com/cs/gui-for-acl.

Ke dni 23.5.2010 se moje komponenta GUI for Acl těší 123 stažení.

ZÁVĚR V ANGLIČTINĚ

In the tests won a clear system phpGACL. This victory is probably mostly influenced by the fact that phpGACL is intended solely for the administration of access rights, and therefore also does not contain anything extra. While Zend Framework and Nette Framework is a comprehensive system that offers endless possibilities for creating web applications.

When comparing the time performance with cache was the clear winner Nette Framework. Without the cache came two frameworks without major differences.

Sample GUI component was created in Nette Framework. This system has been chosen for several reasons - a Czech framework, simpler syntax compared to the Zend Framework, the still growing community around Nette Framwork and a large number of accessories that ease work for web applications.

GUI component "saw the light of day" on April 19th, 2010, which was presented at the forum Nette Framework. About 11 days later component was uploaded to Google Code (http://code.google.com/p/gui-for-acl/), where is also currently available for downloading.

After overcoming the initial problems with installation, when I was forget to add a database export to distributed archives, the entire component is met only with positive feedback and after 5 days from the publication I was asked to be placed between the component accessories Nette Framework - which it did http://addons.nettephp.com/cs/gui-for-acl.

May 23rd, 2010, my component GUI for ACL enjoys 123 downloads.

SEZNAM POUŽITÉ LITERATURY

[1] TICHÝ, Jan. Jan Tichý [online]. 2004 [cit. 2010-05-17]. Autorizace a přístupová práva. Dostupné z WWW: <http://www.jantichy.cz/diplomka/pozadavky/autorizace>.

[2] Nette Foundation. Příručka programátora [online]. 2010 [cit. 2010-05-17]. Autentizace – Přihlašování uživatelů. Dostupné z WWW: <http://doc.nettephp.com/cs/autentizace>.

[3] Nette Foundation. Příručka programátora [online]. 2010 [cit. 2010-05-17]. Autorizace – ověřování oprávnění. Dostupné z WWW: <http://doc.nettephp.com/cs/autorizace>.

[4] BENOIT, Mike. PHP Generic Access Control Lists [online]. 2007 [cit. 2010-05-17]. Applications Using phpGACL. Dostupné z WWW: <http://phpgacl.sourceforge.net/cool_apps.html>.

[5] VRÁNA, Jakub. PHP triky [online]. 2006 [cit. 2010-05-17]. Zend Framework. Dostupné z WWW: <http://php.vrana.cz/zend-framework.php>.

[6] BENOIT, Mike; RUSSELL, James; DAMBEKALNS, Karsten. Generic Access Control Lists with PHP [online]. [s.l.] : Benoit, 2006 [cit. 2010-05-17]. Dostupné z WWW: <http://phpgacl.sourceforge.net/manual.pdf>.

[7] PhpGACL 3.3.7 Developer's Manual [online]. 2006 [cit. 2010-05-17]. PhpGACL. Dostupné z WWW: <http://phpgacl.sourceforge.net/phpdoc/>.

[8] Webmasters and Blog Resources [online]. 2009 [cit. 2010-05-17]. ACL and PHPGACL – part1. Dostupné z WWW: <http://blog.tuvinh.com/acl-and-phpgacl-part1/>.

[9] Zend Framework [online]. 2010 [cit. 2010-05-17]. Programmer's Reference Guide. Dostupné z WWW: <http://framework.zend.com/manual/en/zend.acl.introduction.html>.

[10] Nette Foundation. Příručka programátora [online]. 2009 [cit. 2010-05-17]. Nette\Security\Permission. Dostupné z WWW: <http://doc.nettephp.com/cs/nette-security-permission>.

[11] Nette Foundation. Dibi is Database Abstraction Library for PHP 5. [online]. 2010 [cit. 2010-05-17]. Dibi is Database Abstraction Library for PHP 5. Dostupné z

WWW: <http://dibiphp.com/cs/>.

[12] PERSSON, Viktor. Arcsin : Skapar vänliga webbplatser [online]. 2010 [cit. 2010-05-17]. About. Dostupné z WWW: <http://arcsin.se/en/about/>.

[13] LINUXZONE [online]. 2003 [cit. 2010-05-17]. Apache JMeter. Dostupné z WWW: <http://www.linuxzone.cz/index.phtml?ids=6&idc=772>.

[14] JEŽDÍK, Radek. Osobní web Radka Ježdíka [online]. 2008 [cit. 2010-05-17]. JMeter. Dostupné z WWW: <http://redhead.utf-8.cz/clanky/?arc=33>.

[15] oXy Online s.r.o. Svět hardware [online]. 2010 [cit. 2010-05-17]. Latency. Dostupné z WWW: <http://www.svethardware.cz/glos.jsp?doc=D4E38FF35FDB80B2C1257398007F271B>. ISBN 1213-0818.

[16] Nette Foundation. Příručka programátora [online]. 2010 [cit. 2010-05-17]. Model-View-Presenter (MVP). Dostupné z WWW: <http://doc.nettephp.com/cs/model-view-presenter>.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ACL	Access Control List
ACO	Access Control Objects
ARO	Access Request Objects
AXO	Access eXtension Objects
CMS	Content Management System
GUI	Graphical User Interface
phpGACL	PHP Generic Access Control Lists

SEZNAM OBRÁZKŮ

SEZNAM GRAFŮ

SEZNAM TABULEK