

Srovnávací analýza web frameworků Django, Ruby on Rails a PHP

Comparative analysis of web frameworks Django,
Ruby on Rails and PHP

Bc. Petr Vajdík

Diplomová práce
2011



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2010/2011

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Petr VAJDÍK**
Osobní číslo: **A09436**
Studijní program: **N 3902 Inženýrská informatika**
Studijní obor: **Informační technologie**

Téma práce: **Srovnávací analýza web frameworků Django, Ruby on Rail a PHP**

Zásady pro vypracování:

1. Vypracujte literární rešerši na dané téma.
2. Na příkladu vývoje menší aplikace (cca 30 tabulek, z toho 30% s vazbami M:N, víceúrovňová uživatelská práva) srovnajte frameworky Django (Python), Ruby on Rails a některý z PHP MVC frameworků.
3. Porovnejte dostupnost kvalitních GUI komponent (gridů, záložek, ...).
4. Porovnejte rychlost vývoje aplikací.
5. Porovnejte bezpečnost – ošetření SQL injection, odolnost proti XSS a dalším běžným útokům.
6. Porovnejte podporu víceúrovňové autorizace.
7. Porovnejte zpětnou kompatibilitu API.
8. Porovnejte možnosti definice constraints, které framework dle použité databáze implementuje buď přímo v DB, nebo v modelu.
9. Porovnejte nezávislost logického a fyzického modelu (možnost přejmenovat tabulky i sloupce v DB bez nutnosti přepisovat kód).
10. Porovnejte časové/paměťové složitosti vybraných operací.

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

Seznam doporučené literatury:

1. OPPEL, Andy. Databáze bez předchozích znalostí. [s.l.] Computer Press, a.s., 2006. 320s. ISBN: 80-251-1199-7
2. KOFLER, Michael. ÖGGL, Bernd. PHP 5 a MySQL 5 – Průvodce webového programátora. [s.l.] Computer Press, a.s., 608s. ISBN: 978-80-251-1813-9
3. BÖHMER, Marian. Zend Framework – Programujeme webové aplikace v PHP. [s.l.] Computer Press, a.s., 416s. ISBN: 978-80-251-2965-4
4. KREJČÍ, Lukáš. Programming in Python 3. [s.l.] Computer Press, a.s., 584s. ISBN: 978-80-251-2737-7
5. Zend framework <http://framework.zend.com/>
6. Ruby On Rails <http://rubyonrails.org/>
7. Django <http://www.djangoproject.com/>

Vedoucí diplomové práce:

Ing. Tomáš Dulík

Ústav informatiky a umělé inteligence


Datum zadání diplomové práce:

24. února 2011

Termín odevzdání diplomové práce:

18. května 2011

Ve Zlíně dne 24. února 2011


prof. Ing. Vladimír Vašek, CSc.
děkan




doc. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

ABSTRAKT

Cílem této diplomové práce je srovnat webové frameworky Django, Ruby on Rails a Zend. Testují se různé prvky, jako třeba rychlost vývoje samotné aplikace, dostupnost kvalitních GUI komponent, zpětná kompatibilita API nebo třeba časové a paměťové nároky na běh aplikace. V každém frameworku byla vytvořena aplikace na správu uživatelů a k ní patřičné administrační rozhraní. Dále byla k aplikacím vytvořena dokumentace ve formě diplomové práce s podrobným popisem testovacích aplikací a jejich výsledků. Dokumentace se skládá z teoretické a praktické části.

Klíčová slova: DJANGO, RUBY ON RAILS, ZEND FRAMEWORK.

ABSTRACT

The aim of this thesis is to compare web frameworks Django, Ruby on Rails and Zend. They test the different elements, such as speed of development applications themselves, the availability of high-quality GUI components, backward compatibility API or need time and memory requirements for running applications. Each framework was created to manage users applications and its proper administration interface. Furthermore, the applications created in the documentation of my thesis with a detailed description of test and application of their results. Documentation consists of theoretical and practical parts.

Keywords: DJANGO, RUBY ON RAILS, ZEND FRAMEWORK.

Poděkování, motto

Chtěl bych poděkovat vedoucímu diplomové práce panu Ing. Tomášovi Dulíkovi za pomoc při tvorbě diplomové práce a také za cenné připomínky. Dále bych chtěl poděkovat svým rodičům a přátelům za trpělivost a podporu při studiu.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/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 diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/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 diplomovou/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 – diplomovou/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 diplomové/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 diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/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 diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

OBSAH

ÚVOD	10
I TEORETICKÁ ČÁST	11
1 ZÁKLADNÍ POJMY	12
1.1 FRAMEWORK.....	12
1.2 MVC ARCHITEKTURA.....	12
2 TESTOVANÉ MVC FRAMEWORKY	13
2.1 DJANGO.....	13
2.2 RUBY ON RAILS.....	14
2.3 ZEND FRAMEWORK.....	14
2.3.1 Databázová vrstva Doctrine	16
3 POUŽITÉ KOMPONENTY A TECHNOLOGIE	17
3.1 VÝVOJOVÉ PROSTŘEDÍ	17
3.1.1 Eclipse (Django).....	17
3.1.2 RadRails	17
3.1.3 Zend Studio	17
3.2 DATABÁZE	17
3.2.1 Mysql.....	18
3.3 PROGRAM NA TESTOVÁNÍ ČASOVÝCH A PAMĚŤOVÝCH SLOŽITOSTÍ.....	19
3.3.1 Apache JMeter.....	19
II PRAKTICKÁ ČÁST	20
4 PARAMATRY TESTOVACÍHO POČÍTAČE	21
5 VYTVOŘENÉ TESTOVACÍ APLIKACE	22
5.1 NÁVRH DATABÁZE	22
5.1.1 Popis databáze	22
5.1.2 Popis tabulek	22
5.1.3 Vzájemné vazby	23
5.2 DJANGO, PYTHON.....	24
5.2.1 Ukázka instalace a spuštění Djanga	24
5.2.2 Ukázka zdrojového kódu.....	25
5.2.3 Vytvořená aplikace - správa uživatelů	26
5.3 RUBY ON RAILS.....	30
5.3.1 Ukázka instalace a spuštění Ruby on Rails	30
5.3.2 Ukázka zdrojového kódu.....	31
5.3.3 Vytvořená aplikace - správa uživatelů	31
5.4 ZEND FRAMEWORK.....	35
5.4.1 Ukázka instalace a spuštění Zend frameworku	35
5.4.2 Ukázka zdrojového kódu.....	36
5.4.3 Vytvořená aplikace - správa uživatelů	37

5.5	REALIZACE ČASOVÉHO/PAMĚŤOVÉHO TESTU - JMETER.....	40
5.5.1	Popis nastavení.....	40
6	TESTOVÁNÍ MVC FRAMEWORKŮ	44
6.1	DJANGO, PYTHON.....	44
6.1.1	GUI komponenty	44
6.1.2	Rychlost vývoje aplikace.....	45
6.1.3	Bezpečnost - SQL injection, XSS útoky... ..	45
6.1.4	Víceúrovňová autorizace.....	47
6.1.5	Zpětná kompatibilita API.....	51
6.1.6	Definice constraints v databázi nebo v modelu.....	51
6.1.7	Nezávislost fyzického a logického modelu (přejmenování tabulek).....	52
6.1.8	Časové a paměťové testy.....	54
6.2	RUBY ON RAILS.....	55
6.2.1	GUI komponenty	55
6.2.2	Rychlost vývoje aplikace.....	56
6.2.3	Bezpečnost - SQL injection, XSS útoky... ..	57
6.2.4	Víceúrovňová autorizace.....	58
6.2.5	Zpětná kompatibilita API.....	60
6.2.6	Definice constraints v databázi nebo v modelu.....	61
6.2.7	Nezávislost fyzického a logického modelu (přejmenování tabulek).....	62
6.2.8	Časové a paměťové testy.....	63
6.3	ZEND FRAMEWORK.....	65
6.3.1	GUI komponenty	65
6.3.2	Rychlost vývoje aplikace.....	66
6.3.3	Bezpečnost - SQL injection, XSS útoky... ..	66
6.3.4	Víceúrovňová autorizace.....	67
6.3.5	Zpětná komptabilita API.....	69
6.3.6	Definice constraints v databázi nebo v modelu.....	70
6.3.7	Nezávislost fyzického a logického modelu (přejmenování tabulek).....	71
6.3.8	Časové a paměťové testy.....	72
7	POROVNÁNÍ VÝSLEDKŮ	74
7.1	POROVNÁNÍ DOSTUPNOSTI GUI.....	74
7.2	POROVNÁNÍ RYCHLOSTI VÝVOJE APLIKACÍ.....	74
7.3	POROVNÁNÍ BEZPEČNOSTI.....	74
7.4	POROVNÁNÍ VÍCEÚROVŇOVÉ AUTORIZACE.....	75
7.5	POROVNÁNÍ ZPĚTNÉ KOMPATIBILITY API.....	75
7.6	POROVNÁNÍ DEFINICE CONSTRAINTS	75
7.7	POROVNÁNÍ NEZÁVISLOSTI FYZICKÝCH A LOGICKÝCH MODELŮ	75
7.8	POROVNÁNÍ ČASOVÝCH A PAMĚŤOVÝCH HODNOT.....	76
7.9	DOPLŇUJÍCÍ INFORMACE.....	80
	ZÁVĚR	81
	ZÁVĚR V ANGLIČTINĚ	82

SEZNAM POUŽITÉ LITERATURY.....	83
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	84
SEZNAM OBRÁZKŮ	86
SEZNAM TABULEK.....	88
SEZNAM PŘÍLOH.....	89

ÚVOD

Tato diplomová práce popisuje srovnání známých webových frameworků Django, Ruby on Rails a PHP frameworku Zend podle různých kritérií popsaných v zadání.

Vybral jsem si toto téma, protože mě zajímá tvorba internetových stránek pomocí nejnovějších technologií a trendů. Chtěl jsem se také naučit efektivně programovat pomocí frameworku Zend, který je napsán v programovacím jazyce PHP.

Práce je rozdělena na teoretickou a praktickou část. V teoretické části jsou popsány obecně jednotlivé webové technologie a k nim příslušná vývojová prostředí.

V praktické části je popsán postup tvorby jednotlivých aplikací od samotného návrhu databáze až po samotný návrh jednotlivých aplikací, na kterých jsou následně testovány jednotlivé body testu popsané v zadání. Na závěr jsou popsány výsledky, které vznikly testováním nebo vývojem samotné aplikace.

I. TEORETICKÁ ČÁST

1 ZÁKLADNÍ POJMY

1.1 Framework

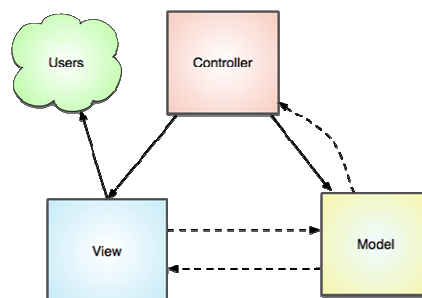
Pod pojmem framework si můžeme představit takovou strukturu, která kladným způsobem rozšiřuje nebo vylepšuje původní softwarovou strukturu programovacího jazyka. Framework obsahuje přidané podpůrné knihovny, helpery nebo také návrhové vzory. Snaží se vyřešit problémy, které programátoři řeší velmi často, jako třeba připojení k databázi, formuláře a validace zadaných dat nebo tvorbu stránkovaní.

Programátor se pak může lépe soustředit na zadaný úkol a nemusí řešit tyto základní problémy. Jedinou velkou nevýhodou používání frameworku je nutnost nastudování funkčnosti daného frameworku, což nemusí být otázka pár hodin. Pokud je ovšem daný framework dostatečně zvládnutý a pochopený, výrazně spoří čas, a to zejména při větších projektech nebo při opakovaném použití.

Téměř každý vyspělejší programovací jazyk má svůj oficiální framework. V programovacím jazyku PHP ovšem neexistuje žádné oficiální řešení, ale naštěstí je zde velké množství Open Source (otevřený zdrojový kód) řešení jako třeba Zend framework, CakePHP, Kohana nebo framework Nette.

1.2 MVC architektura

Model-view-controller (MVC) je moderní softwarová architektura, která má za úkol rozdělit datový model aplikace na uživatelské rozhraní (view), řídicí logiku (controller) a na doménovou nebo datovou vrstvu (model). Toto rozdělení do tří nezávislých komponent aplikace zajistí to, že modifikace některé z nich má jen minimální vliv na ostatní. Tato architektura je dnes hojně využívána v řadě frameworků, které budou popsány níže.



Obr. 1 - MVC architektura

2 TESTOVANÉ MVC FRAMEWORKY

V této kapitole je obecně popsán framework Django, Ruby on Rails, Zend framework a příslušná vývojová prostředí. Dále je zde obecný popis databáze MySQL a aplikace JMeter, která dokáže otestovat paměťové a časové nároky jednotlivých frameworků.

2.1 Django

Django je Open Source (otevřený zdrojový kód) webový aplikační framework napsaný v Pythonu, který se volně drží architektury MVC (model-view-controller). Původně bylo navrženo pro správu několika zpravodajsky orientovaných stránek společnosti The World Company v Lawrenci v Kansasu. Je dalším z řady webových frameworků, jako je např. Ruby on Rails nebo Zend. Ovšem oproti Ruby on Rails nebo Zendu nabízí zajímavou možnost automatické tvorby administrace projektu, která je generována dynamicky podle datového modelu. Hlavní úkol Djanga je snadné vytvoření komplexních a databází řízených webových aplikací. Zaměřuje se na znovupoužitelnosti a připojitelnosti všech komponent, rychlý vývoj a především na koncept „DRY“ (don't repeat yourself – neopakovat se). [1]

Nově vygenerovaná aplikace v Djangu má následující strukturu:

- views.py - obsahuje jednotlivé view funkce
- urls.py - obsahuje mapování url na jednotlivá view
- models.py - obsahuje popis datového modelu aplikace
- tests.py - obsahuje jednotkové testy
- __init__.py - dělá z aplikace balíček Pythonu

Hostování Pythonu a potažmo Django frameworku dnes podporuje celá řada hostingů jako třeba Klenot nebo G-hosting. Seznam zahraničních lze nalézt na webu Django Friendly.

2.2 Ruby on Rails

Ruby on Rails je další plnohodnotný framework pro vývoj webových aplikací napojených na databázi, který používá návrhový vzor MVC (model-view-controller). Rails jsou vytvořeny v programovacím jazyce Ruby. Ke spuštění naší aplikace pak potřebujeme jen databázový a webový server.

Mezi základní princip Rails patří konvence, která má přednost před konfigurací. Programátor tedy konfiguruje pouze ty části aplikace, které se liší od běžného nastavení. Vytvoří-li tedy např. model Person, aplikace bude data automaticky hledat v tabulce person. Chce-li, aby aplikace načítala data z tabulky animal, musí tak učinit výslovně.

Rails jsou postaveny na bázi návrhového vzoru MVC (model-view-controller), který odděluje části aplikace zodpovědné za čtení a ukládání dat včetně manipulace s nimi (model), za zobrazení grafického rozhraní aplikace (view) a za část přijímající vstupy od uživatele a řídící zobrazení dat na výstupu (controller).

Základní vlastnosti Rails:

- automaticky mapují URL na vnitřní řídicí prvky aplikace (routing)
- zajišťují předávání dat mezi controllerem a modelem, mezi controllerem a view
- abstrahují přístup k datům v databázi pomocí mapování záznamů z relační databáze na objekty (pomocí návrhového vzoru ActiveRecord se „řádky“ v databázi převedou na instance objektů, „sloupce“ na jejich atributy)
- obsahují rozsáhlé pomocné knihovny pro snadné generování HTML, pro práci s Ajaxem (využívá javascriptový Framework Prototype), formátování dat a další [2]

2.3 Zend framework

Zend Framework je Open Source objektově orientovaný webový aplikační framework implementovaný v PHP 5 a licencovaný pod New BSD license. Zend Framework je vyvíjen s ohledem na jednoduchý vývoj webových aplikací. Užívá modulární architektury, která umožňuje vývojářům použít jen ty komponenty, které potřebují. Zend framework v sobě zahrnuje komponenty pro MVC aplikace. Dále zahrnuje také autorizaci nebo autentifikaci. Implementuje různé druhy cache, filtrů a validátorů pro uživatelská data a

jazykové komponenty. Začal být vyvíjen na počátku roku 2005, kdy mnoho nových frameworků, jako Ruby on Rails a Spring Framework, získávali na popularitě. [3]

Stručný výčet nejznámějších komponent frameworku:

- Zend_Acl - jednoduchý a flexibilní systém pro správu uživatelských oprávnění
- Zend_Application - knihovna pro načtení a správu základních prvků aplikace (bootstrapping)
- Zend_Auth - autentifikace uživatelů s mnoha druhy úložišť
- Zend_Cache - implementace cache systému s úložišti ve formě paměti, souboru, APC, SQLitea atd...
- Zend_Config - slouží k nastavení aplikace skrze konfigurační soubory
- Zend_Controller - implementace Model-View-Controller (MVC) architektury
- Zend_Date - komponenta pro práci s datem
- Zend_Db - implementace multi-databázové vrstvy
- Zend_Dojo - knihovna pro práci s javascriptovým frameworkem Dojo
- Zend_Filter - komponenta pro filtrování uživatelských dat s velkým množstvím filtrů
- Zend_Form - objektový vývoj webových formulářů včetně filtrování hodnot a jejich validace
- Zend_Layout - správa layoutů aplikace
- Zend_Log - komponenta pro logování určitých dat s množstvím backendů
- Zend_Mail - tvorba e-mailů, správa e-mailových schránek
- Zend_Memory - podpora pro zpracování dat s omezeným množstvím dostupné paměti
- Zend_OpenID - implementace OpenID klienta i serveru
- Zend_Paginator - komponenta pro práci se stránkováním dat
- Zend_Pdf - objektový přístup a vytváření PDF souborů

- Zend_Registry - komponenta pro uchovávání objektů a hodnot v aplikační vrstvě
- Zend_Translate - podpora pro překlady a různé jazykové mutace aplikací
- Zend_View - šablonovací systém
- ZendX_Jquery - podpora javascriptového frameworku jQuery

2.3.1 Databázová vrstva Doctrine

Pro práci s databází ve frameworku Zend můžeme, ale nemusíme využít předností ORM (object-relational mapping) frameworku Doctrine. Jedná se tedy o databázovou vrstvu a v porovnání s již existujícími systémy pro mapování objektů na relační databázi přináší zajímavý posun. Má velkou šanci stát se v budoucnu převládajícím ORM pro aplikace psané v jazyce PHP. Nabízí totiž možnost pracovat nezávisle na různých databázích (MySQL, PostgreSQL...), ochranu proti SQL injection nebo také možnost definovat omezení pro jednotlivé sloupce v databázové tabulce (constraints).

Doctrína je ve skutečnosti rozdělena do tří vrstev:

- Common - definuje základní obecná rozhraní, třídy a knihovny. Například nástroje pro práci s kolekcemi, anotacemi, cachováním, událostmi apod. Ty jsou pak využívány oběma vyššími vrstvami. Common samo o sobě je ale na zbylých vrstvách nezávislé, takže ho teoreticky můžete používat i samostatně. Vše je definováno v namespace Doctrine\Common.
- DBAL (DataBase Abstraction Layer) - abstrahuje zbytek aplikace od konkrétního typu databáze a zavádí tzv. notaci DQL (Doctrine Query Language), což jsou SQL dotazy interpretované Doctriou. DBAL je závislý na Common, ale může se používat i samostatně bez poslední ORM vrstvy. Vše je definováno v namespace Doctrine\DBAL.
- ORM (Object-Relational Mapping) - nejvyšší vrstva, která zajišťuje mapování aplikačních objektů na relační databázi a jejich načítání. ORM je závislá na DBAL i Common. Namespace této vrstvy je Doctrine\ORM. [4]

3 POUŽITÉ KOMPONENTY A TECHNOLOGIE

3.1 Vývojové prostředí

Zde jsou popsána vývojová prostředí, ve kterých byly vyvíjeny testovací aplikace. Pokud nevyužijeme doporučených vývojových prostředí, můžeme využít k tvorbě projektu například PSPad.

3.1.1 Eclipse (Django)

Programovací prostředí Eclipse je Open Source vývojová platforma, která je známa jako vývojové prostředí IDE určené pro programování především v jazyce Java. Flexibilní návrh této platformy dovoluje rozšířit seznam podporovaných programovacích jazyků za pomoci pluginů, například o C++, PHP nebo také pro Django.

3.1.2 RadRails

RadRails je rychlé vývojové prostředí IDE pro Ruby on Rails. Cílem je poskytovat vývojářům všechno, co potřebují. Například rozvíjet, řídit, testovat a nasazovat nové aplikace. RadRails IDE je postaveno také na Eclipse.

3.1.3 Zend Studio

Zend Studio je IDE pro vývoj internetových řešení. Celé je napsáno v Javě, což mu umožňuje relativní platformovou nezávislost. Mezi jeho velké výhody patří PHP debugger, doplňování syntaxe a spolupráce s FTP a SQL servery. Kromě PHP podporuje JavaScript, (X)HTML, XML, CSS a SQL. [5]

3.2 Databáze

Databáze má dnes veliké uplatnění v oblasti webu, bez které se žádný větší systém neobejde. Slouží k ukládání různých záznamů, se kterými se pak následně pracuje. Existuje několik databází. Mezi nejznámější patří MySQL, PostgreSQL, Oracle...

3.2.1 Mysql

MySQL (My Structured Query Language) znamená ve zkratce systém pro řízení databází. Do MySQL lze ukládat různá data (texty, obrázky) s nimiž lze dále jednoduše pracovat (třídít, řadit, filtrovat apod.). Nejčastěji se MySQL používá ve spojení s jazykem PHP, který umožňuje přístup k uloženým datům. Každá databáze v MySQL obsahuje tabulky. Každá tabulka má sloupce a řádky - v každém řádku jsou záznamy předem určeného typu. Databáze MySQL je jeden z prvních hojně rozšířených systémů. Práce s tímto systémem se dá využít v C, C++, Javě, Ruby, PHP nebo také v Pythonu.

Pro jednoduchou správu MySQL databází se používá nástroj phpMyAdmin. PhpMyAdmin je Open Source program napsaný v PHP, který umožňuje zálohování, vytváření tabulek, vkládání, editaci a mazání záznamů v tabulkách nebo vytváření nových databází. PhpMyAdmin je pokročilý nástroj pro kompletní správu MySQL systému přes webové rozhraní. [6]

Tabulka	Akce	Záznamů ¹	Typ	Porovnávání	Velikost	Navic
adresy		20	InnoDB	utf8_czech_ci	16.0 KiB	-
auth_group		0	InnoDB	utf8_czech_ci	32.0 KiB	-
auth_group_permissions		0	InnoDB	utf8_czech_ci	64.0 KiB	-
auth_message		0	InnoDB	utf8_czech_ci	32.0 KiB	-
auth_permission		51	InnoDB	utf8_czech_ci	48.0 KiB	-
auth_user		2	InnoDB	utf8_czech_ci	32.0 KiB	-
auth_user_groups		0	InnoDB	utf8_czech_ci	64.0 KiB	-
auth_user_user_permissions		0	InnoDB	utf8_czech_ci	64.0 KiB	-
django_admin_log		153	InnoDB	utf8_czech_ci	48.0 KiB	-
django_content_type		17	InnoDB	utf8_czech_ci	32.0 KiB	-
django_session		1	InnoDB	utf8_czech_ci	16.0 KiB	-
django_site		1	InnoDB	utf8_czech_ci	16.0 KiB	-
fyzicke_osoby		11	InnoDB	utf8_general_ci	16.0 KiB	-
kontakty		21	InnoDB	utf8_czech_ci	16.0 KiB	-
osoby		18	InnoDB	utf8_general_ci	16.0 KiB	-
osoby_ma_adresu		11	InnoDB	utf8_czech_ci	32.0 KiB	-
osoby_ma_kontakt		9	InnoDB	utf8_czech_ci	32.0 KiB	-
pravnicke_osoby		18	InnoDB	utf8_czech_ci	16.0 KiB	-
pravnicke_osoby_ma_adresu		9	InnoDB	utf8_czech_ci	32.0 KiB	-
pravnicke_osoby_ma_kontakt		12	InnoDB	utf8_czech_ci	32.0 KiB	-
pravnicke_osoby_ma_osobu		7	InnoDB	utf8_czech_ci	32.0 KiB	-

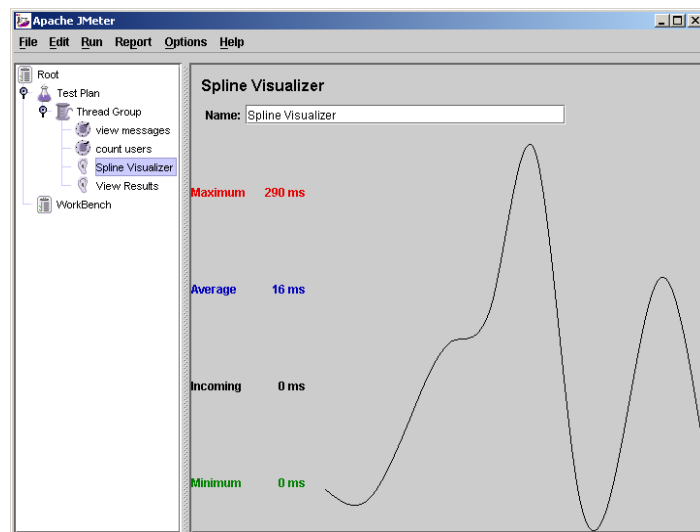
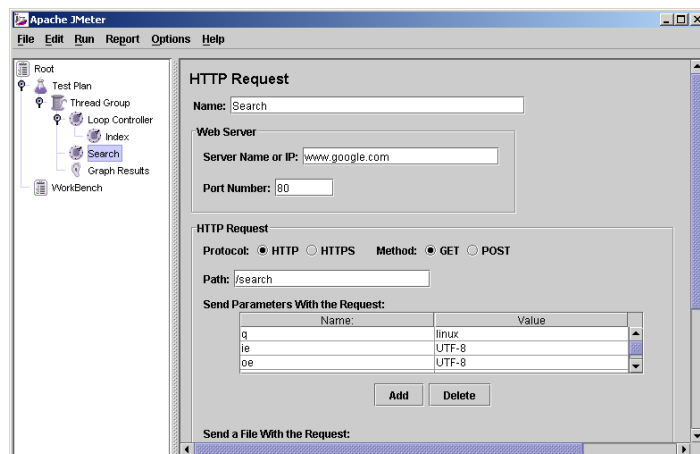
Obr. 2 - databázové rozhraní phpMyAdmin

3.3 Program na testování časových a paměťových složitostí

3.3.1 Apache JMeter

JMeter slouží k měření výkonu webových aplikací HTTP/HTTPS, FTP serverů a relačních databází. Pomocí JMeteru můžeme vytvořit pro aplikaci specifický test, který simuluje reálnou zátěž během provozu. Můžeme specifikovat, kolik klientů současně bude posílat požadavky a s jakými parametry.

Je napsán kompletně v jazyce Java a nabízí grafické rozhraní v podobě swing komponent nebo může běžet v non-GUI módu, kdy nevyžaduje na počítači, ze kterého se testuje, žádné grafické prostředí. Další mód je pro tzv. Remote testing a funguje tak, že se na testovaném serveru spustí pouze serverová část, která odesílá požadavky a výsledky posílá klientovi na jiném stroji. To umožňuje obejít místa mezi klientem a serverem, která zpomalují test. [7]



Obr. 3 - aplikace JMeter

II. PRAKTICKÁ ČÁST

4 PARAMATRY TESTOVACÍHO POČÍTAČE

Celý vývoj a rychlostní testy byly prováděny na notebooku HP Pavilion dv6000 s těmito parametry:

- Procesor: Intel(R) Pentium(R) Dual CPU T2330 1.60 GHz
- Paměť: 3072 MB (PC2-5300 DDR2-333)
- HDD: Samsung 160GB ATA Device (IDE)
- Systém: Windows 7 Professional 64-bit

Popis jednotlivých verzí frameworku a testovacích aplikací budou popsány v příslušných odstavcích.

5 VYTVOŘENÉ TESTOVACÍ APLIKACE

Byly vytvořeny tři testovací aplikace s názvem „Správa uživatelů“ v různých webových jazycích. Na každé dílčí aplikaci jsou testovány různé požadavky podle zadání. Níže budou jednotlivé aplikace popsány.

5.1 Návrh databáze

V testovacích aplikacích je využívána databáze MySQL s administračním rozhraním phpMyAdmin. PhpMyAdmin umožňuje jednoduchou správu navržené databáze a nabízí přehledný náhled na jednotlivé dílčí databáze a tabulky.

5.1.1 Popis databáze

Pro běh databáze byla nainstalovaná aplikace EasyPHP ve verzi 5.3.5.0, která v sobě integruje již zmíněnou databázi MySQL ve verzi 5.1.54.

5.1.2 Popis tabulek

Pro správný běh aplikace „Správa uživatelů“ je vyžadováno devět základních stěžejních tabulek. Jsou to tyto tabulky:

osoby - jedná se o fyzické osoby, které mohou mít přiděleny kontakty nebo adresy

pravnicke_osoby - jedná se o právnické osoby, které mohou mít přiděleny kontakty nebo adresy

kontakty - zde jsou uloženy informace o kontaktech

adresy - zde jsou uloženy informace o adresách

osoba_ma_kontakt - slouží jako propojovací tabulka mezi fyzickou osobou a kontaktem

osoba_ma_adresu - slouží jako propojovací tabulka mezi fyzickou osobou a adresou

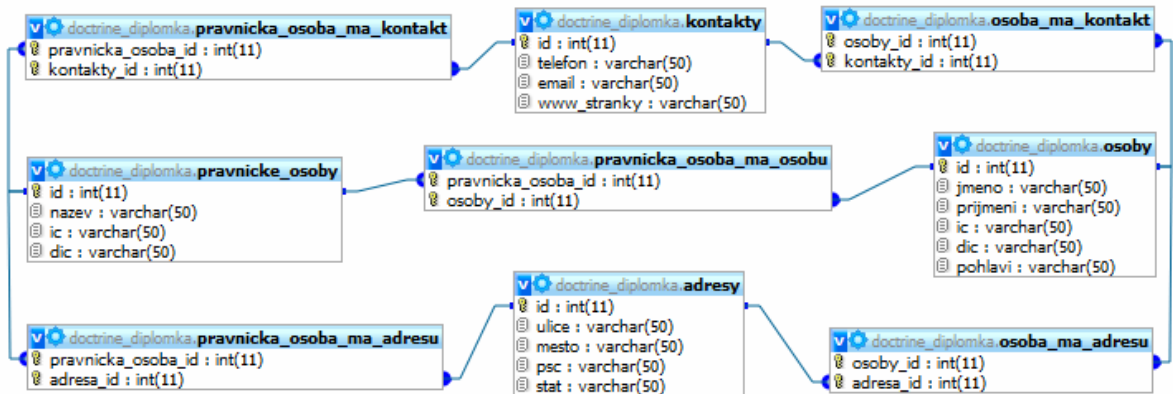
pravnicka_osoba_ma_kontakt - slouží jako propojovací tabulka mezi právnickou osobou a kontaktem

pravnicka_osoba_ma_adresu - slouží jako propojovací tabulka mezi právnickou osobou a adresou

pravnicka_osoba_ma_osobu - slouží jako propojovací tabulka mezi právnickou osobou a fyzickou osobou

5.1.3 Vzájemné vazby

Jednotlivé tabulky a vzájemné vazby je možné vidět na obrázku níže.



Obr. 4 - databázové tabulky pro testovanou aplikaci „Správa uživatelů“

Ovšem pro běh celého administračního systému pro systém Django jsou vyžadovány i další tabulky jako třeba:

auth_group, auth_group_permissions, auth_message, auth_permission, auth_user, auth_user_groups, auth_user_user_permissions, django_admin_log, django_content_type, django_session, django_site.

Jsou to automaticky vygenerované tabulky pomocí administračního rozhraní Django sloužící k ukládání různých nastavení nebo logování uživatele. Na obrázku níže jsou zobrazeny zbylé databázové tabulky.

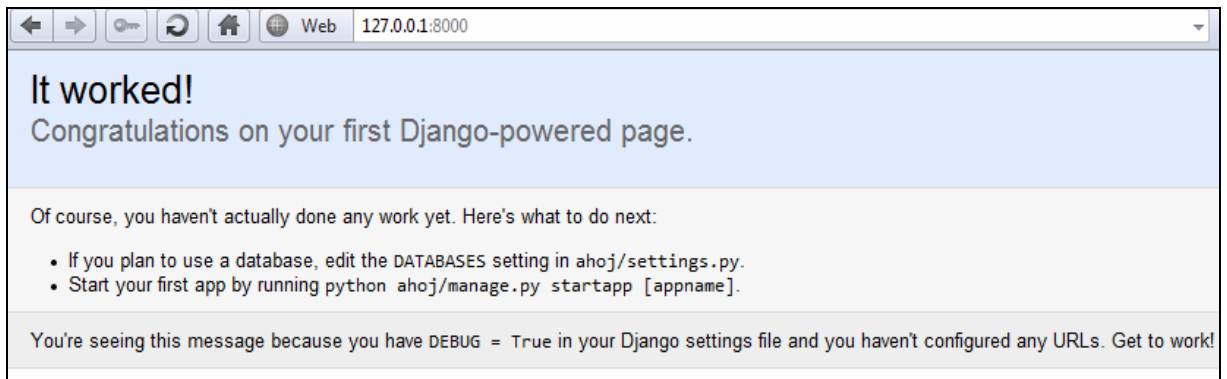

```
django-admin.py startproject nazev_projektu
```

Spustíme server pomocí příkazu:

```
python manage.py runserver
```

Otevřeme webovou stránku:

```
http://127.0.0.1:8000/
```



Obr. 6 - úvodní stránka frameworku Django

Pokud budeme chtít v naší aplikaci pracovat s databází MySQL, musíme ještě doinstalovat databázový balíček **MySQL for Python** např. z adresy <http://sourceforge.net/projects/mysql-python/>.

Podrobný popis instalace Pythonu a Django můžete najít například těchto odkazech:

<http://standa.vaskovi.cz/instalace-python-a-django-pod-windows-7/>,

<http://zdrojak.root.cz/clanky/django-uvod-a-instalace/>

5.2.2 Ukázka zdrojového kódu

```
1 def f_osoba_detail(request, f_id):
2     try:
3         adresy = ""
4         kontakty = ""
5         osoby = Osoby.objects.filter(id=f_id)[0]
6         for p in id:
7             index = (p.adresy_id)
8             adresy = Adresy.objects.filter(id=index)[0]
```

```

9  except IndexError:
10         return HttpResponseRedirect('/fyzicke_osoby/')

```

5.2.3 Vytvořená aplikace - správa uživatelů

Po spuštění vytvořené aplikace na Python serveru se na adrese **http://127.0.0.1:8000/** zobrazí úvodní webová prezentace s přehledem všech zaregistrovaných uživatelů s jejich nastavenými údaji. Lze volit mezi fyzickými nebo mezi právníckými osobami. Pro zobrazení vyplněných údajů je potřeba na vybranou osobu kliknout. Pokud chceme editovat nebo přidávat jednotlivé osoby, musíme přejít do administračního rozhraní. Stačí kliknout na odkaz Admin.



The screenshot shows the Django web interface for user management. At the top left is the Django logo. The main heading is "Správa uživatelů - přehled". Below this, there are two links: "Fyzické osoby" and "Právnícké osoby". The "Fyzické osoby" link is selected, and a table of users is displayed. The table has columns for "Jmeno", "IC", "DIC", and "Pohlavi:". The "Admin" link is also visible on the left side of the table.

<u>Admin</u>	Jmeno	IC	DIC	Pohlavi:
	Petr Nový	78907789	CZ8790789	Muž
	Jana Malá	97897897	CZ8078709	Žena
	Radka Králíková	78978908	CZ5675675	Žena
	David Král	87879898	CZ6565767	Muž
	Jan Kadlčík	353637	CZ6383903	Muž
	Tomáš Holý	3637489	CZ0980998	Muž
	Jan Kraus	3554887	CZ76876989	Muž
	Lucie Pyšná	3876746	CZ7867876	Žena
	Tomáš Zamazal	38497873	CZ8798899	Muž
	Michaela Koválová	7837387	CZ8789709	Žena
	Eva Donutilová	354664	CZ7378378	Žena
	Jan Mantl	7897997	CZ098098	Muž
	Tomáš Dvořák	8798789	CZ897897	Muž
	Jan Donutil	7677389	CZ8789987	Muž
	Zuzana Kovářová	878978	CZ897899	Žena
	Jarka Malá	8978909	CZ9080908	Žena
	Petr Doubrava	8789789	CZ9879897	Muž
	Jindřiška Pokorná	7876878	CZ8787890	Žena

Obr. 7 - úvodní stránka vytvořené aplikace „Správa uživatelů“ v Django

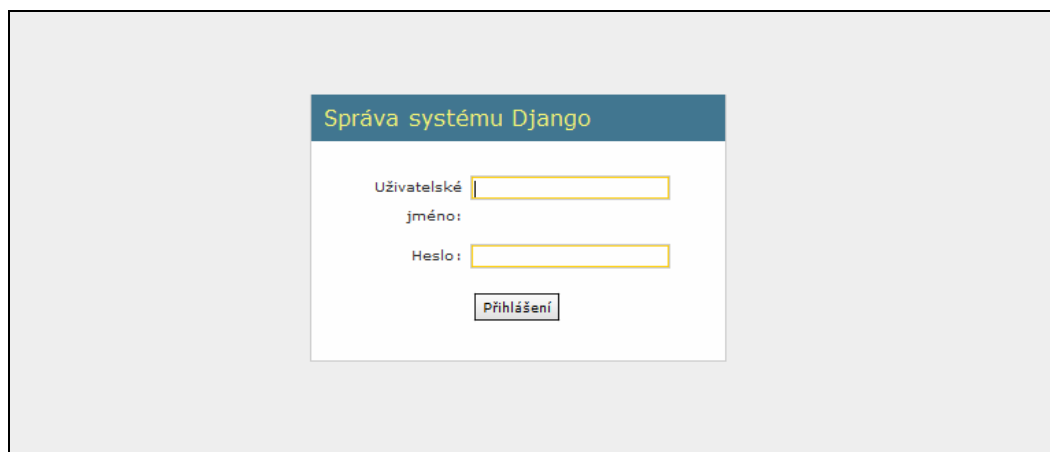


The screenshot shows the Django user management interface. At the top left is the Django logo. The main header is "Správa uživatelů - přehled". On the left side, there are links for "Fyzické osoby", "Právnícké osoby", and "Admin". The main content area displays the profile of "Petr Nový" with the following details:

Ulice:	Březolupská 837
Mesto:	Březolupy
Psc:	878 90
Stat:	CZ
Telefon:	789 789 890
Email:	petr@novy.cz
WWW stránky:	www.novy.cz

Obr. 8 - zobrazení vybraného uživatele v Django

Na obrázku níže je vidět přihlašovací formulář do administračního rozhraní v Django. Po vyplnění přihlašovacích údajů Uživatelského jména: pythondjango a Hesla: pythondjango se dostaneme do administrace.



The screenshot shows the Django login form titled "Správa systému Django". It contains two input fields: "Uživatelské jméno:" and "Heslo:". Below the fields is a "Přihlášení" button.

Obr. 9 - zobrazení přihlašovacího formuláře v Django

Po vstupu do administračního rozhraní uvidíme tuto úvodní obrazovku. Můžeme zde kompletně nastavovat administrační prostředí, jako třeba nastavení přihlašovacích údajů nebo přidávání skupin a uživatelů. Dále zde můžeme přidávat a editovat námi definované fyzické a právnické osoby a jejich kontakty a adresy.

Kvalitní tutoriál na zprovoznění webové prezentace i s administračním rozhraním najdete na odkaze: <http://zdrojak.root.cz/clanky/django-administrace/>.

The screenshot shows the Django administration interface. At the top, there is a header with the title 'Správa systému Django' and a user greeting 'Vítejte, uživateli **Uživatel**. Změnit heslo / Odhlásit se'. Below the header, the main content area is titled 'Správa webu'. It contains several sections, each with a list of items and two action buttons: '+ Přidat' and 'Změnit'. The sections are: 'Auth' (containing 'Skupiny' and 'Uživatelé'), 'Sites' (containing 'Weby'), and 'Sprava_Uzivatelu' (containing 'Adresy', 'Fyzické osoby', 'Kontakty', and 'Právnické osoby'). To the right of the main content, there is a box titled 'Poslední operace' which shows 'Vaše operace' and 'Nic'.

Obr. 10 - zobrazení administračního rozhraní v Django

Na obrázku níže vidíme přehled registrovaných osob po otevření odkazu Fyzické osoby. Můžeme zde přidávat, mazat nebo editovat záznamy.

Správa systému Django Vítejte, uživateli **Uživatel**. Změnit heslo / Odhlásit se

Domů > Sprava_uzivatelu > Fyzické osoby

Vyberte položku Fyzická osoba ke změně Fyzická osoba: přidat +

Operace: **Provést** Vybranych je 0 položek z celkem 18.

<input type="checkbox"/>	Přijmeni	Jmeno	Ic	Dic	Pohlavi
<input type="checkbox"/>	Pokorná	Jindřiška	7876878	CZ6787890	Žena
<input type="checkbox"/>	Doubrava	Petr	8789789	CZ9879897	Muž
<input type="checkbox"/>	Malá	Jarka	8978909	CZ9080908	Žena
<input type="checkbox"/>	Kovářová	Zuzana	878978	CZ897899	Žena
<input type="checkbox"/>	Donutil	Jan	7677389	CZ8789987	Muž
<input type="checkbox"/>	Dvořák	Tomáš	8798789	CZ897897	Muž
<input type="checkbox"/>	Mantl	Jan	7897997	CZ098098	Muž
<input type="checkbox"/>	Donutilová	Eva	354664	CZ7378378	Žena
<input type="checkbox"/>	Kovářová	Michaela	7837387	CZ8789709	Žena
<input type="checkbox"/>	Zamazal	Tomáš	38497873	CZ8798899	Muž
<input type="checkbox"/>	Pyšná	Lucie	3876746	CZ7867876	Žena
<input type="checkbox"/>	Kraus	Jan	3554887	CZ76876989	Muž
<input type="checkbox"/>	Holý	Tomáš	3637489	CZ0980998	Muž
<input type="checkbox"/>	Kadlčík	Jan	353637	CZ6383903	Muž
<input type="checkbox"/>	Král	David	8787898	CZ6565767	Muž
<input type="checkbox"/>	Králíková	Radka	78978908	CZ5675675	Žena
<input type="checkbox"/>	Malá	Jana	97897897	CZ8078709	Žena
<input type="checkbox"/>	Nový	Petr	78907789	CZ8790789	Muž

18 Fyzické osoby

Obr. 11 - zobrazení administračního rozhraní v Django - výpis uživatelů

Zde je vidět, jak vypadá přidání nové fyzické osoby do systému. Po přidání ji můžeme taktéž editovat nebo smazat.

Správa systému Django Vítejte, uživateli **Uživatel**. Změnit heslo / Odhlásit se

Domů > Sprava_uzivatelu > Fyzické osoby > Přidat Fyzická osoba

Fyzická osoba: přidat

Jmeno:

Prijmeni:

Ic:

Dic:

Pohlavi:

Ma adresu:

Výběr více než jedné položky je možný přidáním klávesy "Control" (nebo "Command" na Macu).

Ma kontakt:

Výběr více než jedné položky je možný přidáním klávesy "Control" (nebo "Command" na Macu).

Obr. 12 - zobrazení administračního rozhraní v Django - vytvoření uživatele

5.3 Ruby on Rails

Aplikace psaná v Ruby on Rails byla časově náročnější než ve frameworku Django z toho důvodu, že nenabízí možnost generování administračního rozhraní jako v Django. Rails ale nabízí zajímavou možnost, a to vygenerování obecné počáteční aplikace. Generování se provádí v příkazovém řádku systému Windows pomocí příkazu `rails new aplikace`, následným vkročením do nově vytvořené složky a následným zavoláním příkazu `rails generate scaffold Aplikace jmeno:string prijmeni:string popis:text`, která nám vytvoří základní šablonu webu (lešení) s funkčními formuláři (s parametry `jmeno`, `prijmeni`, `popis`) a s funkční komunikací s databází. Před prvním použitím databáze je ale nutno databázi synchronizovat pomocí příkazu `rake db:migrate`. Pro test byla použita verze Rubyinstaller 1.9.2 s rubygems 1.7.2.

5.3.1 Ukázka instalace a spuštění Ruby on Rails

Ke spuštění Ruby on Rails je nutné mít nainstalované Ruby a příslušné gemy. Dalším krokem je spuštění příkazového řádku (systém Windows) a ve složce, kde je nainstalované Ruby, vytvořit nový projekt:

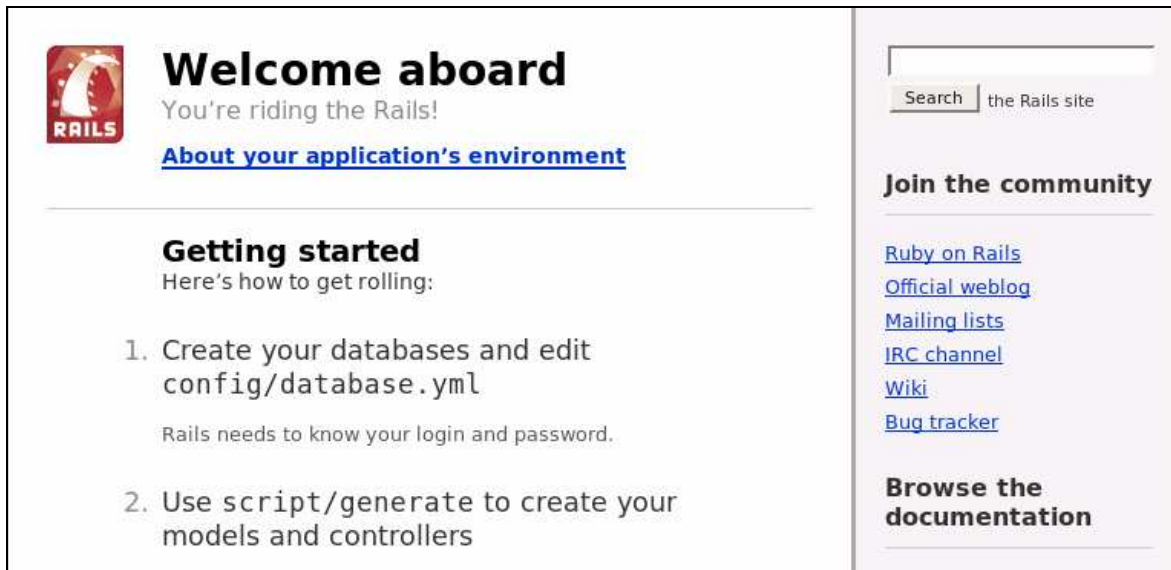
```
rails new novy_projekt
```

Pak už jen spustíme server pomocí příkazu:

```
rails server
```

Otevřeme webovou stránku:

```
http://127.0.0.1:3000/
```



Welcome aboard
You're riding the Rails!

[About your application's environment](#)

Getting started
Here's how to get rolling:

1. Create your databases and edit config/database.yml
Rails needs to know your login and password.
2. Use script/generate to create your models and controllers

Join the community

- [Ruby on Rails](#)
- [Official weblog](#)
- [Mailing lists](#)
- [IRC channel](#)
- [Wiki](#)
- [Bug tracker](#)

Browse the documentation

Obr. 13 - úvodní stránka frameworku Rails

Kompletní návod na instalaci Ruby a zprovoznění úvodní aplikace najdete na odkaze: http://guides.rubyonrails.org/getting_started.html.

5.3.2 Ukázka zdrojového kódu

```
1 class AdminController < ApplicationController
2   # GET /osobies
3   def index
4     @osobies = Osoby.all
5     respond_to do |format|
6       format.xml { render :xml => @osobies }
7     end
8   end
end
```


5.3.3 Vytvořená aplikace - správa uživatelů

Po spuštění vytvořené aplikace na Ruby serveru se na adrese **http://127.0.0.1:3000/** zobrazí úvodní webová prezentace s přehledem všech zaregistrovaných uživatelů s jejich nastavenými údaji. Lze volit mezi fyzickými nebo mezi právníckými osobami. Pro zobrazení vyplněných údajů je potřeba kliknout na jméno uživatele. Pokud chceme

editovat nebo přidávat jednotlivé osoby, musíme přejít do administračního rozhraní. Stačí kliknout na odkaz Admin.

Menu

[Fyzické osoby](#)
[Právnícké osoby](#)
[Admin](#)



Správa uživatelů

Fyzické osoby - přehled

Jméno	IČ	DIČ	Pohlaví
Petr Nový	78907789	CZ8790789	Muž
Jana Malá	97897897	CZ8078709	Žena
Radka Králíková	78978908	CZ5675675	Žena
David Král	87879898	CZ6565767	Muž
Jan Kadlčík	353637	CZ6383903	Muž
Tomáš Holý	3637489	CZ0980998	Muž
Jan Kraus	3554887	CZ76876989	Muž
Lucie Pyšná	3876746	CZ7867876	Žena
Tomáš Zamazal	38497873	CZ8798899	Muž
Michaela Koválová	7837387	CZ8789709	Žena
Eva Donutilová	354664	CZ7378378	Žena
Jan Mantl	7897997	CZ098098	Muž
Tomáš Dvořák	8798789	CZ897897	Muž
Jan Donutil	7677389	CZ8789987	Muž
Zuzana Kovářová	878978	CZ897899	Žena
Jarka Malá	8978909	CZ9080908	Žena
Petr Doubrava	8789789	CZ9879897	Muž
Jindřiška Pokorná	7876878	CZ8787890	Žena

Obr. 14 - úvodní stránka vytvořené aplikace „Správa uživatelů“ v Rails



The screenshot shows a web application interface for user management. On the left is a sidebar menu with the following items: "Menu", "Fyzické osoby", "Právnícké osoby", and "Admin". The main content area is titled "Správa uživatelů" and features a Rails logo. Below the title is a section "Fyzické osoby - zobraz" displaying the details of a selected user:

- Jméno:** Petr
- Příjmení:** Nový

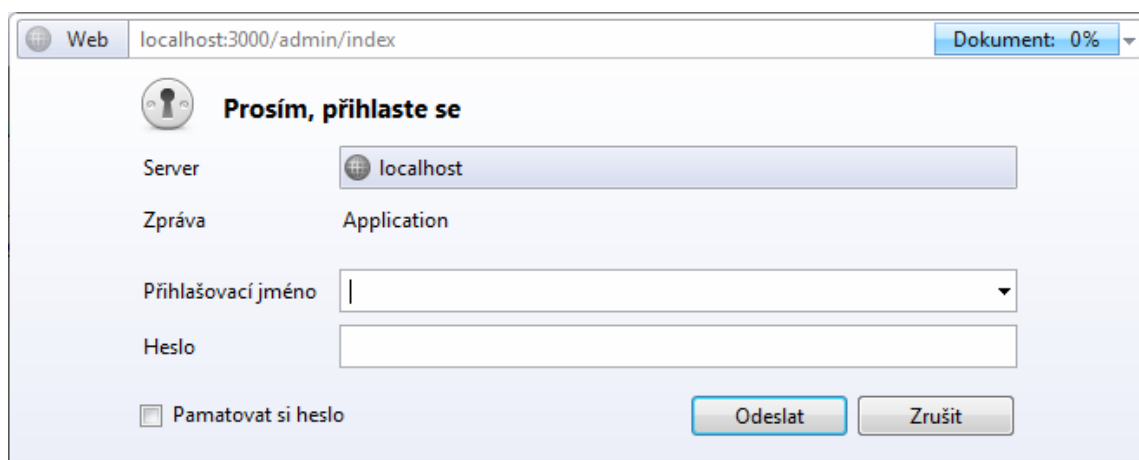
- Kontakt na osobu**
 - Telefon:** 789 789 890
 - Email:** petr@novy.cz
 - www stránky:** www.novy.cz

- Adresa osoby**
 - Ulice:** Březolupská 837
 - Město:** Březolupy
 - PSČ:** 878 90
 - Stát:** CZ

At the bottom of the details section is a link labeled "Zpět".

Obr. 15 - zobrazení vybraného uživatele v Rails

Na obrázku níže je vidět přihlašovací formulář do administračního rozhraní napsaného v Ruby on Rails . Po vyplnění přihlašovacích údajů Přihlašovacího jména: rubyonrails a Hesla: rubyonrails se dostaneme do administrace.



The screenshot shows a web browser window displaying a login form. The address bar shows "localhost:3000/admin/index" and the document is at 0% zoom. The form is titled "Prosím, přihlaste se" and includes the following fields and controls:

- Server:** localhost
- Zpráva:** Application
- Přihlašovací jméno:** (empty text input field)
- Heslo:** (empty password input field)
- Pamatovat si heslo
- Odeslat** (submit button)
- Zrušit** (cancel button)

Obr. 16 - zobrazení přihlašovacího formuláře v Rails

Po vstupu do administračního rozhraní uvidíme tuto úvodní obrazovku. Jsou zde vypsány jednotlivé osoby a jejich přiřazené údaje. Červený odkaz znamená, že osoba nemá údaj vyplněný a zelený znamená, že osoba má údaj vyplněný. Dále zde můžeme přidávat a editovat námi definované fyzické a právnické osoby.

Menu

[Fyzické osoby](#)

[Právnické osoby](#)



Správa uživatelů

Právnické osoby - přehled

Název	IČ	DIČ	
ČEZ, s.r.o.	7876786	CZ7868789	Adresa Kontakt Spojeni  
KONY	4878347	CZ8749834	Adresa Kontakt Spojeni  
SONY	847 834 928	CZ7689493	Adresa Kontakt Spojeni  
HP tronic	787 873 938	CZ78709309	Adresa Kontakt Spojeni  
Peugeot	877 878 989	CZ8989800	Adresa Kontakt Spojeni  
SEO	787 987 088	CZ09889080	Adresa Kontakt Spojeni  
Press, s.r.o.	789 789 897	CZ878987	Adresa Kontakt Spojeni  
Bike, s.r.o.	9384989	CZ93898893	Adresa Kontakt Spojeni  
Enzo	7837638	CZ98897987	Adresa Kontakt Spojeni  
Oskar, s.r.o.	78736628	CZ87897889	Adresa Kontakt Spojeni  
Zend	738838727	CZ8978798	Adresa Kontakt Spojeni  
Tecko, s.r.o.	7376836	CZ87373	Adresa Kontakt Spojeni  
RUBY, s.r.o.	8379483	CZ873783	Adresa Kontakt Spojeni  
CSOB	7738837	CZ8373790	Adresa Kontakt Spojeni  
COFIDUS, s.r.o.	737483878	CZ83738920	Adresa Kontakt Spojeni  
Aukro cz	7363838	CZ8373873	Adresa Kontakt Spojeni  
Stab, a.s.	7837398	CZ8979879	Adresa Kontakt Spojeni  
Geopard, a.s.	7376389	CZ8937989	Adresa Kontakt Spojeni  

[Vytvoř novou osobu](#)

Obr. 17 - zobrazení administračního rozhraní v Rails

Na obrázku níže vidíme vytvoření nové právnické osoby.



The screenshot shows a web application interface for user management. On the left is a sidebar menu with the title 'Menu' and two items: 'Fyzické osoby' and 'Právnické osoby'. The main content area is titled 'Správa uživatelů' and features a 'RAILS' logo. Below the logo, the heading 'Nová právnická osoba' is displayed. There are three text input fields labeled 'Název', 'IČ', and 'DiČ'. Below these fields is a button labeled 'Create Pravnice osoby' and a link labeled 'Zpět'.

Obr. 18 - zobrazení administračního rozhraní v Rails - vytvoření uživatele

5.4 Zend framework

Tvorba aplikace „Správa uživatelů“ patřila v Zendu mezi nejpomalejší. Zend totiž nenabízí žádné základní lešení webu jako třeba Rails nebo Django. Zend jen umožňuje vytvoření aplikace v příkazové řádce systému Windows s rozložením všech složek podle architektury MVC příkazem `zf create project nase_aplikace`. Pro založení nového projektu můžeme také využít funkci **Zend Studio**, které dokáže také založit nový projekt se správnou hierarchií složek.

Velkou předností Zend frameworku je to, že stojí na dobře známém programovacím jazyku PHP. Jazyk PHP je velmi rozšířen ve webových aplikacích, proto kdo už programoval jakékoliv webové aplikace v PHP, určitě si poradí i se Zend frameworkem. Pro test byl využit framework ve verzi 1.11.3.

5.4.1 Ukázka instalace a spuštění Zend frameworku

Pro založení a spuštění projektu na frameworku Zend je nejlépe využít přímo nabízené programy **Zend Studio** a **Zend Server**. Odpadá tedy práce s příkazovým řádkem. Stačí si

tyto programy pouze nainstalovat a v **Zend Studiu** založit nový projekt, který se v něm také za pomoci **Zend Serveru** spustí jako webová prezentace.



Obr. 19 - úvodní stránka frameworku Zend

Podrobný návod na instalaci a vytvoření úvodní aplikace najdete na odkaze: <http://framework.zend.com/manual/en/learning.quickstart.create-project.html>.

5.4.2 Ukázka zdrojového kódu

```
1 public function pravnickeosobynewAction()
2 {
3     $osoby = new Osoby();
4     $osobyList = $osoby->Vypisosoby();
5     $this->osobyList = $osobyList;
6     $form = $this->getpravnickeForm();
7     $this->view->form = $form;
8 }
```

5.4.3 Vytvořená aplikace - správa uživatelů

Po spuštění vytvořené aplikace na serveru se na adrese **http://127.0.0.1/** zobrazí úvodní webová prezentace s přehledem všech zaregistrovaných uživatelů s jejich nastavenými údaji. Lze volit mezi fyzickými nebo mezi právnickými osobami. Pro zobrazení vyplněných údajů je potřeba kliknout na odkaz jméno osoby. Pokud chceme editovat nebo přidávat jednotlivé osoby, musíme přejít do administračního rozhraní. Stačí kliknout na odkaz Admin.

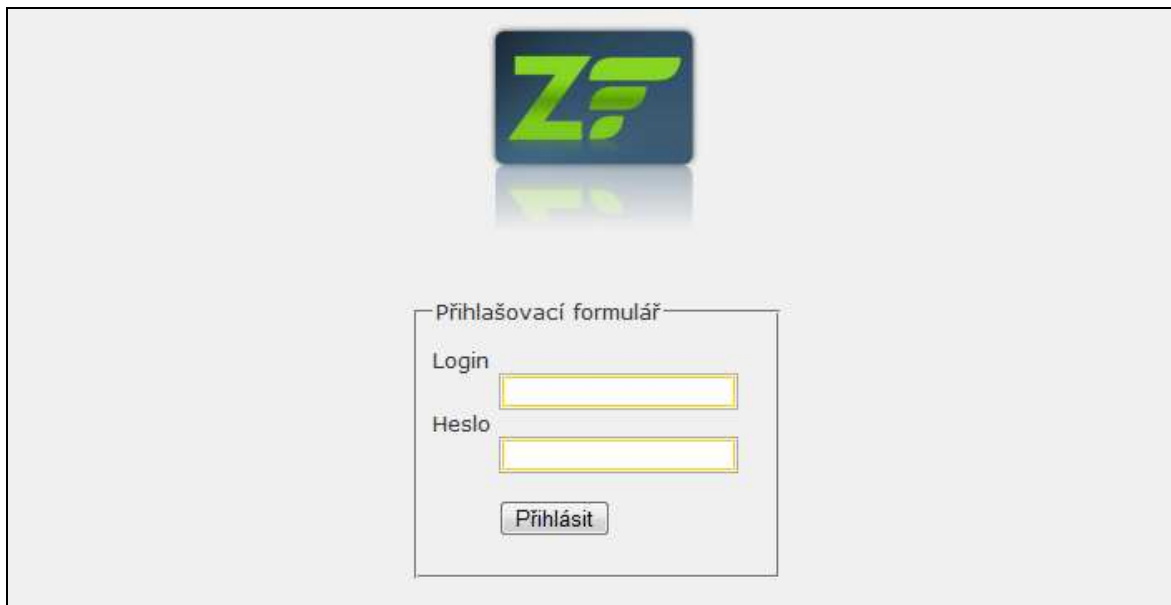
Přehled osob - fyzické osoby				
Menu Fyzické osoby Právnické osoby Admin	Fyzická osoba	IČ	DIČ	Pohlaví
	Petr Nový	78907789	CZ8790789	Muž
	Jana Malá	97897897	CZ8078709	Žena
	Radka Králiková	78978908	CZ5675675	Žena
	David Král	87879898	CZ6565767	Muž
	Jan Kadlčík	353637	CZ6383903	Muž
	Tomáš Holý	3637489	CZ0980998	Muž
	Jan Kraus	3554887	CZ76876989	Muž
	Lucie Pyšná	3876746	CZ7867876	Žena
	Tomáš Zamazal	38497873	CZ8798899	Muž
	Michaela Kovalová	7837387	CZ8789709	Žena
	Eva Donutilová	354664	CZ7378378	Žena
	Jan Mantl	7897997	CZ098098	Muž
	Tomáš Dvořák	8798789	CZ897897	Muž
	Jan Donutil	7677389	CZ8789987	Muž
	Zuzana Kovářová	878978	CZ897899	Žena
	Jarka Malá	8978909	CZ9080908	Žena
	Petr Doubrava	8789789	CZ9879897	Muž
	Jindřiška Pokorná	7876878	CZ8787890	Žena

Obr. 20 - úvodní stránka vytvořené aplikace „Správa uživatelů“ v Zendu

Registrované osoby - fyzické																	
Menu Zpět k výběru	<table border="0"> <tr> <td>Tomáš Holý</td> <td>detail účtu</td> </tr> <tr> <td>Telefon:</td> <td>734 635 245</td> </tr> <tr> <td>Email:</td> <td>holy@tomas.cz</td> </tr> <tr> <td>Internetové stránky:</td> <td>www.holytomas.cz</td> </tr> <tr> <td>Ulice:</td> <td>Trutnov 987</td> </tr> <tr> <td>Město:</td> <td>Trutnov</td> </tr> <tr> <td>PSČ:</td> <td>848 43</td> </tr> <tr> <td>Stát:</td> <td>CZ</td> </tr> </table>	Tomáš Holý	detail účtu	Telefon:	734 635 245	Email:	holy@tomas.cz	Internetové stránky:	www.holytomas.cz	Ulice:	Trutnov 987	Město:	Trutnov	PSČ:	848 43	Stát:	CZ
Tomáš Holý	detail účtu																
Telefon:	734 635 245																
Email:	holy@tomas.cz																
Internetové stránky:	www.holytomas.cz																
Ulice:	Trutnov 987																
Město:	Trutnov																
PSČ:	848 43																
Stát:	CZ																

Obr. 21 - zobrazení vybraného uživatele v Zendu

Na obrázku níže je vidět přihlašovací formulář do administračního rozhraní napsaného v Zendu. Po vyplnění přihlašovacích údajů Login: phpzend a Hesla: phpzend se dostaneme do administrace.



The image shows a login form titled "Přihlašovací formulář" (Login form) centered on a light gray background. At the top center is the Zend logo, a blue square with a green stylized 'Z'. Below the logo is a white rectangular box containing the form fields. The form has two input fields: "Login" and "Heslo" (Password). Below these fields is a button labeled "Přihlásit" (Login). The form is styled with a simple, clean design typical of early web applications.

Obr. 22 - zobrazení přihlašovacího formuláře v Zendu

Po vstupu do administračního rozhraní uvidíme tuto úvodní obrazovku. Jsou zde vypsány jednotlivé osoby a jejich přiřazené údaje. Červený křížek znamená, že osoba nemá údaj vyplněný a zelená fajfka znamená, že osoba má údaj vyplněný. Dále zde můžeme přidávat a editovat námi definované fyzické a právnické osoby.

Administrace osob - fyzické osoby					
Menu					
Fyzické osoby					
Právnické osoby					
Odhlásit					
Fyzická osoba	Má kontakt	Má adresu	Je i právnická osoba	Edit	Del
Petr Nový	✓	✓	✓		
Jana Malá	✗	✓	✗		
Radka Králíková	✓	✓	✗		
David Král	✗	✓	✗		
Jan Kadlčík	✗	✓	✓		
Tomáš Holý	✓	✓	✓		
Jan Kraus	✓	✗	✗		
Lucie Pyšná	✓	✗	✗		
Tomáš Zamazal	✗	✓	✗		
Michaela Koválová	✗	✓	✓		
Eva Donutilová	✓	✗	✗		
Jan Mantl	✓	✗	✗		
Tomáš Dvořák	✗	✓	✗		
Jan Donutil	✗	✗	✓		
Zuzana Kovářová	✓	✗	✓		
Jarka Malá	✗	✓	✗		
Petr Doubrava	✓	✓	✓		
Jindřiška Pokorná	✗	✗	✗		

Obr. 23 - zobrazení administračního rozhraní v Zendu

Na obrázku níže vidíme editaci fyzické osoby.

The screenshot shows a web interface titled "Registrované osoby - fyzické". On the left is a "Menu" with a link "Zpět k výběru". The main content area is titled "Tomáš Dvořák - editace uživatele". It contains a "Kontakty:" label with a green "doplnit" link. Below is a form with a tabbed interface. The "Adresa" tab is active, showing input fields for "Ulice" (Dvořákova 1829), "Mesto" (Ostrava), "PSČ" (789 79), and "Stát" (CZ). The "Zpracování" tab is also visible, containing an "Edituj uživatele" button.

Obr. 24 - zobrazení administračního rozhraní v Zendu - editace uživatele

5.5 Realizace časového/paměťového testu - JMeter

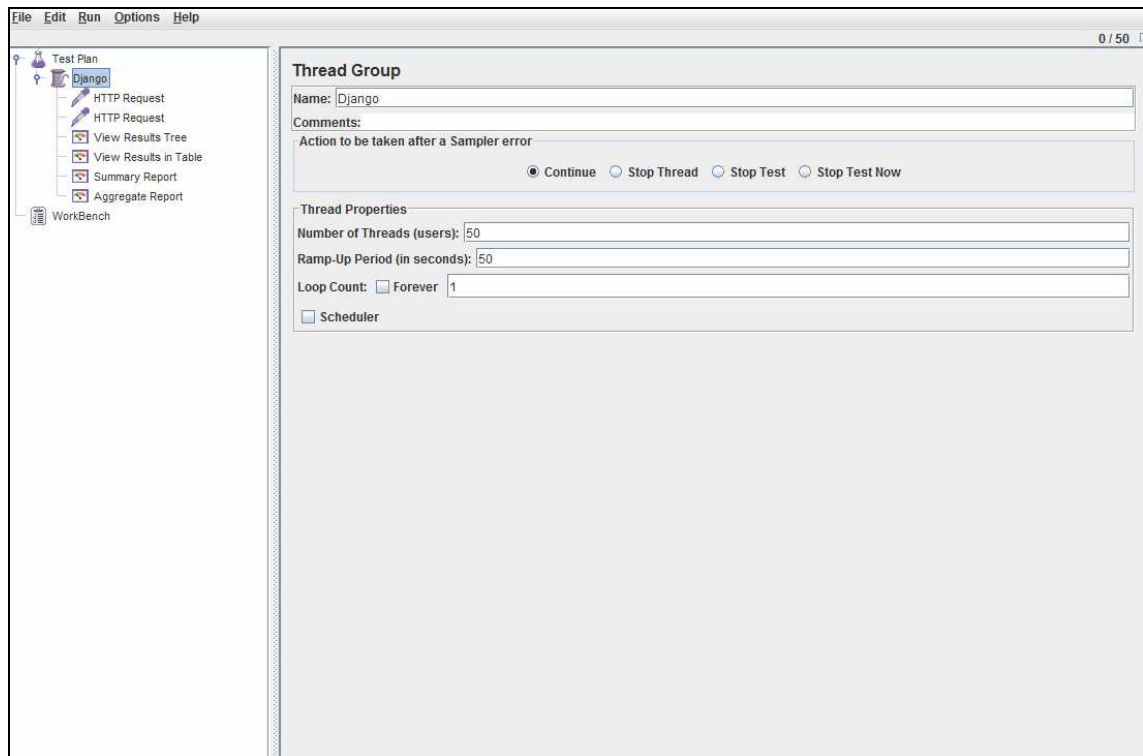
Pro měření časových a paměťových nároků je použit JMeter ve verzi 2.4, který lze stáhnout z oficiálních stránek http://jakarta.apache.org/site/downloads/downloads_jmeter.cgi. Pro správný běh programu je nutné mít nainstalovanou Javu (Java Virtual Machine) ve verzi minimálně 1.5.

5.5.1 Popis nastavení

Na obrázku níže je vidět základní nastavení JMetru pro měření doby zobrazení všech zaregistrovaných uživatelů v jednotlivých frameworkích. Vlevo vidíme testovací plán a vpravo jeho nastavení.

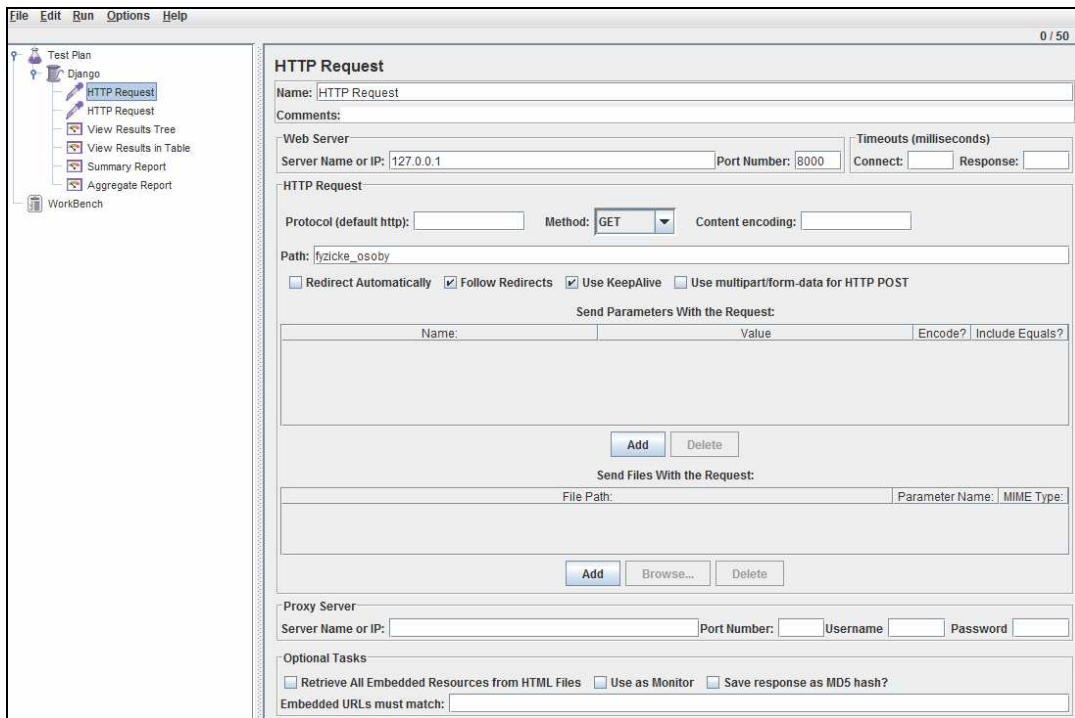
- Number of Threads (uživatelé) - zde se nastavuje počet uživatelů, kteří budou k danému testu přistupovat. Pro test bylo zvoleno 50 uživatelů.

- Ramp up Period (v sekundách) - označuje čas, za jak dlouho dojde k vytvoření nastaveného počtu uživatelů. Čas byl nastaven na 50 sekund, což znamená, že každou sekundu se vytvořil nový uživatel.

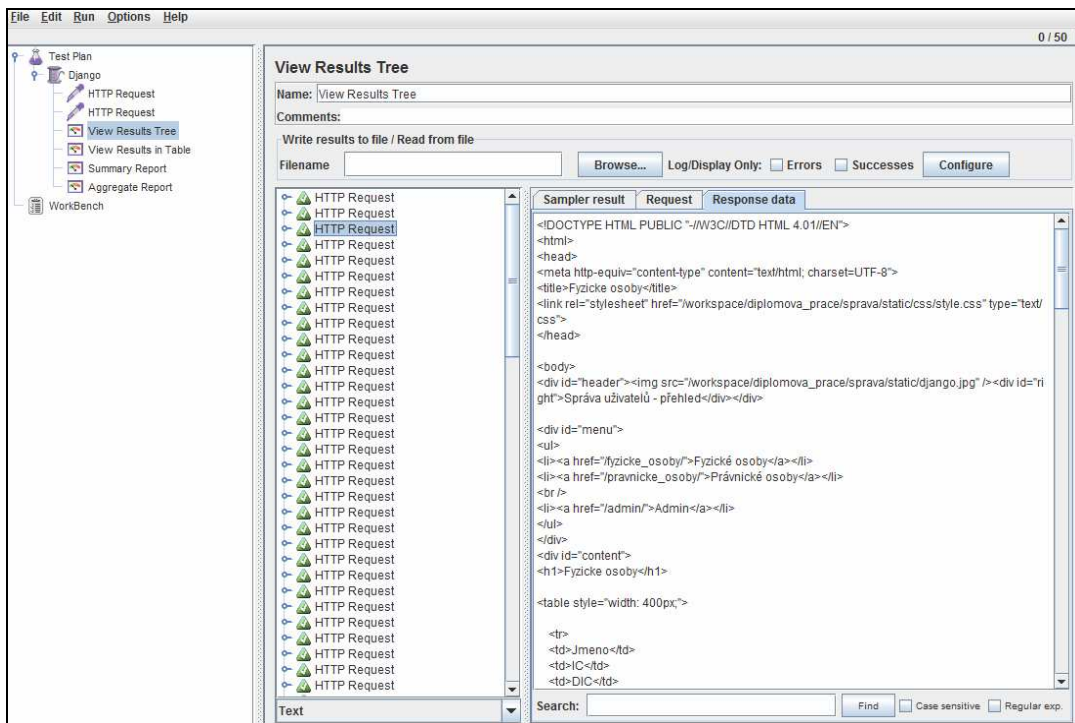


Obr. 25 - nastavení aplikace JMeter

Na obrázku níže vidíme zadanou URL adresu testované stránky. Pokud testujeme ještě formuláře, vyplní se jeho parametry, tj. proměnná „Name“ a její hodnota „Value“.

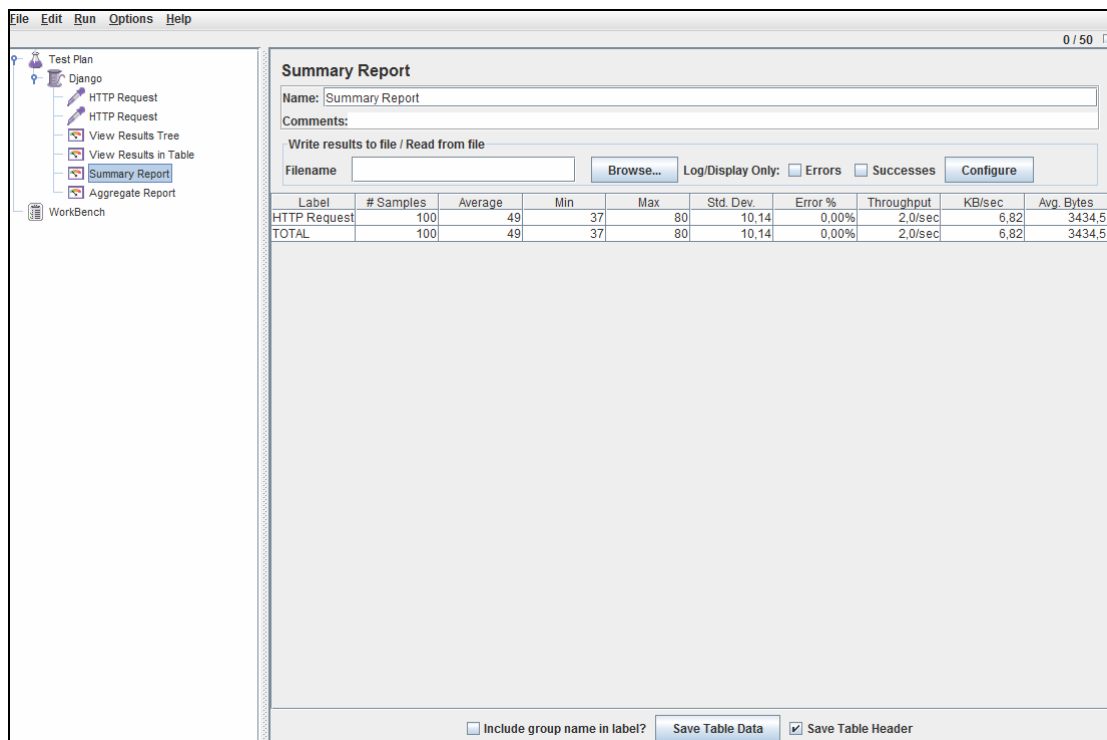


Obr. 26 - nastavení aplikace JMeter



Obr. 27 - zobrazení odesílaných dat aplikací JMeter

Na závěr vidíme průměrné výsledky pro testovanou stránku. Jsou zde hodnoty jako průměr, minimální a maximální hodnota, počet chyb, propustnost a využití paměti.



The screenshot shows the JMeter Summary Report window. The left sidebar displays a test plan for 'Django' with two 'HTTP Request' elements, 'View Results Tree', 'Summary Report', and 'Aggregate Report'. The main window displays the following data:

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
HTTP Request	100	49	37	80	10.14	0.00%	2.0/sec	6.82	3434.5
TOTAL	100	49	37	80	10.14	0.00%	2.0/sec	6.82	3434.5

At the bottom of the window, there are checkboxes for 'Include group name in label?' (unchecked), 'Save Table Data' (checked), and 'Save Table Header' (checked).

Obr. 28 - zobrazení výsledků v aplikaci JMeter

6 TESTOVÁNÍ MVC FRAMEWORKŮ

Testy probíhaly tak, že se zadaný úkol provedl na všech frameworkích zvlášť. Chtěl bych podotknout, že testy probíhaly na stroji s určitým výkonem (odstavec 4) a výsledky se na různých strojích mohou lišit. Doufám, že testy budou objektivní, protože při tvoření aplikací jsem narazil na různé problémy, které se v každém frameworku řešily jinak. Proto je důležité při testování různých frameworků mít přehled, co se jak tvoří a neřešit vzniklé problémy přímo při vývoji aplikace.

Vyzdvihl bych to, že různé frameworky, jako např. Django, si v databázi vytvoří své tabulky, takže dodržet původní koncepci zadané databáze bylo obtížné.

Naproti tomu v Rails došlo k automatickému přejmenování tabulek z českých slov na množná čísla anglického tvaru, takže např. tabulka **osoby** přešla v tabulku **osobies**.

6.1 Django, Python

6.1.1 GUI komponenty

Jako GUI komponenty jsou chápány především gridy a záložky. Pro systém Django existuje spousta externích gridů a záložek. Mezi větší projekty patří např. grid s názvem **DhtmlxGrid**, **Dojo DataGrid**, **Flexigrid** nebo třeba **jQuery Grid**. Většina gridů a záložek jsou napsány v javascriptu. Je tedy nutné nakopírovat požadované soubory do vytvářené aplikace a přistupovat k daným funkcím. Bohužel i u velkých projektů jsem se setkal s nepřesnou nebo zastaralou dokumentací. U aplikace „Správa uživatelů“ napsané v Django je použit grid, který přímo nabízí administrační rozhraní.

Správa systému Django Vítejte, uživatel **Uživatel**. Změnit heslo / Odhlásit se

Domů > Správa_uzivatelu > Fyzické osoby

Vyberte položku Fyzická osoba ke změně Fyzická osoba: přidat +

Operace: ----- **Provést** Vybraných je 0 položek z celkem 18.

<input type="checkbox"/>	Příjmení	Jmeno	Ic	Dic	Pohlavi
<input type="checkbox"/>	Pokorná	Jindřiška	7876878	CZ8787890	Žena
<input type="checkbox"/>	Doubrava	Petr	8789789	CZ9879897	Muž
<input type="checkbox"/>	Malá	Jarka	8978909	CZ9080908	Žena
<input type="checkbox"/>	Kovářová	Zuzana	878978	CZ897899	Žena
<input type="checkbox"/>	Donutil	Jan	7677389	CZ8789987	Muž
<input type="checkbox"/>	Dvořák	Tomáš	8798789	CZ897897	Muž
<input type="checkbox"/>	Mantl	Jan	7897997	CZ098098	Muž
<input type="checkbox"/>	Donutilová	Eva	354664	CZ7378378	Žena
<input type="checkbox"/>	Koválová	Michaela	7837387	CZ8789709	Žena
<input type="checkbox"/>	Zamazal	Tomáš	38497873	CZ8798899	Muž
<input type="checkbox"/>	Pyšná	Lucie	3876746	CZ7867876	Žena
<input type="checkbox"/>	Kraus	Jan	3554887	CZ76876989	Muž
<input type="checkbox"/>	Holý	Tomáš	3637489	CZ0980998	Muž
<input type="checkbox"/>	Kadlčík	Jan	353637	CZ6383903	Muž
<input type="checkbox"/>	Král	David	87879898	CZ6565767	Muž
<input type="checkbox"/>	Králíková	Radka	78978908	CZ5675675	Žena
<input type="checkbox"/>	Malá	Jana	97897897	CZ8078709	Žena
<input type="checkbox"/>	Nový	Petr	78907789	CZ8790789	Muž

18 Fyzické osoby

Obr. 29 - grid pro Django

6.1.2 Rychlost vývoje aplikace

Tvorba webové prezentace „Správa uživatelů“ ve frameworku Django patřila mezi nejrychleji vytvořenou aplikaci ze zadaných frameworků. Django totiž nabízí možnost vygenerovat kompletní administrační rozhraní, které ušetří spoustu práce. Velkou mírou k tomu také přispělo velké množství kvalitních českých tutoriálů zmíněných např. v odstavci 5.2.1.

6.1.3 Bezpečnost - SQL injection, XSS útoky...

SQL injection je technika napadení databázové vrstvy programu vsunutím (odtud „injection“) kódu přes neošetřený vstup a vykonání vlastního, samozřejmě pozměněného SQL dotazu. Toto nechtěné chování vzniká při propojení aplikační vrstvy s databázovou vrstvou (téměř vždy se totiž jedná o dva různé programy) a zabraňuje se mu pomocí jednoduchého escapování potenciálně nebezpečných znaků nebo využívání moderních frameworků. [8]

Zde je popsána ukázka **SQL útoku** na přihlašovací formuláři s buňkami **jméno** a **heslo**. Zadejme tedy správné přihlašovací údaje, a to **jméno**: django a **heslo**: admindjango.

Provede se následující příkaz SQL, který vrátí počet zaregistrovaných uživatelů s těmito zadanými údaji. Jedná se tedy o korektní přihlášení.

```
select count(*) from uzivatele where jmeno='django' and heslo='admindjango'
```

Pokud ale zadáme místo správných přihlašovacích údajů **jméno**: django a **heslo** '**or 1=1** – provede se následující SQL dotaz, který nám taktéž vrátí počet zaregistrovaných uživatelů se jménem django. Jedná se tedy o nekorektní přihlášení, protože jsme se přihlásili bez správného hesla.

```
select count(*) uzivatele where jmeno='django' and heslo='' or 1=1 --'
```

Moderní frameworky jako Django však řeší tento problém za uživatele tzv. „zneškodněním“ (escape) nebezpečných znaků. Pokud využíváme daných funkcí API (application programming interface) pro komunikaci s databází, není se čeho obávat.

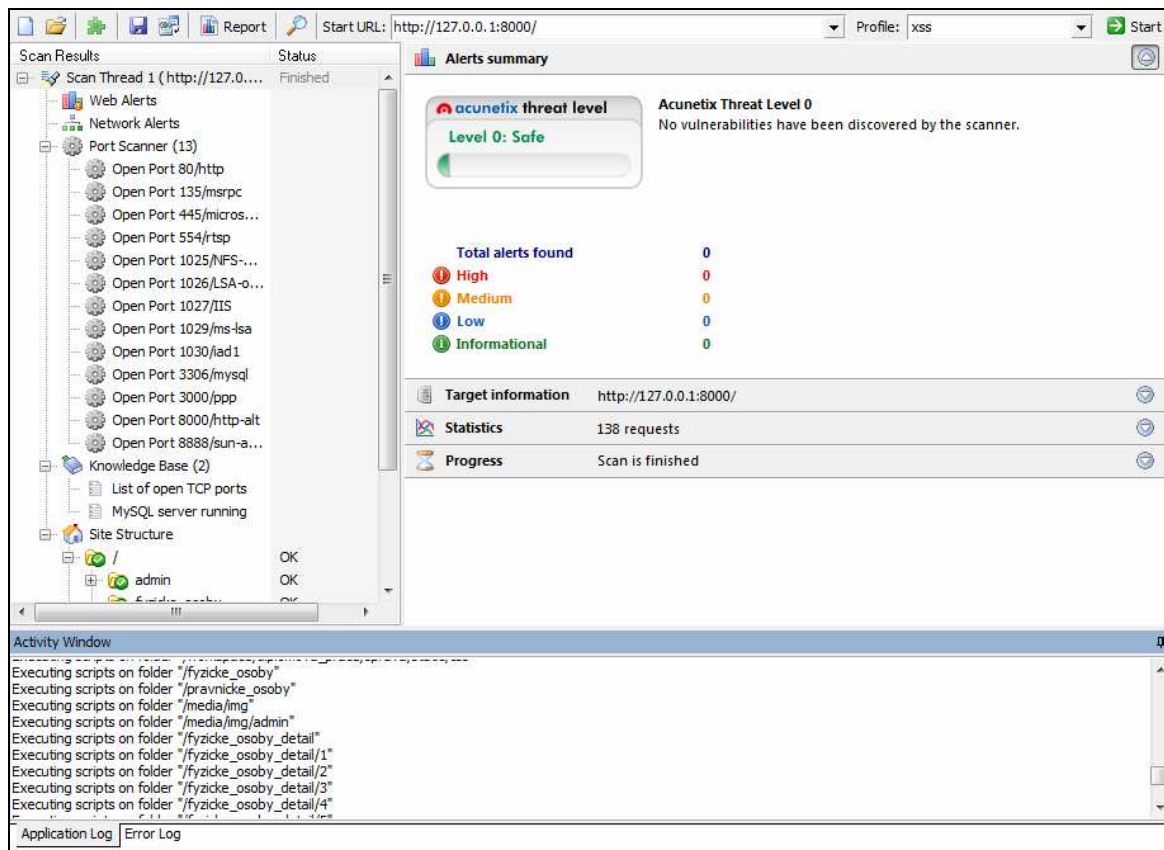
XSS útoky (Cross-site scripting) jsou dalším rozšířeným útokem na webové aplikace. Ačkoliv o něm již bylo napsáno mnoho, jsou **XSS útoky** i nadále jedny z nejčastějších způsobů napadení současných webových aplikací.

Zde je popsána jednoduchá ukázka **XSS útoku** přes URL stránky. Mějme v libovolné šabloně kód `<h1>Ahoj, {{ jmeno }}!</h1>` a pokud zadáme URL adresu ve tvaru <http://example.com/hello/jmeno=Tome>, vypíše se nám na dané stránce hláška `<h1>Ahoj, Tome!</h1>`.

Pokud ale zadáme uživatelem pozměněnou URL adresu <http://example.com/hello/jmeno=<i>Tome</i>>, vypíše se nám na dané stránce hláška `<h1>Ahoj, Tome!</h1>` napsána kurzívou. Tento nenápadný zásah nám otevírá bránu k daleko zákeřnějším útokům např. pomocí tagu `<script>`, který podstrčí stránce JavaScriptový kód.

Funkční obranou u frameworku Django je vypisování dané hlášky v šabloně tímto zápisem `<h1>Ahoj, {{ jmeno|escape }}!</h1>`.

XSS útoky provedené na aplikaci napsanou v Django byly otestovány pomocí programu **Acunetix Web Vulnerability Scanner**, který otestuje různé druhy útoků. Na obrázku níže je vidět výsledek testu s nulovými výsledky napadení.



Obr. 30 - testování SQL a XSS útoků na aplikaci napsanou v Django

Existují i další útoky, jako například **Cross-site Request Forgery (CSRF)** nebo **E-mail header injection**.

6.1.4 Víceúrovňová autorizace

Django využívá tzv. trojrozměrnou správu přístupových práv, která kombinuje role, zdroje a jejich práva. Obecné vysvětlení těchto pojmů je níže.

- Role - reprezentují jednotlivé registrované uživatele a skupiny. Role určuje, jaký typ uživatele je přihlášen. Může se jednat o Administrátora nebo o obyčejného registrovaného uživatele webu. K identifikaci se obvykle využívá uživatelské jméno nebo ID.

- Zdroje - představují prostředky, které jsou využívány rolemi. Zdroje mohou prezentovat například články, příspěvky, menu atd. Administrátor pak jednotlivým uživatelům (rolím) nastaví, do kterých částí má uživatel přístup povolen a do kterých ne.
- Práva - jsou přidělené operace, na které má uživatel (role) právo. Na běžném webu to mohou být práva typu vkládání příspěvků, editace příspěvků, mazání nebo třeba hlasování v anketách.

ACL (access control list) knihovny pro Django:

V Django jako i v dalších frameworkcích existují knihovny na **autentizaci** (někdy psáno též jako autentikace, případně autentifikace), což je proces pro ověření identity uživatele. Pomocí **autentizace** tedy zjistíme, jaký typ uživatele je právě přihlášen. **Autorizace** se zase stará o kontrolu oprávnění, tedy aby přístup k aplikaci měli jenom oprávnění uživatelé.

Autentizace a autorizace v Django je vytvářena pomocí modulu `django.contrib.auth`. Hlavní součástí je model `User`, který představuje přihlášeného uživatele. Nepřihlášení uživatelé jsou reprezentováni odvozeným modelem `AnonymousUser`. Ať už je uživatel přihlášen nebo ne, můžeme k jeho modelu přistupovat v pohledech pomocí objektu `request.user` a v šablonách přes proměnnou `{{user}}`. Instance modelu uživatele obsahuje několik zajímavých atributů:

- `user.username`: uživatelské jméno
- `user.first_name`: křestní jméno
- `user.last_name`: příjmení
- `user.email`: e-mail
- `user.is_staff`: příznak, zda má uživatel povolený vstup do administrace
- `user.is_active`: příznak, zda je uživatel aktivní

Kromě atributů máme k dispozici několik důležitých metod:

- `user.is_authenticated()`: zkontroluje, zda je uživatel přihlášen

- `user.get_full_name()`: celé jméno uživatele
- `user.set_password()`: nastaví uživatelské heslo
- `user.check_password()`: ověří, jestli se zadané heslo shoduje s heslem uživatele

Popis mnoha dalších atributů a metod lze najít v dokumentaci např. zde: <http://docs.djangoproject.com/en/dev/topics/auth/#django.contrib.auth.models.User>.

Protože je uživatel reprezentován modelem, zacházíme s ním podobně jako s ostatními databázovými modely. Můžeme přidávat nové uživatelské účty, odstraňovat je a upravovat uživatelské údaje. Vytváření nového uživatelského účtu se vytváří pomocí funkce `create_user`. Celý příkaz vypadá takto:

```
user=User.objects.create_user('pavel', 'pavel@seznam.cz', 'pavlovoheslo')
```

Hesla se neukládají do databáze přímo, ale ukládají se tam pouze jejich otisky. Pro změnu nebo ověření hesla musíme použít metodu `set_password` nebo `check_password`.

Přihlášení a odhlášení uživatele se dá ověřit pomocí funkce `authenticate`, která vrátí objekt uživatele v případě úspěšného ověření, nebo prázdný objekt, pokud se ověření nezdaří. Poté uživatele můžeme přihlásit pomocí metody `login` a kdykoliv později odhlásit pomocí metody `logout`.

V aplikaci „Správa uživatelů“ je těchto modulů využito v přihlašování a odhlašování do administračního rozhraní.

<input type="checkbox"/>	Uživatelské jméno ▾	E-mailová adresa	Křestní jméno	Příjmení	Administrační přístup
<input type="checkbox"/>	pythondjango	python@django.cz	Uživatel	UTB	✓
<input type="checkbox"/>	user		Uživatel	UTB	✖

Obr. 31 - nastavení přístupu do administrace

Správa systému Django Vítejte, uživateli **Uživatel**. Změnit heslo / Odhlásit se

Domů > Auth > Uživatelé > pythondjango

uživatel: změnit Historie Zobrazení na webu →

Uživatelské jméno:
Požadováno. 30 znaků nebo méně. Pouze písmena, číslice a znaky @/./+/_/.

Heslo:
Použijte buď tvar "[algo]\${salt}\${hexdigest}" nebo formulář pro změnu hesla.

Osobní údaje

Křestní jméno:

Příjmení:

E-mailová adresa:

Obr. 32 - nastavení údajů registrovaného uživatele

Oprávnění

Aktivní
Určuje, zda bude uživatel považován za aktivního. Použijte tuto možnost místo odstranění účtů.

Administráční přístup
Určuje, zda se uživatel může přihlásit do správy tohoto webu.

Superuživatel
Určuje, že uživatel má veškerá oprávnění bez jejich explicitního přiřazení.

Uživatelská oprávnění:

Dostupné položky: uživatelská oprávnění

- admin | log entry | Can add log entry
- admin | log entry | Can change log entry
- admin | log entry | Can delete log entry
- auth | group | Can add group
- auth | group | Can change group
- auth | group | Can delete group
- auth | message | Can add message
- auth | message | Can change message
- auth | message | Can delete message
- auth | permission | Can add permission
- auth | permission | Can change permission
- auth | permission | Can delete permission

Vybrat vše

Vybrané položky: uživatelská oprávnění

Vyberte si a klikněte +

Odebrat vše

Důležitá data

Poslední přihlášení: Datum: Dnes | Nyní |

Datum registrace: Datum: Dnes | Nyní |

Obr. 33 - nastavení uživatelského oprávnění v administračním rozhraní

K dispozici jsou také externí pluginy na autentizaci a autorizaci napsaných pro Django. Nabízí se například plugin **Django-authority**, který se dá stáhnout ze stránek <http://pypi.python.org/pypi/django-authority/0.4>.

6.1.5 Zpětná kompatibilita API

API (application programming interface) označuje v informatice rozhraní pro programování aplikací. Tento termín používá softwarové inženýrství. Jde o sbírku procedur, funkcí či tříd nějaké knihovny (ale třeba i jiného programu nebo jádra operačního systému), které může programátor využívat. API určuje, jakým způsobem se funkce knihovny volají ze zdrojového kódu programu. [9]

S každým novým vývojem Django se přidávají nové funkce a knihovny. Je tedy nemožné udržet zpětnou kompatibilitu API i přes snahu vývojářů. Verze 1.0 ale přichází s dopřednou kompatibilitou, takže budoucí verze jako 1.x budou s 1.0 na úrovni API kompatibilní s minimálními změnami.

6.1.6 Definice constraints v databázi nebo v modelu

Každý framework si vytvoří model databáze, který převádí reálnou databázi (MySQL, Postgresql) a její tabulky do tříd, se kterými pak daný framework pracuje a přistupuje k datům objektově (ORM). Má to mnohé výhody jako třeba možnost vytvoření tzv. **constraints** (omezení) přímo v modelu frameworku, protože některé databáze jako MySQL nepodporují definici složitějších **constraints** přímo v databázi. **Constraints** omezují vkládání údajů do databáze podle definovaných pravidel. Například kontrolu správnosti emailové adresy.

Pro použití **constraints** v systému Django je nutné stáhnout příslušné složky s názvem django-check-constraints z adresy <https://github.com/theju/django-check-constraints>. Pak už jen stačí námi definované **constraints** (ve zdrojovém kódu modře) vložit do našeho databázového modelu. V tabulce adresy je definovaná podmínka CZ pro stát. Jiná vložená hodnota způsobí chybu.

```
1 from django.db import models
2 from check_constraints import Check
3
4 class Adresy(models.Model):
5     __metaclass__ = CheckConstraintMetaClass
6
7     ulice = models.CharField(max_length=50)
```

```
8     mesto = models.CharField(max_length=50)
9     psc = models.CharField(max_length=50)
10    stat = models.CharField(max_length=50)
11    class Meta:
12        db_table = u'adresy'
13        verbose_name = 'Adresa'
14        verbose_name_plural = 'Adresy'
15        constraints =(
16            ("check_stat ",Check(product_stat = ("CZ"))),
17        )
18
19    def __unicode__(self):
20        vypis = self.mesto+', '+self.mesto+', '+self.psc+', '+self.stat
21        return vypis
```

6.1.7 Nezávislost fyzického a logického modelu (přejmenování tabulek)

Pro tvorbu určitých aplikací je důležité, aby byl model nezávislý na databázi z pohledu názvů tabulek. To znamená, že můžeme přepsat názvy tabulek v databázi a k nim přepsat příslušné názvy tabulek v modelech. Celý tento proces nám zaručí to, že nemusíme zasahovat do již napsaných zdrojových kódu a přepisovat jednotlivé dotazy.

Na příkladu změníme v databázi tabulku **adresy** na tabulku **address** a v modelu přepíšeme řádek číslo 12 z `db_table = u'adresy'` na `db_table = u'address'`. Z toho plyne, že třída **Adresy** bude obsluhovat stále tu stejnou tabulku jako doposud, jen s jiným názvem.

```
1 from django.db import models
2 from check_constraints import Check
3
4 class Adresy(models.Model):
5     __metaclass__ = CheckConstraintMetaClass
6
7     ulice = models.CharField(max_length=50)
8     mesto = models.CharField(max_length=50)
```

```
9     psc = models.CharField(max_length=50)
10    stat = models.CharField(max_length=50)
11    class Meta:
12        db_table = u'adresy'
13        verbose_name = 'Adresa'
14        verbose_name_plural = 'Adresy'
15        constraints =(
16            ("check_stat ",Check(product_stat = ("CZ"))),
17        )
18    def __unicode__(self):
19        vypis = self.mesto+', '+self.mesto+', '+self.psc+', '+self.stat
20        return vypis
```

6.1.8 Časové a paměťové testy

1) Časové výsledky pro výpis všech registrovaných uživatelů ve frameworku Django.
Využití paměti pro test: **343,5 kB**.

Vzorky	Průměr[ms]	Median[ms]	Min[ms]	Max[ms]	Propustnost [kB/s]
<i>První test</i>					
100	52	51	39	213	6,8
<i>Druhý test</i>					
100	48	47	37	83	6,8
<i>Třetí test</i>					
100	48	45	37	71	6,8
<i>Průměrné výsledky</i>					
100	49,33	47,66	-	-	6,8

Tab. 1 - časové výsledky Django

2) Časové výsledky pro zobrazení údajů zvoleného uživatele ve frameworku Django.
Využití paměti pro test: **107,9 kB**.

Vzorky	Průměr[ms]	Median[ms]	Min[ms]	Max[ms]	Propustnost [kB/s]
<i>První test</i>					
30	52	49	40	105	3,4
<i>Druhý test</i>					
30	51	48	41	80	3,4
<i>Třetí test</i>					
30	52	49	40	68	3,4
<i>Průměrné výsledky</i>					
30	51,66	48,66	-	-	3,4

Tab. 2 - časové výsledky Django

3) Časové výsledky pro editaci zvoleného uživatele ve frameworku Django. Využití paměti pro test: **539,4 kB**.

Vzorky	Průměr[ms]	Median[ms]	Min[ms]	Max[ms]	Propustnost [kB/s]
<i>První test</i>					
30	54	51	49	243	3,1
<i>Druhý test</i>					
30	57	56	44	198	3,1
<i>Třetí test</i>					
30	59	55	44	250	3,1
<i>Průměrné výsledky</i>					
30	56,66	54	-	-	3,1

Tab. 3 - časové výsledky Django

4) Časové výsledky pro vytvoření nového uživatele ve frameworku Django. Využití paměti pro test: **472,1 kB**.

Vzorky	Průměr[ms]	Median[ms]	Min[ms]	Max[ms]	Propustnost [kB/s]
<i>První test</i>					
30	47	46	55	206	3,2
<i>Druhý test</i>					
30	58	53	53	233	3,2
<i>Třetí test</i>					
30	54	51	40	197	3,2
<i>Průměrné výsledky</i>					
30	52,66	50	-	-	3,2

Tab. 4 - časové výsledky Django

6.2 Ruby on Rails

V dalším textu bude název Ruby on Rails pro jednoduchost zkracován na ROR.

6.2.1 GUI komponenty

Jako GUI komponenty jsou chápány především gridy a záložky. Pro systém ROR existuje spousta externích gridů a záložek. U aplikace „Správa uživatelů“ byl využit grid s názvem

dhtmlxGrid. Mezi další externí gridy, které lze integrovat do ROR, je grid s názvem **Flexgrid on Rails** nebo také **Adobe Flex Data Grid**. Bohužel i u velkých projektů jsem se setkal s nepřesnou nebo zastaralou dokumentací.



Menu

[Fyzické osoby](#)
[Právnícké osoby](#)

Správa uživatelů

dhtmlxGrid

ID	Jméno	Příjmení	IČ	DIČ	Pohlaví
1	Petr	Nový	78907789	CZ8790789	Muž
2	Jana	Malá	97897897	CZ8078709	Žena
3	Radka	Králiková	78978908	CZ5675675	Žena
4	David	Král	87879898	CZ6565767	Muž
5	Jan	Kadlčík	353637	CZ6383903	Muž
6	Tomáš	Holý	3637489	CZ0980998	Muž
7	Jan	Krauz	3554887	CZ76876989	Muž
8	Lucie	Pyšná	3876746	CZ7867876	Žena
9	Tomáš	Zamazal	38497873	CZ8798899	Muž
10	Michaela	Koválová	7837387	CZ8789709	Žena
11	Eva	Donutilová	354664	CZ7378378	Žena
12	Jan	Mantl	7897997	CZ098098	Muž
13	Tomáš	Dvořák	8798789	CZ897897	Muž

new user

Obr. 34 - grid pro Rails

6.2.2 Rychlost vývoje aplikace

Tvorba webové prezentace „Správa uživatelů“ v systému ROR nebyla tak rychlá jako u frameworku Django, nicméně byla rychlejší než u frameworku Zend. Práci ušetří především automatické vygenerování tzv. lešení webu, které vytvoří základní návrh webu s funkčními formuláři a s funkční databází. ROR má také jako Django velké množství kvalitních tutoriálů, např. zde <http://www.root.cz/clanky/ruby-on-rails-uvod/>. Velkou pomocí jsou částečně přeložené originální stránky projektu ROR do češtiny. Odkaz na originální a přeložené stránky najdete zde <http://rubyonrails.org/>, <http://rubyonrails.cz/>.

6.2.3 Bezpečnost - SQL injection, XSS útoky...

Princip **SQL útoku** je popsán v odstavci 6.1.3. Framework ROR je přirozeně chráněn proti **SQL útokům** vestavěným filtrem, a to tak, že nahrazuje nebezpečné speciální znaky ' , " vložené do formuláře za znaky, které nijak nenaruší vzniklý SQL dotaz. Ukázka chráněného dotazu na přihlášení v ROR vypadá následovně:

```
User.find(:first, :conditions=>[" jmeno=? AND heslo=? ", params[:jmeno],
params[:heslo]])
```

nebo takto v Rails 1.2 :

```
User.find(:first, :conditions => {:jmeno => params[:jmeno],
:heslo => params[:heslo]})
```

Za parametry **jmeno** a **heslo** se tedy vloží díky ROR neškodné hodnoty a nedojde tak k žádnému **SQL útoku**.

Chránit se proti útoku můžeme přímo v definici modelu dané tabulky v databázi. Na příkladu níže je vidět nezabezpečená (řádek č. 2) a zabezpečená (řádek č. 5) definice modelu.

```
1 class Uzivatel < ActiveRecord::Base
2   def self.authenticate_unsafely(user_name, password)
3     where("user_name='#{user_name}' AND password='#{password}'").first
4   end
5   def self.authenticate_safely(user_name, password)
6     where("user_name = ? AND password = ?", user_name, password).first
7   end
8 end
```

XSS útoky jsou rovněž popsány v odstavci 6.1.3. Jde nejčastěji o útok v kombinaci HTML s JavaScriptem. Je tedy důležité ošetřit všechny vstupy speciálními funkcemi, které ROR nabízí.

Příklad na neoprávněné zobrazení cookies podstrčené libovolné stránce např. v URL:

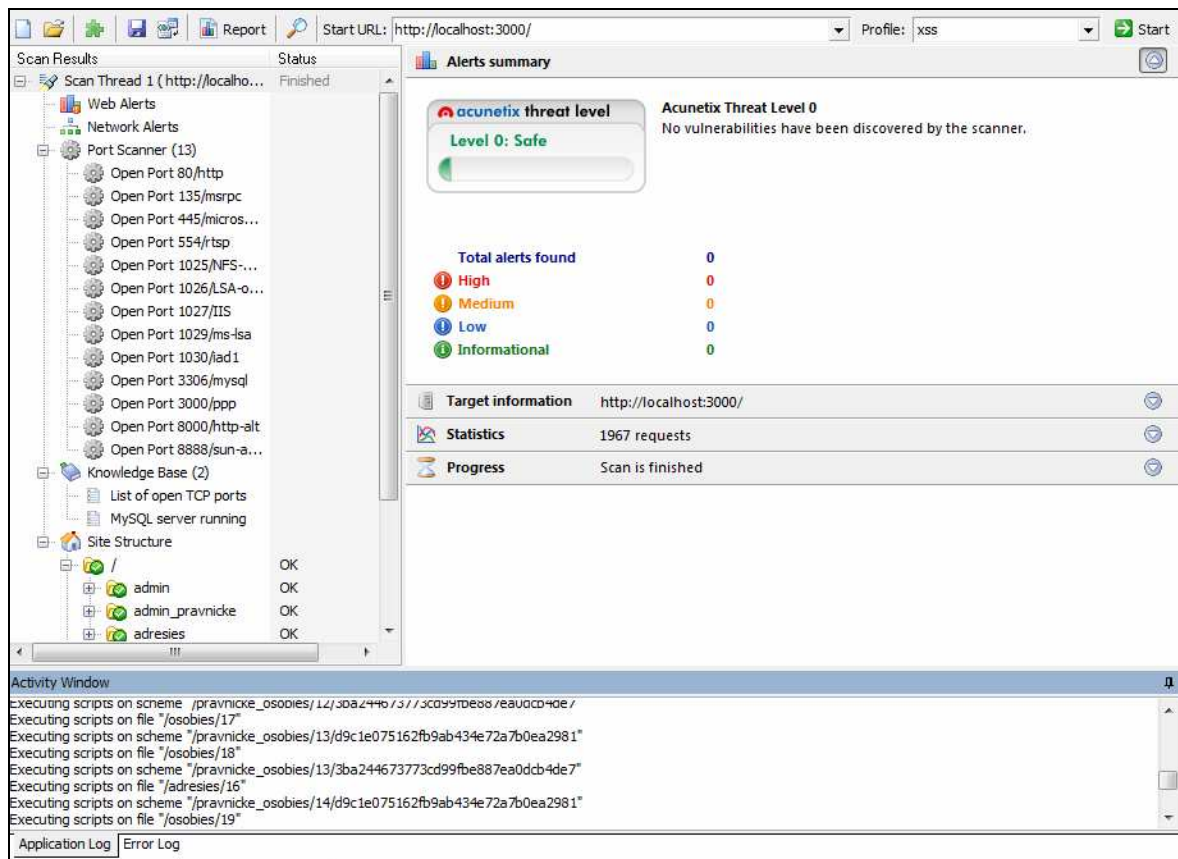
```
<script>document.write(document.cookie);</script>
```

a způsob, jak tomu na dané stránce zabránit:

```
strip_tags("<script>document.write(document.cookie);</script>")
```

Výstupem bude jen neškodný text `<script>document.write(document.cookie);</script>`, který se nijak nevykonává.

XSS útoky provedené na aplikaci napsanou v ROR byly otestovány pomocí programu **Acunetix Web Vulnerability Scanner**, který otestuje různé druhy útoků. Na obrázku níže je vidět výsledek testu s nulovými výsledky napadení.



Obr. 35 - testování SQL a XSS útoků na aplikaci napsanou v Ruby on Rails

Podrobnější popis **XSS útoků**, dalších útoků a funkcí můžete najít například zde:

<http://guides.rubyonrails.org/security.html#cross-site-scripting-xss>.

6.2.4 Víceúrovňová autorizace

ROR podobně jako Django využívá tzv. trojrozměrnou správu přístupových práv, která kombinuje role, zdroje a jejich práva. Obecné vysvětlení těchto pojmů je níže.

- Role - reprezentují jednotlivé registrované uživatele a skupiny. Role určuje, jaký typ uživatele je přihlášen. Může se jednat o Administrátora nebo o obyčejného registrovaného uživatele webu. K identifikaci se obvykle využívá uživatelské jméno nebo ID.
- Zdroje - představují prostředky, které jsou využívány rolemi. Zdroje mohou prezentovat například články, příspěvky, menu atd. Administrátor pak jednotlivým uživatelům (rolím) nastaví, do kterých částí má uživatel přístup povolen a do kterých ne.
- Práva - jsou přidělené operace, na které má uživatel (role) právo. Na běžném webu to mohou být práva typu vkládání příspěvků, editace příspěvků, mazání nebo třeba hlasování v anketách.

ACL (access control list) knihovny pro Ruby on Rails:

Pro framework ROR existují ACL pluginy, které se starají o přidělování rolí, zdrojů a práv. Mezi nejznámější plugin patří **ActiveACL**, který lze najít na oficiálních stránkách <http://rubyforge.org/projects/activeacl>. Dalšími většími pluginy pro správu autorizace a autentizace jsou **ACL System2** a **Goldberg**. Na zdrojovém kódu níže je ukázka definice tabulek `UserGroup` a `User`, přidělení a následné ověření práv pomocí pluginu **ActiveACL**.

```
1 class UserGroup < ActiveRecord::Base
2   acts_as_nested_set
3   acts_as_access_group
4   has_and_belongs_to_many :users
5 end
6 class User < ActiveRecord::Base
7   has_and_belongs_to_many :user_groups
8   acts_as_access_object :grouped_by => :user_groups
9   privilege_const_set('LOGIN')
10 end
11
12 registered_users = UserGroup.find_by_name('registered_users')
```

```
13 acl = ActiveAcl::Acl.create :section =>
14 ActiveAcl::AclSection.create(:description => 'users')
15 acl.allow = true # true is default
16 acl.privileges << User::LOGIN
17 acl.requester_groups << registered_users
18 acl.save
19 peter.has_privilege?(User::LOGIN) #=> true
20 jane.has_privilege?(User::LOGIN) #=> true
21 anonymous.has_privilege?(User::LOGIN) #=> false
```

6.2.5 Zpětná kompatibilita API

Co se týče **zpětné API kompatibility**, aktuální verze Ruby 1.9.x obsahuje vlastnosti nekompatibilní se staršími verzemi 1.8.x. Mezi nejdůležitější rozdílné způsoby práce patří např.:

- 1) Odlišná práce s argumenty bloku. Argument bloku již nemůže být globální proměnná ani členská proměnná objektu. Tedy žádné `|$global|` nebo `|object.member|`.
- 2) Řetězce již neobsahují `Enumerable`. Řetězce tedy již nepůjde iterovat metodou `each`. Ta ve stávající verzi iteruje přes jeho řádky. Podle zakladatele tedy není jasné, přes co by měla tato metoda iterovat, a proto místo ní přibudou metody `lines`, `chars`, `bytes`, které budou vracet příslušná pole, kterými již půjde bez problému iterovat pomocí `each`.
- 3) Řetězce vědí o svém kódování (výchozí hodnota UTF-8) a prvky řetězce vrací znaky. Kódování ovšem půjde předefinovat. Například při otevírání souboru půjde určit, v jakém kódování očekáváme obsah. Související změnou je, že přístup na jednotlivé znaky řetězce bude vracet daný znak (řetězec o délce jedna) a ne jeho ASCII kód, jak je tomu nyní. Například zápis `"ahoj"[1]` nevrátí 104, ale "h".

Dodržet tedy zpětnou kompatibilitu v průběhu času je velmi obtížné i pro takové velké projekty jako je Ruby on Rails. Přibývá tedy mnoho nových funkcí, které nejsou zpětně kompatibilní. Toto ovšem neplatí jen u softwarových projektů, ale i všude jinde.

6.2.6 Definice constraints v databázi nebo v modelu

Stejně jako u jiných frameworků je možnost u ROR definovat v modelech různé omezení pro vkládané hodnoty do databáze. Omezení se v modelech nezapisuje pomocí funkce **constraints**, ale pomocí funkce **validates**. Můžeme definovat mnoho podmínek pro splnění tvaru vložené hodnoty, jako třeba definice požadované minimální délky řetězce nebo definice tvaru emailu. Na příkladu je vidět požadovaný zápis pro kontrolu správnosti zadaného emailu.

```
1 class Kontakty < ActiveRecord::Base
2   belongs_to :osoba_ma_kontakts
3   belongs_to :pravnicka_osoba_ma_kontakts
4   validates_format_of :email, :with => /\A([\^@\s]+)((?:[-
5   a-z0-9]+\.\.)+[a-z]{2,})\Z/i, :on => :create
6 end
```

Existuje řada různých funkcí na validaci modelů. Jsou vypsány níže.

- `validates_acceptance_of`
- `validates_associated`
- `validates_confirmation_of`
- `validates_each`
- `validates_exclusion_of`
- `validates_format_of`
- `validates_inclusion_of`
- `validates_length_of`
- `validates_numericality_of`
- `validates_presence_of`
- `validates_size_of`
- `validates_uniqueness_of`

Podrobný popis těchto funkcí najdete zde:

<http://ar.rubyonrails.org/classes/ActiveRecord/Validations/ClassMethods.html#M000087>.

6.2.7 Nezávislost fyzického a logického modelu (přejmenování tabulek)

Pro tvorbu určitých aplikací je důležité, aby byl model nezávislý na databázi z pohledu názvů tabulek. To znamená, že můžeme přepsat názvy tabulek v databázi a k nim přepsat příslušné názvy tabulek v modelech. Celý tento proces nám zaručí to, že nemusíme zasahovat do již napsaných zdrojových kódů a přepisovat jednotlivé dotazy.

Na zdrojovém kódu níže vidíme model pro tabulku **osoby**. Tabulku **osoby** v databázi přejmenujeme na tabulku **persons** a dáme o tom vědět modelu pomocí funkce `set_table_name`.

```
1 class Osoby < ActiveRecord::Base
2   has_many :osoba_ma_adresus
3   has_many :osoba_ma_kontakts
4   has_many :pravnicka_osoba_ma_osobus
5   set_table_name "persons"
6 end
```

Pomocí řádku 5 jsme dané třídě řekli, že má pracovat s tabulkou **persons** a ne s tabulkou **osoby**, jež byla nastavena jako výchozí hodnota.

6.2.8 Časové a paměťové testy

1) Časové výsledky pro výpis všech registrovaných uživatelů ve frameworku Ruby on Rails. Využití paměti pro test: **962,7 kB**.

Vzorky	Průměr[ms]	Median[ms]	Min[ms]	Max[ms]	Propustnost [kB/s]
<i>První test</i>					
100	287	271	245	1116	19,0
<i>Druhý test</i>					
100	269	266	238	347	19,0
<i>Třetí test</i>					
100	274	272	244	342	19,0
<i>Průměrné výsledky</i>					
100	276,66	269,66	-	-	19,0

Tab. 5 - časové výsledky Ruby on Rails

2) Časové výsledky pro zobrazení údajů zvoleného uživatele ve frameworku Ruby on Rails. Využití paměti pro test: **650,9 kB**.

Vzorky	Průměr[ms]	Median[ms]	Min[ms]	Max[ms]	Propustnost [kB/s]
<i>První test</i>					
30	1166	1191	453	1969	14,8
<i>Druhý test</i>					
30	1128	1187	339	1853	14,8
<i>Třetí test</i>					
30	1076	1013	330	1778	15,0
<i>Průměrné výsledky</i>					
30	1123,33	1130,33	-	-	14,86

Tab. 6 - časové výsledky Ruby on Rails

3) Časové výsledky pro editaci zvoleného uživatele ve frameworku Ruby on Rails. Využití paměti pro test: **811,6 kB**.

Vzorky	Průměr[ms]	Median[ms]	Min[ms]	Max[ms]	Propustnost [kB/s]
<i>První test</i>					
30	449	438	419	1919	17,1
<i>Druhý test</i>					
30	498	490	465	1857	17,2
<i>Třetí test</i>					
30	480	476	532	1902	17,2
<i>Průměrné výsledky</i>					
30	475,66	468	-	-	17,16

Tab. 7 - časové výsledky Ruby on Rails

4) Časové výsledky pro vytvoření nového uživatele ve frameworku Ruby on Rails. Využití paměti pro test: **879,8 kB**.

Vzorky	Průměr[ms]	Median[ms]	Min[ms]	Max[ms]	Propustnost [kB/s]
<i>První test</i>					
30	411	410	432	1701	16,8
<i>Druhý test</i>					
30	418	418	330	1625	16,8
<i>Třetí test</i>					
30	435	428	534	1720	16,8
<i>Průměrné výsledky</i>					
30	421,33	418,66	-	-	16,8

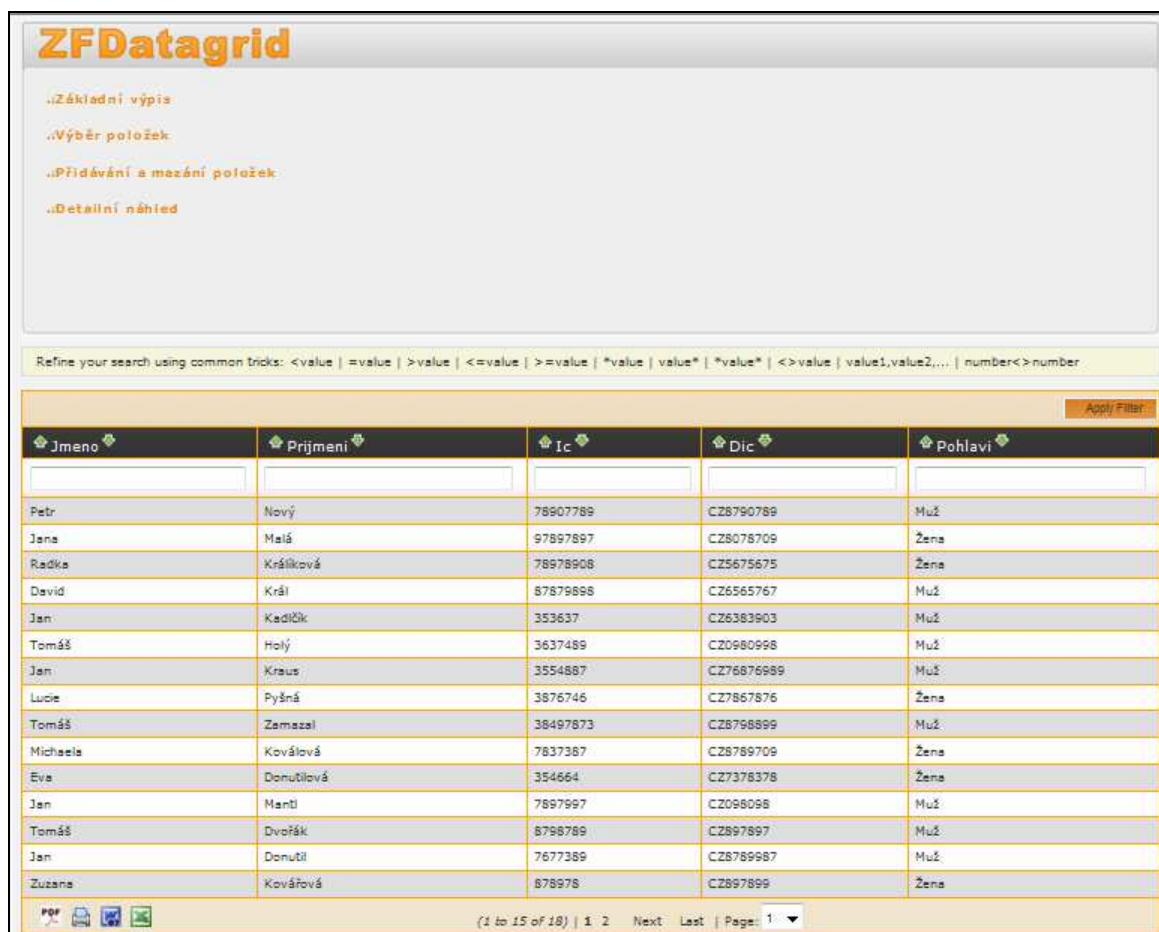
Tab. 8 - časové výsledky Ruby on Rails

6.3 Zend framework

Za PHP framework byl vybrán dobře známý framework Zend, za kterým stojí i vývojáři přímo z PHP.

6.3.1 GUI komponenty

Jako GUI komponenty jsou chápány především gridy a záložky. Pro systém Zend existuje spousta externích gridů a záložek. U aplikace „Správa uživatelů“ byl využit nejrozšířenější grid s názvem **ZFDataGrid**. Mezi další externí gridy, které lze integrovat, patří grid s názvem **Dojo grid**. Bohužel i u těchto velkých projektů jsem se setkal s nepřesnou nebo zastaralou dokumentací.



ZFDataGrid

..Základní výpis
..Výběr položek
..Přidávání a mazání položek
..Detailní náhled

Refine your search using common tricks: <value | =value | >value | <=value | >=value | *value | value* | *value* | <>value | value1,value2,... | number<>number

Jmeno	Prijmeni	Id	Dic	Pohlavi
Petr	Nový	78907789	CZ8790789	Muž
Jana	Malá	97897897	CZ8078709	Žena
Radka	Králiková	78978908	CZ5675675	Žena
David	Král	87879898	CZ6565767	Muž
Jan	Kadlčík	353637	CZ6383903	Muž
Tomáš	Holý	3637489	CZ0980998	Muž
Jan	Kraus	3554887	CZ76876989	Muž
Lucie	Pyšná	3876746	CZ7867876	Žena
Tomáš	Zemazal	38497873	CZ8798899	Muž
Michaela	Kovářová	7837387	CZ8789709	Žena
Eva	Donutilová	354664	CZ7378378	Žena
Jan	Mantl	7897997	CZ098098	Muž
Tomáš	Dvořák	8798789	CZ897897	Muž
Jan	Donutil	7677389	CZ8789987	Muž
Zuzana	Kovářová	878978	CZ897899	Žena

(1 to 15 of 18) | 1 2 Next Last | Page: 1

Obr. 36 - grid pro Zend

6.3.2 Rychlost vývoje aplikace

Z pohledu vývoje aplikace „Správa uživatelů“ v Zend frameworku trval vývoj nejdéle ze zadaných frameworků. Zend totiž neumožňuje vytvořit jedním příkazem jakoukoliv funkční prvotní aplikaci s formuláři a komunikací s databází jako třeba ROR. Neumožňuje ani aktivaci administračního rozhraní jako Django. Umožňuje jen vytvoření úvodního projektu s patřičným rozložením složek a souborů - což je čistý projekt. Jedná se tedy o „surové“ psaní kódu. Velkou výhodou je to, že framework stojí na objektovém programovacím jazyku PHP, který patří mezi nejrozšířenější programovací jazyky pro webové aplikace a lze najít spoustu návodů a tutoriálů.

6.3.3 Bezpečnost - SQL injection, XSS útoky...

V aplikaci „Správa uživatelů“ pro Zend byla použita databázová vrstva (ORM) Doctrine 1.2, která se stará o komunikaci mezi Zend frameworkem a databází. Vrstva Doctrine sama o sobě poskytuje **SQL injection** ochranu. Pro tuto aplikaci byla tato vrstva zvolena kvůli možnosti definovat libovolné omezení vložených záznamů v modelech tzv. constraints. Samotný Zend framework má také databázovou vrstvu a také poskytuje základní ochranu proti **SQL injection**. Dokáže tedy pracovat s databází i bez Doctrine.

Mějme dva různé zápisy pro vkládání hodnot do databáze pomocí Zendu. Zápis níže není zabezpečený proti **SQL injection**, protože vkládané proměnné jsou vkládané přímo do dotazu SQL.

```
$db->query("INSERT INTO kontakty (telefon, email) VALUES ('$telefon', '$email')");
```

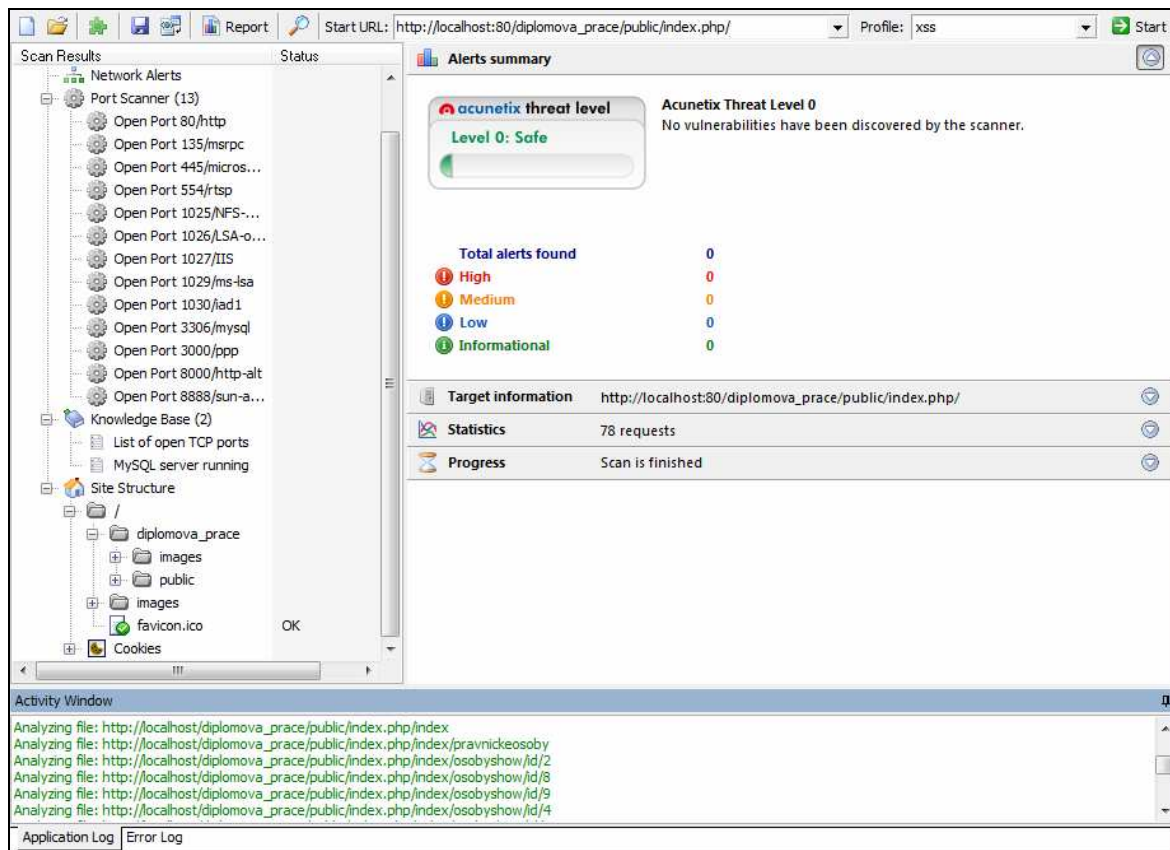
Zbylé dva dotazy jsou zabezpečené, protože se nevkładají řetězce přímo do dotazu, ale pomocí pole (první případ) nebo objektově (druhý případ), kde zasáhne databázová vrstva Zendu.

```
1 $db->query("INSERT INTO kontakty (telefon, email) VALUES (?,?)",  
    array($telefon, $email));
```

```
1 $data = array('telefon' => $telefon, 'email' => $email);
```

```
2 $db->insert('kontakty', $data);
```

Na aplikaci napsanou ve frameworku Zend byly také provedeny testovací **XSS útoky**. Testy byly opět provedeny v programu **Acunetix Web Vulnerability Scanner**. Na obrázku níže je vidět výsledek testu s nulovými výsledky napadení.



Obr. 37 - testování SQL a XSS útoků na aplikaci napsanou v Zendu

6.3.4 Víceúrovňová autorizace

Framework Zend využívá tzv. trojrozměrnou správu přístupových práv, která kombinuje role, zdroje a jejich práva. Obecné vysvětlení těchto pojmů je níže.

- **Role** - reprezentují jednotlivé registrované uživatele a skupiny. K jejich tvorbě Zend použijte třídy `Zend_Acl_Role` nebo `Zend_Acl_Role_Interface`. K identifikaci se obvykle využívá uživatelské jméno nebo ID. Role se uchovávají v registrech, což je vnitřní struktura `Zend_Acl` popisující vztahy mezi rolemi.
- **Zdroje** - představují prostředky, které jsou využívány rolemi. Zdroje mohou prezentovat například články, příspěvky, menu atd. Administrátor pak jednotlivým

uživatelům (rolím) nastaví, do kterých částí má uživatel přístup povolen a do kterých ne.

- Práva - jsou přidělené operace, na které má uživatel (role) právo. Na běžném webu to mohou být práva typu vkládání příspěvků, editace příspěvků, mazání nebo třeba hlasování v anketách.

ACL (access control list) knihovny pro Zend:

Zend jako ostatní frameworky nabízí knihovny pro autorizaci a autentizaci. `Zend_Acl` je knihovna, která poskytuje služby autorizace a `Zend_Auth` poskytující služby autentizace. `Zend_Auth` se implementuje pomocí metody `Zend_Auth::getInstance()`, díky které se nemusí držet v session objekt konkrétního uživatele, ale stačí si vyžádat pouze objekt `Zend_Auth`. Na zdrojovém kódu je vidět, jak se získají přihlašovací hodnoty (login, heslo) z databáze a porovnají se s hodnotami z přihlašovacího formuláře. Pokud je vše v pořádku, dojde k přesměrování a přihlášení požadovaného uživatele.

```
1 $adapter = new Zend_Auth_Adapter_DbTable(  
2     Zend_Db_Table_Abstract::getDefaultAdapter(),  
3     'uzivatele',  
4     'login',  
5     'heslo',  
6     'SHA1(?)'  
7 );  
8 $adapter->setIdentity($form->getValue('login'));  
9 $adapter->setCredential($form->getValue('heslo'));  
10  
11 $result = Zend_Auth::getInstance()->authenticate($adapter);  
12 if (!$result->isValid()) {  
13     echo "Špatné přihlašovací údaje!";  
14 } else {  
15     $this->_redirect('/');  
}
```

Zend_Acl má na starosti kontrolu uživatelských práv. To znamená, že kontroluje, jestli přihlášený uživatel (role) má dostatečná práva navštívit určité stránky. Administrátor má přístup do všech částí webu, kdežto registrovaný uživatel jen do přidělených částí webu.

Všechny role má na starosti rozhraní Zend_Acl_Role_Interface a samotné role se pak vytváří pomocí objektu Zend_Acl_Role. Tento objekt obsahuje pouze jedinou metodu, a to getRoleId. Vytváření zdrojů je podobné jako vytváření rolí. Je nutné implementovat Zend_Acl_Resource_Interface, která obsahuje metodu getResourceId. Role se ve výsledku registrují metodou addRole a zdroje metodou addResource. Využití knihovny Zend_Acl a definování rolí je vidět na zdrojovém kódu níže.

```
1 $acl = new Zend_Acl();
2 $acl ->addRole(new Zend_Acl_Role('host'))
3     ->addRole(new Zend_Acl_Role('clen'))
4     ->addRole(new Zend_Acl_Role('admin'));
5 $parent = array('host', 'clen', 'admin');
6 $acl->addRole(new Zend_Acl_Role('role'), $parent);
7 $acl->add(new Zend_Acl_Resource('zdroj'));
8 $acl->deny('host', 'zdroj');
9 $acl->allow('clen', 'zdroj');
10 echo $acl->isAllowed('role', 'zdroj')? 'allowed':'denied';
```

6.3.5 Zpětná komptabilita API

Framework Zend se snaží zachovat zpětnou komptabilitu, nicméně ve verzi 1.8.0 došlo k významné změně v práci s automatickým nahráváním tříd do projektu. V dřívějších verzích jsme byli zvyklí, že na začátku bootstrapu se zavolala metoda Zend_Loader::registerAutoload(), která dokázala nahrávat třídy z knihoven s přesně daným prefixem za nás. Tato možnost je i v nových verzích k dispozici, je však ale označena jako zavržená a ve verzi 2.0.0 bude definitivně odstraněna. Na zdrojovém kódu níže je popsána stará metoda pro nahrávání požadovaných tříd.

```
1 require_once 'Zend/Loader.php';
2 Zend_Loader::registerAutoload();
```

A zde je nová metoda pro nahrávání tříd

```
1 require_once 'Zend/Loader/Autoloader.php';
2 $loader = Zend_Loader_Autoloader::getInstance();
3 $loader->registerNamespace('App_');
```

Při vývoji aplikace „Správa uživatelů“ byla využita verze frameworku 1.11.3, ale v čase psaní teoretické a praktické části vyšla nová verze 1.11.5, která už např. podporuje české překlady hlášek ze Zend_Validate. Zatímco v aplikaci „Správa uživatelů“ a verzi frameworku 1.11.3 musely být chybové hlášky přeloženy samotným programátorem.

Popis jednotlivých verzí frameworku a jejich úpravy můžeme najít např. zde: <http://framework.zend.com/changelog/>.

6.3.6 Definice constraints v databázi nebo v modelu

Celá komunikace s databází v aplikaci „Správa uživatelů“ je realizována pomocí ORM vrstvy Doctrine, ve které jsou také definované patřičné **constraints**. Na zdrojovém kódu níže je ukázka definice **constraints** v modelu pro položku **email** a pro položku **stát**.

```
1 $this->hasColumn('email', 'string', 50, array(
2     'type' => 'string',
3     'length' => 50,
4     'fixed' => false,
5     'unsigned' => false,
6     'primary' => false,
7     'notnull' => true,
8     'autoincrement' => false,
9     'email' => true,
10    ));
1 $this->hasColumn('stat', 'string', 50, array(
2     'type' => 'string',
3     'length' => 50,
4     'fixed' => false,
5     'unsigned' => false,
6     'primary' => false,
```

```
7     'autoincrement' => false,  
8     'notnull' => true,  
9     'country' => true,  
10    ));
```

Tyto **constraints** si vynutí správný tvar vložené hodnoty do formuláře. To znamená, že email musíme zadat ve správném tvaru např. admin@zend.cz a stát musíme zadat v mezinárodním označení CZ. Nesprávnost zadaných údajů dá Doctrine najevo chybovou hláškou.

6.3.7 Nezávislost fyzického a logického modelu (přejmenování tabulek)

Pro tvorbu určitých aplikací je důležité, aby byl model nezávislý na databázi z pohledu názvů tabulek. To znamená, že můžeme přepsat názvy tabulek v databázi a k nim přepsat příslušné názvy tabulek v modelech. Celý tento proces nám zaručí to, že nemusíme zasahovat do již napsaných zdrojových kódů a přepisovat jednotlivé dotazy.

Na zdrojovém kódu níže vidíme model pro tabulku **osoby**. Tabulku **osoby** v databázi přejmenujeme na tabulku **persons** a dáme o tom vědět modelu pomocí funkce `setTableName()` na řádce číslo 5.

```
1 abstract class BaseOsoby extends Doctrine_Record  
2 {  
3     public function setTableDefinition()  
4     {  
5         $this->setTableName('persons');  
6         $this->hasColumn('id', 'integer', 4, array(  
7             'type' => 'integer',  
8             'length' => 4,  
9             'fixed' => false,  
10            'unsigned' => false,  
11            'primary' => true,  
12            'autoincrement' => true,  
13            ));
```

6.3.8 Časové a paměťové testy

1) Časové výsledky pro výpis všech registrovaných uživatelů ve frameworku Zend. Využití paměti pro test: **1244 kB**.

Vzorky	Průměr[ms]	Median[ms]	Min[ms]	Max[ms]	Propustnost [kB/s]
<i>První test</i>					
100	89	83	75	133	24,7
<i>Druhý test</i>					
100	91	92	76	158	24,7
<i>Třetí test</i>					
100	89	91	75	130	24,7
<i>Průměrné výsledky</i>					
100	89,66	88,66	-	-	24,7

Tab. 9 - časové výsledky Zend

2) Časové výsledky pro zobrazení údajů zvoleného uživatele ve frameworku Zend. Využití paměti pro test: **512,1 kB**.

Vzorky	Průměr[ms]	Median[ms]	Min[ms]	Max[ms]	Propustnost [kB/s]
<i>První test</i>					
30	113	98	82	496	16,1
<i>Druhý test</i>					
30	94	87	81	150	16,2
<i>Třetí test</i>					
30	98	97	81	147	16,2
<i>Průměrné výsledky</i>					
30	101,66	94	-	-	16,16

Tab. 10 - časové výsledky Zend

3) Časové výsledky pro editaci zvoleného uživatele ve frameworku Zend. Využití paměti pro test: **1139,9 kB**.

Vzorky	Průměr[ms]	Median[ms]	Min[ms]	Max[ms]	Propustnost [kB/s]
<i>První test</i>					
30	109	109	132	106	17,1
<i>Druhý test</i>					
30	99	98	109	150	17,1
<i>Třetí test</i>					
30	96	90	101	175	17,1
<i>Průměrné výsledky</i>					
30	101,33	99	-	-	17,1

Tab. 11 - časové výsledky Zend

4) Časové výsledky pro vytvoření nového uživatele ve frameworku Zend. Využití paměti pro test: **1019,8 kB**.

Vzorky	Průměr[ms]	Median[ms]	Min[ms]	Max[ms]	Propustnost [kB/s]
<i>První test</i>					
30	98	96	122	178	19,2
<i>Druhý test</i>					
30	99	98	137	181	19,2
<i>Třetí test</i>					
30	96	95	98	147	19,2
<i>Průměrné výsledky</i>					
30	97,66	96,33	-	-	19,2

Tab. 12 - časové výsledky Zend

7 POROVNÁNÍ VÝSLEDKŮ

7.1 Porovnání dostupnosti GUI

GUI komponenty, jako jsou gridy a záložky, jsou v dnešní době dostupné pro všechny testovací frameworky. Testované frameworky jsou vyvíjeny už po několik let, tudíž je dostupné i velké množství větších i menších projektů. Většinou také není problém externí gridy implementovat do našeho projektu. Nepříjemnou věcí snad mohou být zastaralé dokumentace, které značně ztěžují implementaci daných gridů do projektu.

7.2 Porovnání rychlosti vývoje aplikací

Aplikace „Správa uživatelů“ byla nejrychleji napsána ve frameworku Django, a to kvůli automaticky generovanému administračnímu rozhraní a velkému množství českých tutoriálů. Administrační systém je také nejpropracovanější ze všech aplikací.

Na druhou stranu nejpomaleji napsaná aplikace byla ve frameworku Zend, a to kvůli nemožnosti vygenerovat jakoukoliv funkční část aplikace, což umožňuje např. Ruby on Rails. Ruby on Rails umožňuje vygenerovat tzv. „lešení“ aplikace, což jsou funkční formuláře a jejich práce s databází.

7.3 Porovnání bezpečnosti

Bezpečnost u všech frameworků je na vysoké úrovni. Každý framework se po svém vypořádává s útoky jako jsou SQL injection nebo XSS (cross-site scripting). Pokud se ovšem pracuje přímo s funkcemi daného frameworku tak jak má, není umožněn žádný SQL útok. Útok je umožněn jen tehdy, pokud programátor píše určité dotazy „natvrdo“ pomocí SQL jazyka a obchází tak přímo napsané funkce daného frameworku. Dnes se s těmito útoky navíc vypořádávají servery samotné, které neumožňují vkládání jakéhokoliv zdrojového kódu do URL adresy nebo do políček formuláře.

Jednotlivé útoky byly u jednotlivých aplikací také otestovány testovacím programem **Acunetix Web Vulnerability Scanner 7**, který nabízí testování jak SQL injection tak XSS útoků. Další test byl proveden v programu **Netsparker 1.7.2.13**, který také nenašel žádné možnosti útoků, ale upozornil na fakt, že se v aplikaci nachází volně dostupné uživatelské

emaily. Emaily mohou být následně zaindexovány vyhledávacími roboty (Google, Seznam...) a využity pro doručování nevyžádané pošty.

7.4 Porovnání víceúrovňové autorizace

Každý framework má své metody a pluginy na řešení autorizace a autentizace. Pro aplikaci „Správa uživatelů“ v Zendu a v Ruby on Rails bylo nutné nastudovat dokumentaci pro správné používání dostupných funkcí autorizace. Tato akce odpadla u aplikace napsané v Djangu, protože vygenerované administrační rozhraní vytvoří přihlašovací formulář s potřebnými funkcemi za nás. Zde tedy na plné čáře z pohledu jednoduchosti vyhrál framework Django. Otázkou však zůstává to, pokud by uživatel chtěl použít autorizaci i jinde, než u administrace. To by už také vyžadovalo nastudování dokumentace.

7.5 Porovnání zpětné kompatibility API

Porovnání API kompatibility je obecně komplikovaná záležitost. Každý framework se snaží být zpětně kompatibilní, ovšem jak už bylo popsáno výše, nepodaří se udržet zpětnou kompatibilitu u frameworku Zend a u Ruby on Rails. Jediný framework, kterému se zatím daří být zpětně kompatibilní, je Django.

7.6 Porovnání definice constraints

U Ruby on Rails a Zend frameworku lze poměrně jednoduše definovat do databázových modelů constraints. Problém nastává u frameworku Django, kde je důležité nejdříve stáhnout příslušné složky s knihovnamy pro definici constraints.

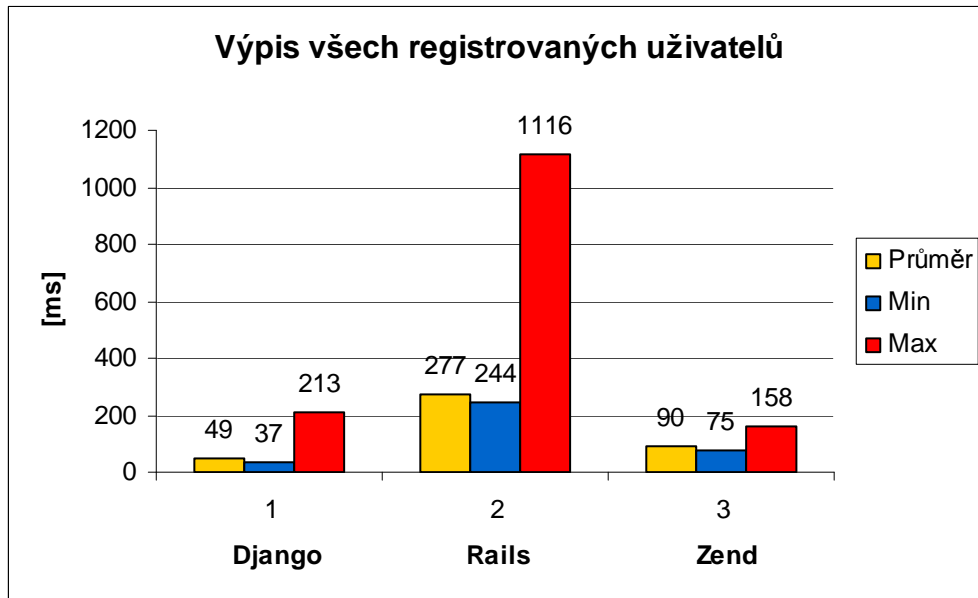
Nejvíce se mi líbila definice constraints ve frameworku Zend s ORM vrstvou Doctrine. Definice constraints v Doctrine je velice flexibilní a vše je také pěkně popsáno v dokumentaci Doctrine.

7.7 Porovnání nezávislosti fyzických a logických modelů

Všechny tři frameworky nabízejí možnost přejmenovat databázové tabulky v příslušných modelech. Přejmenování tabulek v modelu je u všech frameworků stejně snadné.

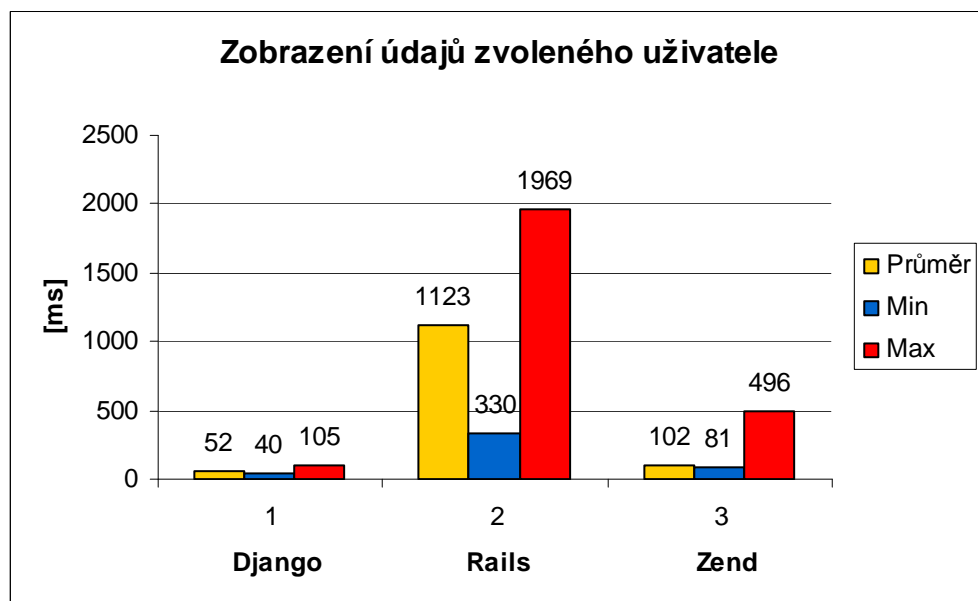
7.8 Porovnání časových a paměťových hodnot

Na grafech jsou prezentovány jednotlivé výsledky testu. Na grafu níže je výpis všech fyzických osob, které najdeme na obrázcích 7, 14 a 20.



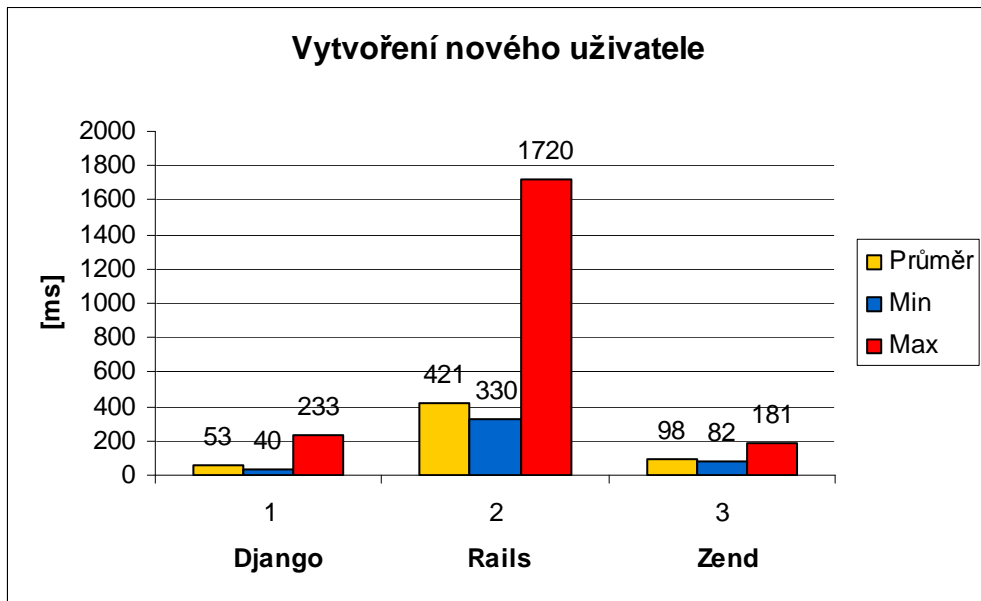
Obr. 38 - časové srovnání frameworků

Na grafu níže je výpis tří fyzických osob s různě vyplněnými údaji.



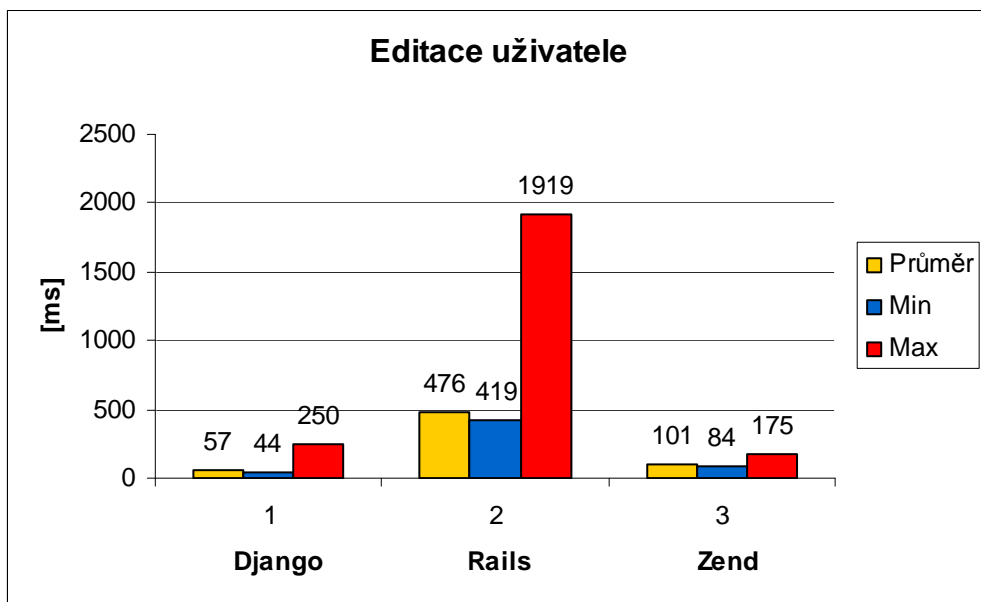
Obr. 39 - časové srovnání frameworků

Na grafu níže je vytvoření jedné fyzické osoby.



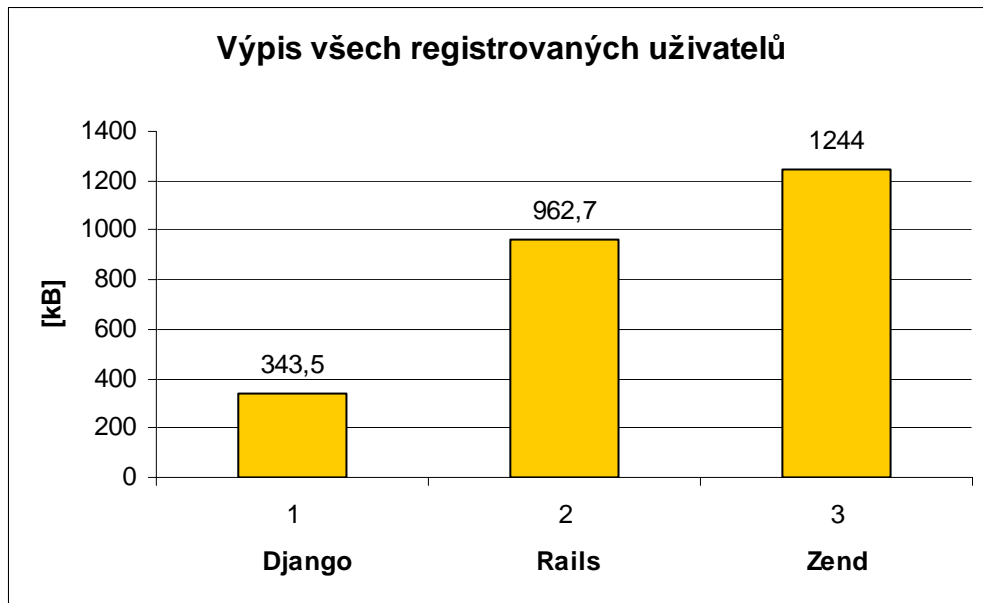
Obr. 40 - časové srovnání frameworků

Na grafu níže je editace jedné fyzické osoby.



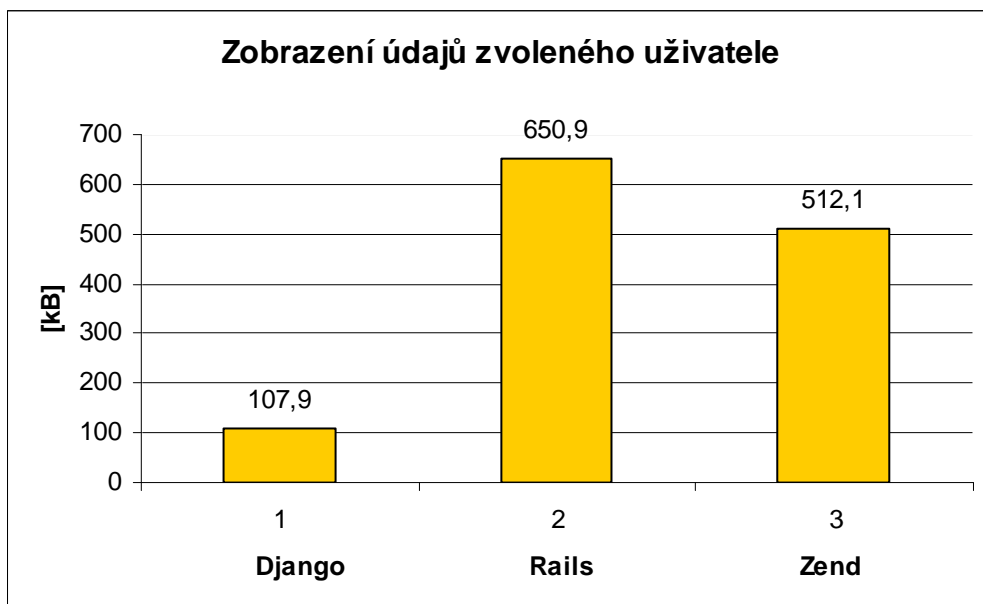
Obr. 41 - časové srovnání frameworků

Na grafu níže je výpis všech fyzických osob, které najdeme na obrázcích 7, 14 a 20.



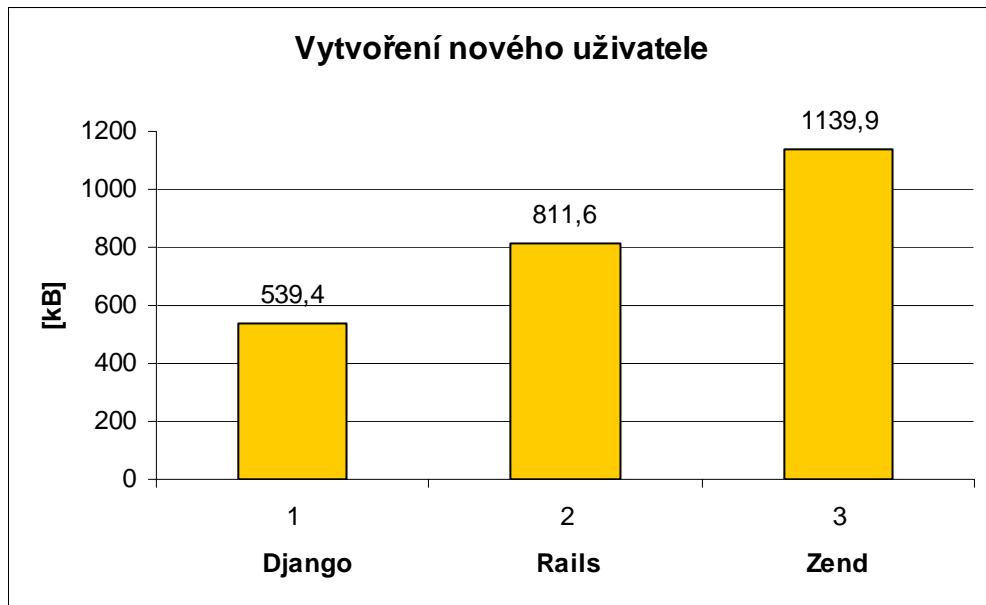
Obr. 42 - paměťové srovnání frameworků

Na grafu níže je výpis tří fyzických osob s různě vyplněnými údaji.



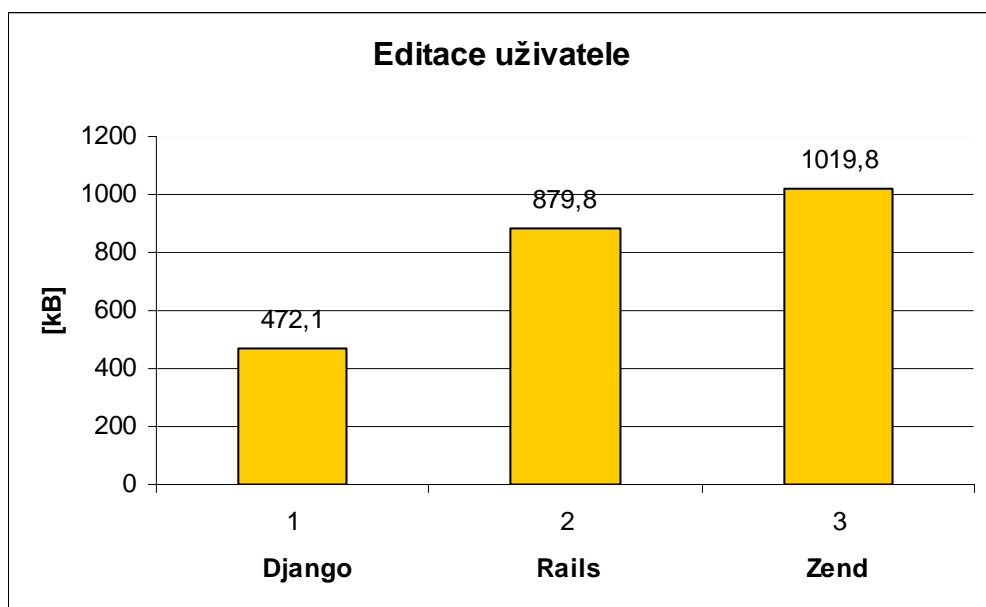
Obr. 43 - paměťové srovnání frameworků

Na grafu níže je vytvoření jedné fyzické osoby.



Obr. 44 - paměťové srovnání frameworků

Na grafu níže je editace jedné fyzické osoby.



Obr. 45 - paměťové srovnání frameworků

7.9 Doplnující informace

V tabulce níže jsou po zkušenostech popsány doplňující informace k daným frameworkům.

Frameworky	Dostupnost informací v českém jazyce	Hardwarová náročnost	Celková doba práce na aplikacích a testech
Django	<i>Velmi dobrá</i>	<i>Nízká</i>	<i>cca. 40 hodin</i>
Ruby on Rails	<i>Dobrá</i>	<i>Střední</i>	<i>cca. 80 hodin</i>
Zend	<i>Velmi dobrá</i>	<i>Střední</i>	<i>cca.120 hodin</i>

Tab. 13 - doplňující informace

ZÁVĚR

Tato diplomová práce by měla začínajícímu uživateli říci, který ze zadaných MVC frameworků si vede v jednotlivých bodech zadání lépe než ostatní. Diplomová práce je rozdělena na teoretickou část, kde se stručně dozvíme o testovaných frameworkcích a testovacích aplikacích, a praktickou část, která prezentuje, jak nainstalovat dané frameworky a jaké aplikace byly pro dané frameworky použity. Celá diplomová práce je také protkána důležitými internetovými odkazy, které ještě více rozšiřují jednotlivé body diplomové práce. Jednotlivé body zadání jsou podrobně probrány zvláště u každého frameworku, na konci se nalézají jen stručné srovnání bodů ze zadání.

Mě osobně zajímalo, jak si celkově povede framework Zend. Ten je totiž napsaný v PHP, kterému rozumím. Ovšem v testu neobstál nejlépe z hlediska jednoduchosti psaní kódu, ačkoliv na internetu je velké množství tutoriálů. Framework Ruby on Rails zase absolutně neobstál při časových a paměťových testech. To také potvrzuje fakt, že po spuštění virtuálního serveru trvá cca. 13 sekund, než dané stránky naběhnou. Absolutním vítězem ve většině bodů zadání (kromě definice constraints) se stal framework Django, se kterým nebyl při vývoji aplikace žádný problém a který měl i nejmenší časové a paměťové požadavky. Vývoj aplikace „Správa uživatelů“ v Djangu trval nejkratší dobu, a to především díky velkému množství tutoriálů, ať už v českém nebo anglickém jazyce. Jediné, co bych vytknul, je nutnost doinstalování databázového balíčku pro práci s databází MySQL a balíčku pro definici constraints.

Práce bude podle všeho využita k dalším účelům. Tudiž je možné, že se najdou lidé, kteří zadaným frameworkům rozumí na daleko vyšší úrovni a můžou zpochybnit některé výsledky. Jediným zpochybňujícím faktem snad může být jen to, že aplikace „Správa uživatelů“ vypadá v každém frameworku jinak, což se může projevit na časových a paměťových testech. Myslím si, že je ale prakticky nemožné napsat tři stejné aplikace v různých frameworkcích, a to už jen kvůli tomu, že každý framework s databází pracuje jinak a dělá v ní patřičné úpravy. Na obhajobu bych chtěl podotknout, že na nastudování všech frameworků jsem měl jen pár týdnů oproti těm, kteří se určitému frameworku věnují déle.

ZÁVĚR V ANGLIČTINĚ

The thesis should tell the novice who entered the MVC framework is doing at various points writing more than others. The thesis is divided into a theoretical part, where he briefly learn about the test frameworks and test applications, and a practical section that shows how to install the frameworks, and what applications were used for the frameworks. The whole thesis is also riddled with major Internet links harder to extend the points of the thesis. Individual entry points are discussed in detail separately for each framework, in the end found only a brief comparison of points of entry.

I personally wonder how the overall lead of Zend Framework. This is what is written in PHP, which I understand. But most failed the test in terms of simplicity of writing code, even though the Internet is a large number of tutorials. Ruby on Rails again completely failed the test of time and memory. It also confirms the fact that after the virtual server it takes approx. 13 seconds before the relay site. That is an absolute winner in most award points (except the definition of constraints) became the Django framework, with which the development application was no problem and that was the smallest time and memory requirements. Application Development "User Management" in Django took the shortest time, mainly due to the large number of tutorials, either in Czech or English. The only thing I rebuked, the need for installation of the remainder of database package for working with MySQL database package for the definition of constraints.

The work will probably be used for other purposes. Therefore, it is possible that there will be people who entered frameworkům mean at a much higher level and can affect some results. Perhaps only the fact that challenge may be just the application "User Management" looks different in every framework, which can result in time and memory tests. I think it is basically impossible to write three of the same application in different frameworks, if only because each framework with a database works differently, and it makes the appropriate adjustments. On defense, I would like to point out that the preparation of frameworks, I had a couple of weeks as opposed to those who give more specific framework.

SEZNAM POUŽITÉ LITERATURY

- [1] Django [online]. [cit. 2011-05-5].
Dostupný z www: < <http://cs.wikipedia.org/wiki/Django> >
- [2] Doctrine [online]. [cit. 2011-05-5].
Dostupný z www: < <http://zdrojak.root.cz/clanky/doctrine-2-uvod-do-systemu/> >
- [3] Ruby on Rails [online]. [cit. 2011-05-5].
Dostupný z www: < http://cs.wikipedia.org/wiki/Ruby_on_Rails >
- [4] Zend framework [online]. [cit. 2011-05-5].
Dostupný z www: < http://cs.wikipedia.org/wiki/Zend_Framework >
- [5] Zend studio [online]. [cit. 2011-05-5].
Dostupný z www: < <http://www.abclinuxu.cz/software/programovani/ide/zend-studio> >
- [6] Mysql [online]. [cit. 2011-05-5].
Dostupný z: < <http://www.artic-studio.net/slovnicek-pojmu/databaze-mysql/> >
- [7] JMeter [online]. [cit. 2011-05-5].
Dostupný z www: < <http://www.linuxzone.cz/index.phtml?ids=6&idc=772> >
- [8] SQL injection [online]. [cit. 2011-05-5].
Dostupný z www: < http://cs.wikipedia.org/wiki/SQL_injection >
- [9] API [online]. [cit. 2011-05-5].
Dostupný z www: < <http://cs.wikipedia.org/wiki/API> >
- [10] Böhmer, Marian. Zend Framework - Programujeme webové aplikace v PHP. [s.l.] Computer Press, a.s., 416 s. 978-80-251-2965-4.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

PHP	Hypertext Preprocessor - je skriptovací programovací jazyk určený především pro programování dynamických internetových stránek.
MVC	Model-view-controller - je softwarová architektura, která rozděluje datový model aplikace, uživatelské rozhraní a řídicí logiku do tří nezávislých komponent.
DRY	Don't Repeat Yourself - zkratka vyjadřující vlastnost na neopakovatelnost zdrojového kódu.
URL	Uniform Resource Locator - je řetězec znaků s definovanou strukturou, který slouží k přesné specifikaci umístění zdrojů informací na Internetu.
HTML	HyperText Markup Language - je jedním z jazyků pro vytváření stránek v systému World Wide Web, který umožňuje publikaci dokumentů na Internetu.
PDF	Portable Document Format - je souborový formát vyvinutý firmou Adobe pro ukládání dokumentů nezávisle na softwaru i hardwaru, na kterém byly pořízeny.
ORM	Objektově relační mapování - je programovací technika v softwarovém inženýrství, která zajišťuje automatickou konverzi dat mezi relační databází a objektově orientovaným programovacím jazykem.
MySQL	Je databázový systém, vytvořený švédskou firmou MySQL AB. Je k dispozici jak pod bezplatnou licenci GPL tak pod komerční placenou licenci.
PostgreSQL	Je objektově-relační databázový systém vydáván je pod licenci typu MIT a tudíž se jedná o free a Open Source software.
SQL	Structured Query Language - je standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích.
DQL	Doctrine Query Language - je dotazovací jazyk, který používá Doctrine pro práci s daty v relačních databázích.

DBAL	DataBase Abstraction Layer - abstrahuje zbytek aplikace od konkrétního typu databáze a zavádí tzv. notaci DQL (Doctrine Query Language).
IDE	Integrated Development Environment - zkratka pro vývojové rozhraní k daným frameworkům.
C++	Je objektově orientovaný programovací jazyk.
FTP	File Transfer Protocol - je v informatice protokol pro přenos souborů mezi počítači pomocí počítačové sítě.
XML	Extensible Markup Language - je obecný značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C.
CSS	Cascading Style Sheets - kaskádové styly, jazyk pro definici pravidel formátování internetových stránek.
phpMyAdmin	Je nástroj napsaný v jazyce PHP umožňující jednoduchou správu obsahu databáze MySQL prostřednictvím webového rozhraní.
GUI	Graphical User Interface - je uživatelské rozhraní, které umožňuje ovládat počítač pomocí interaktivních grafických ovládacích prvků.
EasyPHP	Je software pro běh jazyka PHP a databáze MySQL.
JMeter	Je software pro měření rychlosti webových aplikací.
XSS	Cross-site scripting - je metoda narušení WWW stránek využitím bezpečnostních chyb ve skriptech (především neošetřené vstupy).
CSRF	Cross-site Request Forgery - je jedna z metod útoku do internetových aplikací (typicky implementovaných skriptovacími jazyky nebo cgi).
API	Application Programming Interface - označuje v informatice rozhraní pro programování aplikací. Tento termín používá softwarové inženýrství.
CZ	Czech republic - vyjadřuje mezinárodní zkratku pro Českou republiku.
ROR	Ruby on Rails - zkratka frameworku Ruby on Rails

SEZNAM OBRÁZKŮ

<i>Obr. 1 - MVC architektura</i>	12
<i>Obr. 2 - databázové rozhraní phpMyAdmin</i>	18
<i>Obr. 3 - aplikace JMeter</i>	19
<i>Obr. 4 - databázové tabulky pro testovanou aplikaci „Správa uživatelů“</i>	23
<i>Obr. 5 - databázové tabulky pro administrační rozhraní v Django</i>	24
<i>Obr. 6 - úvodní stránka frameworku Django</i>	25
<i>Obr. 7 - úvodní stránka vytvořené aplikace „Správa uživatelů“ v Django</i>	26
<i>Obr. 8 - zobrazení vybraného uživatele v Django</i>	27
<i>Obr. 9 - zobrazení přihlašovacího formuláře v Django</i>	27
<i>Obr. 10 - zobrazení administračního rozhraní v Django</i>	28
<i>Obr. 11 - zobrazení administračního rozhraní v Django - výpis uživatelů</i>	29
<i>Obr. 12 - zobrazení administračního rozhraní v Django - vytvoření uživatele</i>	29
<i>Obr. 13 - úvodní stránka frameworku Rails</i>	31
<i>Obr. 14 - úvodní stránka vytvořené aplikace „Správa uživatelů“ v Rails</i>	32
<i>Obr. 15 - zobrazení vybraného uživatele v Rails</i>	33
<i>Obr. 16 - zobrazení přihlašovacího formuláře v Rails</i>	33
<i>Obr. 17 - zobrazení administračního rozhraní v Rails</i>	34
<i>Obr. 18 - zobrazení administračního rozhraní v Rails - vytvoření uživatele</i>	35
<i>Obr. 19 - úvodní stránka frameworku Zend</i>	36
<i>Obr. 20 - úvodní stránka vytvořené aplikace „Správa uživatelů“ v Zendu</i>	37
<i>Obr. 21 - zobrazení vybraného uživatele v Zendu</i>	37
<i>Obr. 22 - zobrazení přihlašovacího formuláře v Zendu</i>	38
<i>Obr. 23 - zobrazení administračního rozhraní v Zendu</i>	39
<i>Obr. 24 - zobrazení administračního rozhraní v Zendu - editace uživatele</i>	40
<i>Obr. 25 - nastavení aplikace JMeter</i>	41
<i>Obr. 26 - nastavení aplikace JMeter</i>	42
<i>Obr. 27 - zobrazení odesílaných dat aplikací JMeter</i>	42
<i>Obr. 28 - zobrazení výsledků v aplikaci JMeter</i>	43
<i>Obr. 29 - grid pro Django</i>	45
<i>Obr. 30 - testování SQL a XSS útoků na aplikaci napsanou v Django</i>	47
<i>Obr. 31 - nastavení přístupu do administrace</i>	49

<i>Obr. 32 - nastavení údajů registrovaného uživatele</i>	50
<i>Obr. 33 - nastavení uživatelského oprávnění v administračním rozhraní</i>	50
<i>Obr. 34 - grid pro Rails</i>	56
<i>Obr. 35 - testování SQL a XSS útoků na aplikaci napsanou v Ruby on Rails</i>	58
<i>Obr. 36 - grid pro Zend</i>	65
<i>Obr. 37 - testování SQL a XSS útoků na aplikaci napsanou v Zendu</i>	67
<i>Obr. 38 - časové srovnání frameworků</i>	76
<i>Obr. 39 - časové srovnání frameworků</i>	76
<i>Obr. 40 - časové srovnání frameworků</i>	77
<i>Obr. 41 - časové srovnání frameworků</i>	77
<i>Obr. 42 - paměťové srovnání frameworků</i>	78
<i>Obr. 43 - paměťové srovnání frameworků</i>	78
<i>Obr. 44 - paměťové srovnání frameworků</i>	79
<i>Obr. 45 - paměťové srovnání frameworků</i>	79

SEZNAM TABULEK

<i>Tab. 1 - časové výsledky Django</i>	54
<i>Tab. 2 - časové výsledky Django</i>	54
<i>Tab. 3 - časové výsledky Django</i>	55
<i>Tab. 4 - časové výsledky Django</i>	55
<i>Tab. 5 - časové výsledky Ruby on Rails</i>	63
<i>Tab. 6 - časové výsledky Ruby on Rails</i>	63
<i>Tab. 7 - časové výsledky Ruby on Rails</i>	64
<i>Tab. 8 - časové výsledky Ruby on Rails</i>	64
<i>Tab. 9 - časové výsledky Zend</i>	72
<i>Tab. 10 - časové výsledky Zend</i>	72
<i>Tab. 11 - časové výsledky Zend</i>	73
<i>Tab. 12 - časové výsledky Zend</i>	73
<i>Tab. 13 - doplňující informace</i>	80

SEZNAM PŘÍLOH

- PI** Příloženo dokumentační CD obsahující elektronickou verzi diplomové práce a všechny dokumenty včetně zdrojových kódů.

PŘÍLOHA P I: NÁZEV PŘÍLOHY

CD – Diplomová práce