Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

Katedra informačních technologií

Porovnání Nette a Zend frameworku

BAKALÁŘSKÁ PRÁCE

Student : Petr Olišar

Vedoucí : Ing. Jan Mittner

Oponent : Ing. Lukáš Burkoň

2012

Poděkování

Chtěl bych vyjádřit poděkování Ing. Janu Mittnerovi za cenné připomínky, odborné rady a podněty k zamyšlení během vypracování bakalářské práce.

Abstrakt

Tato bakalářská práce je zaměřena na představení a porovnání Zend a Nette frameworku v jejich nejnovějších verzích (29. listopadu). Toto představení a porovnání je dosaženo dvěma cestami. Jednak čerpáním z veřejných zdrojů a jednak vypracováním dvou totožných aplikací za použití Nette a Zend frameworku. Na základě těchto dvou aplikací došlo k měření, jehož výsledky jsou hlavním výstupem aplikace. Cílem práce je tedy porovnat schopnosti, které dané frameworky mají, a jejich uživatelskou pohodlnost při používání. Práce je strukturována do 3 celků. První část seznamuje s metodikou porovnání, metrikami a bodovým ohodnocením jednotlivých kritérií. V druhé části je uvedeno, co to je Framework, a je představen návrhový vzor MVC, na kterém oba frameworky stojí. Třetí část je vlastní porovnání obou frameworků na základě metriky uvedené v předchozí části.

Klíčová slova

Framework, Nette, Zend, návrhový vzor, PHP, MVC, model, pohled, kontrolor, Cross Site Scripting, SQL Injection, Cross Site Request Forgery.

Abstract

This thesis is focused on presentation and comparison of Zend and Nette frameworks in their latest versions, which were available on 29 November. This is achieved by 2 ways: using the public sources and writing out 2 almost the same web aplications using Nette and Zend frameworks. The main output of the thesis is comparison of qualities of the two frameworks, which is based on undertaken measurements. The work has 3 sections. The first section introduces us with a method, which was used to compare frameworks, then with metrics and scoring of each criterion. The introduction of Zend and Nette framework and MVC pattern can be found in second section. The last section is crucial, it finally copares Zend and Nette framework.

Keywords

Framework, Nette, Zend, design patterns, PHP, MVC, model, view, controller, Cross Site Scripting, SQL Injection, Cross Site Request Forgery.

1. Úvod

Neustálé zlepšování programovacích jazyků a programátorských postupů vede ke psaní složitějších a komplexnějších aplikací. Pro ulehčení práce vznikly frameworky. Frameworků je velká řada téměř ve všech programovacích jazycích. Jako se porovnávají jazyky navzájem, tak se porovnávají i frameworky mezi sebou. Cílem takových srovnání je zjistit, který z nich je nejlepší, nejrychlejší, nejbezpečnější, atd.

Cílem této práce je objasnit, co to framework je, seznámit s návrhovým vzorem MVC, na kterém srovnávané frameworky stojí, a porovnat Nette se Zendem na základě níže uvedené metodiky. Jedná se u nás pravděpodobně o nejrozšířenější frameworky. Výsledky této práce by mohly sloužit jako jeden z možných zdrojů informací pro programátory, kteří se rozhodují, který framework se naučit. Jde o porovnání dvou frameworků od začátku, tzn. od stažení až téměř po nasazení v praxi. Celá práce je podložena aplikací, která je téměř totožně naprogramována jak v Nette, tak v Zend frameworku. Je velmi pravděpodobné, že pokud by tuto práci vypracovali ti, kteří dané frameworky ovládají velmi dobře, že by se výsledky více či méně lišili. Tato práce je napsána „ze zelené louky“ mých znalostí obou frameworků, proto se do výsledku obou aplikací i práce samotné promítla kvalita dokumentace.

Aplikace, která je součástí práce, je zaměřena na nejběžnější práci spojenou s vývojem webové aplikace. Zejména se jedná o práci s databází, formulářovými prvky a přihlašováním uživatelů.

Hlavním zdrojem informací pro tuto práci byla naprogramovaná aplikace, dokumentace obou frameworků a předchozí a jiná podobná srovnání.

1.1. Rešerše

Prací, které by se týkaly porovnání frameworků, je poměrně hodně. Tyto práce jsou většinou dostupné pouze na internetu, protože se jedná o tak rychle se měnící prostředí, že už v době vydání knížky nebo jiné tištěné publikace by byla daná publikace zastaralá. Já jsem se rozhodl porovnat dva PHP (rekurzivní zkratka - PHP: Hypertext Preprocessor) frameworky Nette a Zend. Budu proto uvažovat pouze práce, ve kterých se porovnávají tyto dva frameworky. Dále budu uvažovat veškeré jejich verze, protože aktuální srovnání Nette 2 a Zend 2 není žádné.

- Webová stránka PHP frameworks (PHPFrameworks.com, 2012) je zaměřena na základní porovnání vlastností jednotlivých frameworků. Sice zde není zastoupen Nette framework, nicméně jsem některá kritéria použil v bodovém porovnání níže v práci.
- Best web frameworks (Bestframeworks.com, 2011): Tato stránka porovnává podobně jako předchozí bodově frameworky; po rozkliknutí obsahuje stručnou charakteristiku frameworku. V tomto srovnání je zastoupen už i Nette, nicméně nejedná se o srovnání nejaktuálnějších verzí.
- Velký test PHP frameworků (Daněk, 2008): Toto je asi nejpodobnější porovnání. Porovnává ranou verzi Nette (verze 0.9) se Zend frameworkem 1.5.1. Srovnává je na základě rychlosti a popisuje i přívětivost dokumentace. Jedná se o srovnání přes 4 roky staré a za tu dobu se oba frameworky značně vyvinuly, takže lze toto srovnání považovat již za irelevantní.
- Souboj frameworků (Knesl, 2012): je zaměřen na rychlost vývoje pěti různých frameworků (Zend není součástí), nicméně se zmiňuje i o čistotě kódu a tak nějak zevrubně porovnává dané frameworky. Kromě porovnání rychlosti vývoje aplikace je tato aplikace převážně o subjektivním pocitu autora. Že se nejedná o kvalitní porovnání, píše například i jeden z účastníků samotného souboje frameworků Jakub Vrána (2012).

Porovnání je tedy na internetu několik. Vzhledem k velmi rychlému vývoji jsou však většinou zastaralá nebo porovnávají tyto frameworky jen velmi zevrubně.

2. Metodika porovnávání frameworků

Metodiku porovnávání by šlo rozdělit na dvě oblasti. Na část objektivní (např.: měření rychlosti aplikace) a část subjektivní (např.: dokumentace, práce s frameworkem). Jednotlivé části srovnání mají různou váhu, která je vyjádřena v procentech, a na závěr je na základě bodového ohodnocení a subjektivních zkušeností uveden verdikt. Za každou část tedy bude možné získat tolik bodů, kolik je její váha v procentech, jelikož jeden bod se rovná jednomu procentnímu bodu.

V práci je snaha vybrat a zaměřit se na nejdůležitější aspekty, které přidávají přidanou hodnotu programátorovi a na jejich použitelnost. Vždy první odstavec u jednotlivých metodik

porovnání objasňuje, proč jej považuji za důležitý. Na závěr každé metody je uvedeno, na jakém základě dochází k rozdělení vah důležitosti mezi jednotlivými oblastmi, a jsou rozděleny jednotlivé body v rámci jedné části.

Jednotlivé váhy jsou zvoleny tak, aby byli vůči sobě relativní. Rychlost a bezpečnost jsou nejdůležitějšími kritérii úspěšného frameworku. Kvůli jejich důležitosti mají společně váhu ¾ všech bodů. Dokumentace má váhu 1/5 bodů. Vzhledem k vážnosti rychlosti a bezpečnosti je dokumentace celkem nedůležitá. Ovšem to je právě to, na základě čeho se programátoři rozhodují, který framework se naučí a jestli u něj zůstanou. Pokud se nemají z čeho učit, jak framework používat, pravděpodobně se nebude jednat o framework, který by si vybrali. Zbývajících 5% je určeno pro mírnou korekci na základě zkušeností z používání. Ne vše je možné exaktně změřit. Někdy člověk ani není schopný říct, co mu vlastně vadí – prostě mu „něco na tom nesedí“. Takovou více či méně obhájenou výpovědí je právě tato poslední kapitola.

Kritérií by se mohlo vymyslet i více, například rychlost vývoje aplikace za použití frameworku nebo testovatelnost aplikace. Kvůli rozsahu práce jsem se rozhodl vybrat 3 kritéria, která podle mě nejvíce ovlivňují výběr frameworku, plus praktické poznatky z používání.

2.1. Bezpečnost

Jedná se o vůbec nejdůležitější test. Internetové aplikace často vyžadují po uživatelích, aby jim poskytly různé citlivé údaje. Typicky se jedná minimálně o přihlašovací jméno a heslo. Často internetové aplikace vyžadují i další informace jako je e-mailová adresa, bydliště, telefonní číslo a další. Jde tedy o velice citlivá data a každý provozovatel webových služeb musí zajistit jistou formu bezpečí pro tato uživatelská data.

Existuje několik útoků, pomocí kterých se může útočník zmocnit některých nebo dokonce všech uživatelských dat, manipulovat s databází či jen podstrčit obrázek na cizí web. Právě na toto se zaměřuje první část porovnání. Práce si neklade za cíl porovnat bezpečnost vůči všem útokům, ale je zaměřena jen na 3 nejčastější útoky: SQL (Structured Query Language) injection, Cross Site Scripting (XSS), Cross Site Request Forgery (CSRF). Výsledkem tohoto porovnání je, jak moc jsou frameworky uživatelsky přívětivé vůči programátorům. Ideální Framework je takový, který se o bezpečnostní stránku aplikace postará sám, nebo alespoň upozorní programátora na případný, bezpečnostní problém. Naopak nejhorší Framework je

ten, který programátora nevaruje, že napsal nebezpečný skript, který nezabezpečuje stránky proti případnému útočníkovi. Útoky byly provedeny vůči aplikaci v produkčním režimu. Váha všech útoků je stejná. Kvůli počtu bodů, který není dělitelný 3, mají útoky XSS a CSRF o půl bodu vyšší váhu. Je to z toho důvodu, že dokáží přímo manipulovat s daty databáze.

2.1.1. Cross site scripting

Útok probíhal tak, že na aplikace, které byly naprogramovány bez jakéhokoli důrazu na bezpečnost, se provedl útok na **homepage/about/date=$variable**. Útok spočíval ve vložení obrázku (**http://www.romea.cz/images/servis/anonymous-cz.jpg**) do stránky. Poté se útok opakoval s vypnutím/zapnutím escapování. Proti JavaScriptovému útoku oba frameworky obstály, to znamená, že se mi nepodařilo vložit a spustit `<script>alert('útok')</script>`. Výsledky jsou níže v kapitole 6.1.

2.1.2. SQL Injection

Tento útok probíhal tak, že se pokoušel vložit SQL kód `'; DELETE FROM customers WHERE 38'` jednak do editačního pole ve formuláři a jednak předat přímo v aplikaci tabulce.

2.1.3. Cross Site Request Forgery

Tento test byl proveden proti formulářům i proti proměnným předávaným ve formě GET. Test proti formulářům byl proveden tak, že byl vložen přesný HTML kód pro formulář editace produktu, který byl vygenerován Nette respektive Zend frameworkem do útočné stránky a poté odeslán. Test proti proměnné předávané jako parametr formou GET byl proveden následovně: byl vytvořen obrázek, do jehož těla byl vložen odkaz na stránku pro smazání produktu.

Jelikož se jedná o nejdůležitější část měření, její váha je **43%**.

2.2. Rychlost aplikace

Rychlost aplikace je dalším velmi důležitým kritériem při výběru frameworku. Je prokázáno, že čím déle se webová stránka načítá, tím spíše ji lidé opustí bez toho, aby si byť jen prohlédli její obsah. Podle různých studií vychází i různě dlouhá doba, po kterou jsou lidé ochotní počkat na vykreslení. Důležité je, že i malá změna rychlosti v psychice lidí hraje velkou roli. Mohlo by se říci, že v podstatě záleží i na desítkách milisekund. (Website Optimization, 2008)

Porovnání bylo provedeno za pomoci nástroje pro vývojáře, který je zabudován v prohlížeči Google Chrome. Jeho záložka „network“ je schopna zobrazit jednotlivé přenášené soubory, počet požadavků na server, celkovou velikost přenesených dat a hlavně dobu, za kterou je stránka načtena. Kromě kompletního načtení zobrazuje i za jak dlouho je načten a zpracován v prohlížeči HTML soubor bez obrázků, css (Cascading Style Sheets) a JavaScriptu. To je čas, který je níže porovnáván a hodnocen. Odpovídá to rychlosti zpracování PHP na serveru, vygenerování, odeslání a zpracování HTML v prohlížeči. Aplikace jsou nahrané na Localhostu (vlastní počítač), kvůli tomu, aby výkyvy v rychlosti internetu nezkreslili měření. Konfigurace počítače:

- Procesor: Intel Core i5-2500 (3,30 GHz
- Paměť: 4 GB
- HDD: SSD 128 GB

Prostředí:

- Apache: 2.4.2
- PHP: 5.4.3
- SQL: 5.5.24

Srovnání tohoto typu (rychlosti frameworku) je dostupných celá řada. Nejpodobnější tomuto kritériu je starší Velký test PHP frameworků, které provedl Petr Daněk (2008) na serveru root.cz. Tento test porovnává rychlost a dokumentaci jednotlivých frameworků. Jedná se o velký test, který obsahuje 10 PHP frameworků a čisté PHP. Srovnání rychlosti je hlavním výstupem toho testu a je velmi podrobné. Naopak aplikace je velmi jednoduchá a dokumentace jen shrnuta v jednom odstavci.

Systém porovnání v této práci je dost podobný. Každý test rychlosti byl proveden třicetkrát a celé testování se opakovalo třikrát (tedy 3x30 měření). Průměrná hodnota všech měření je brána jako výsledek. Měření probíhalo vždy stejně. Měřilo se zároveň, znovunačtením stránky (reload) vždy jedna aplikace, druhá aplikace a poté se zapsaly výsledky. Pořadí se střídalo následovně: 1. Měření: Nette, Zend; 2. Měření: Zend, Nette; 3. Měření: ½ Zend, ½ Nette. Níže v příloze lze nalézt jednotlivá měření.

1) První test spočívá v zobrazení úvodní stránky (homepage). Na této stránce je zobrazen statický obsah s vygenerovaným menu dvou úrovní. Jedná se tedy o test přístupu k databázi s příkazem SELECT WHERE. Rychlost tohoto bude promítnuta do pozadí celého testování, protože dynamicky generované menu je na všech stránkách aplikace.
2) Dalším testem je test rychlosti SELECT přes všechny položky, které je realizováno na stránce uživatelů v administraci. Tato stránka vypíše seznam všech uživatelů aplikace o 20-ti členech. Nette zde mělo trochu ztížené podmínky, protože byl vložen formulář pro editaci přímo do stránky, zatímco v Zend aplikaci byl jen odkaz na editaci.
3) Předposledním testem je test na příkaz EDIT v editaci produktů.

Jedná se o druhý nejdůležitější test, který má bodové ohodnocení **32%.**

2.3. Dokumentace

Jedná se o subjektivní část testu, ve které bylo porovnáno, jak lehce se daný framework z dokumentace učí a jak je dokumentace přehledná a dostačující.

Dokumentace je velmi důležitá pro programátora, který začíná s vybraným frameworkem. Avšak programátor, který perfektně zná framework, se kterým pracuje, ji už využívá minimálně a jsou pro něj důležitější kritéria, jako je právě bezpečnost a rychlost.

2.3.1. První aplikace

V této kapitole je popsáno, co by měl takový tutoriál (Nette - První aplikace, Zend – Getting started) obsahovat a jak tento obsah naplňují oba frameworky. Tato část je oceněna 10 body, přičemž každá položka, která je v dokumentaci obsažena získá jeden bod. Jedinou výjimkou je bezproblémovost, kde se hodnotí, jestli programování podle tutoriálu vedlo bez problému k naprogramování ukázkové aplikace (aplikace není součástí práce). Tato položka je ohodnocena 2 body.

2.3.2. Příručka

Kapitola Příručka se věnuje už samotné dokumentaci. Tabulka ohodnocení obsahuje pět kritérií, která byla zvažována a každé kritérium tedy mělo stejnou váhu, dva body. Dva body se mohou slovně reprezentovat jako „ano“, jeden bod jako „částečně“ a nula bodů jako „ne“.

Váha je rozdělena přesně na půl mezi tutoriál a příručku. Je to z toho důvodu, že pokud je špatně napsaný tutoriál, tak to může odradit (a z pravidla odradí) nové uživatele frameworku. Pokud je špatně napsaná dokumentace, framework se stává těžce použitelným a programátora to může vést k poohlédnutí se po jiném. Pro získání a udržení programátora je důležité, aby byli tutoriál i příručka napsány kvalitně oba při přibližně stejné důležitosti.

Dokumentace je tedy nejdůležitější v začátku, ale její význam postupně klesá. Z toho důvodu a z důvodu nemožnosti exaktně porovnat dokumentace Nette a Zend frameworku (závisí tedy na subjektivním pohledu) je váha tohoto kritéria **20%.**

2.4. Programování za pomoci frameworku

Poslední bod testování je spíše souhrn poznatků z učení se jednotlivých frameworků. Tato část je velmi subjektivní a shrnuje jen některé nejvýraznější rozdíly v práci s frameworky Zend a Nette. Nechá se říci, že se jedná o jakýsi dojem z frameworku a jako takový se může (a pravděpodobně se bude) lišit programátor od programátora.

Z důvodu velmi značné subjektivity je toto poslední kritérium ohodnoceno vahou **5%.**

3. Framework

3.1. Co je to Framework

„Framework je softwarová struktura, která slouží jako podpora při programování, vývoji a organizaci jiných softwarových projektů. Může obsahovat podpůrné programy, knihovny API (Application Programming Interface), podporu pro návrhové vzory nebo doporučené postupy při vývoji.“ (Wikipedia foundantion, 2012a) Konkrétně v oblasti programování lze frameworkem označit jako nadstavbu nad programovacím jazykem jako takovým. Jedná se o sadu vhodně poskládaných komponent, které do sebe zapadají a navzájem na sebe navazují. Framework zpravidla zavádí nové klíčové výrazy. Nejsou tím myšleny klíčové výrazy programovacího jazyka jako takového, ale proměnné, které mají v daném frameworku ustálený tvar a voláním takové proměnné se stane konkrétní událost. Framework se může snažit i o lehké vylepšení jazyka jako takového zavedením striktnějších požadavků tam, kde by mohlo při volněji nastavených požadavcích docházet k nějakým druhům problémů.

3.2. K čemu slouží Framework

„Cílem frameworku je převzetí typických problémů dané oblasti, čímž se usnadní vývoj tak, aby se návrháři a vývojáři mohli soustředit pouze na své zadání.“ (Wikipedia foundantion, 2012a) Hlavním cílem frameworku je ulehčit programování tím, že framework se z pravidla stará o rutinní záležitosti a poskytuje vývojářům API jak jednoduše a efektivně řešit daný problém.

Pohled na framework se může lišit a záleží zejména na programované aplikaci a osobních preferencích programátora. Při vývoji menších a středně velkých projektů může být vnímán framework negativně, kvůli jeho náročnosti na čas zpracování, a často i napsání, kódu. Framework se snaží pokrýt velkou škálu problémů, které musí programátor řešit a poskytnout mu k tomu pohodlné API, čímž dochází k jeho nabobtnalosti a poměrně složité struktuře, která zpomaluje zpracování kódu. Tak široká škála, kterou framework pokrývá, bývá v menších projektech zbytečně veliká a nevyužitá. Výhody plynoucí z využití aplikace nepřeváží nevýhody v podobě pomalejšího zpracování kódu a nutnosti učení se nových příkazů.

Na druhou stranu pro velké projekty, a znovu opakující se aplikace, se použití frameworku přímo vybízí. Je to proto, že rozsáhlá aplikace využije veškerý potenciál nebo alespoň většinu možností frameworku a ten tudíž není zbytečně rozsáhlý. U znovu opakujících se aplikací je situace obdobná s tím rozdílem, že není využit veškerý potenciál, ale konstrukce mnohdy zjednodušuje často složité nebo jinak rozsáhlé konstrukce, které by musel jinak programátor řešit, např. zpracování formulářů.

3.3. Zend Framework

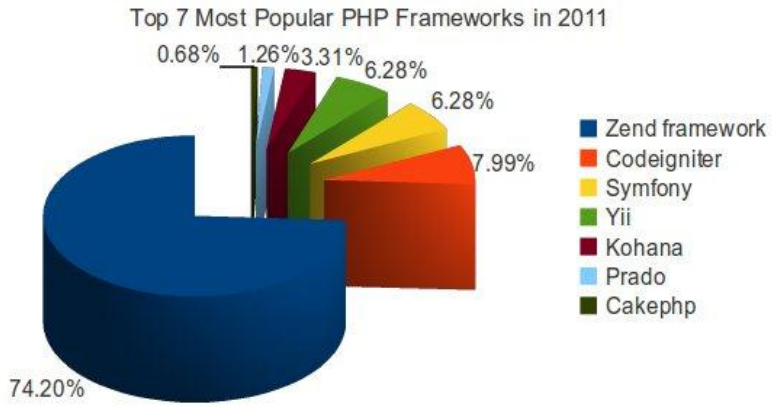
3.3.1. O Zend Frameworku

Zend framework je vyvíjen společností Zend Technologies Ltd., kterou založili Andi Gutmans a Zeev Suraski v roce 1997. Tito dva lidé často také přispívají k vývoji samotného jádra PHP a jsou mezinárodně uznávanými odborníky na poli PHP. (Zend Technologies Ltd., 2010)

Zend framework je objektově orientovaný a využívá mnoho z nových schopností PHP 5.3 jako jsou například namespace (jmenné prostory). Je také velmi zaměřen na bezpečnost a

testování, pomocí PHPUnit. Dovoluje vytvářet více bezpečné, spolehlivé a moderní aplikace s propojením na API třetích stran jako je Google, Amazon, Yahoo!, … (Zend Technologies Ltd., 2012a)

Zend framework byl v roce 2011 nejpoužívanějším frameworkem na světě. To je dáno silnou uživatelskou základnou, zavedeností a vyspělostí frameworku. Myšleno tak, že do finální verze se dostanou jen vlastnosti, které jsou 100%, nebo téměř 100%, kompatibilní s celou verzí. Na druhou stranu to znamená rigidní vývoj a Zend framework není tak moderní a pokrokový jako jsou jiné frameworky, které, ale z pravidla nebývají tolik stabilní. Z toho plyne, že Zend framework je často nasazován ve větších projektech, které upřednostňují stabilitu a komplexnost před novými funkcemi. (Codex-m, 2012)



Obrázek 1 Sedm nejpopulárnějších frameworků v roce 2011 [Zdroj: (Codex-m, 2012)]

3.3.2. Licencování

Zend framework využívá licenci new BSD (nová, modifikovaná, revidovaná či trojbodová BSD [Berkeley Software Distribution]), která patří mezi nejvolnější licence vůbec. Vznikla při obchodní organizaci University of California, Berkeley, která tuto licenci vyvinula pro používání s operačním systémem BSD. Licence new BSD dovoluje využívat software téměř bez omezení, prakticky je nutné pouze v programu uvádět jména autorů a zřeknout se

odpovědnosti. BSD licence dovoluje dokonce komerční využití, kdy nemusí být zveřejněny zdrojové kódy aplikace. (Wikipedia foundantion, 2012b)

3.3.3. Verze

Aktuální stabilní verze frameworku je 2.0.5, která vyšla 29. listopadu 2012 (Zend Technologies Ltd., 2012b). Tato nová verze Zend framework 2(ZF2), obsahuje spoustu novinek oproti verzi 1.x.x. Verze ZF2 je vyvíjena od roku 2010, kdy 6. Srpna byla vydána vývojová verze (development release). Tato první vývojová verze odstraňovala `require_once` výrazy, zavedla jmenné prostory PHP 5.3, přepsala `Zend\Session`, refaktorovala testovací sadu a další. V dalších verzích byly přidány nové vlastnosti, jako například snížení počtu singleton (jedináček) tříd. Další změny a postupy jsou diskutovány s komunitou Zend frameworku. (Wikipedia foundantion, 2012c)

3.4. Nette Framework

3.4.1. O Nette frameworku

Nette framework je vyvíjen organizací Nette Foundation, která navazuje svou činností na Davida Grudla, jenž je původním autorem Nette frameworku. Nette framework je český projekt a jeho výhodou i nevýhodou je dokumentace i komunita komunikující převážně v češtině. Nevýhodou je to proto, že se tím komunita okolo Nette stává užší. V minulosti byl problém s jednotlivými verzemi. Mezi verzí 0.9 a 2.0 byl vývoj Nette frameworku značně turbulentní. Problém byl zejména v tom, že nové verze nebyly zpětně kompatibilní a byla tudíž nutnost přepsat celý kód aplikace, při přechodu na novější verzi.

3.4.2. Licencování

Licencování je stejně jako u Zend frameworku, tedy new BSD. Nette framework ještě nabízí, jako variantu pro vývojáře použití licencování GNU GPL (General Public Licence). Tato licence je v podstatě pravým opakem copyrightu. (Nette foundantion, 2012a) Říká, že „Při vytvoření odvozeného díla z díla, jež je dostupné jen pod GNU GPL licencí, musí být toto odvozené dílo nabízeno pod stejnou licencí jako dílo původní“. (Wikipedia foundantion, 2012d)

3.4.3. Verze

První verzí Nette frameworku, jenž lze stáhnout z oficiálních stránek je verze 0.7. Dlouhou dobu, byla poslední stabilní verze 0.9, ale 2. Února 2012 byla vydána nová stabilní verze 2.0. Aktuální verze je 2.0.7, která byla vydána 28. Listopadu 2012. Tato verze je ve dvou verzích jak s podporou PHP 5.3, které dovolují užití jmenných prostorů, tak staršího PHP 5.2 bez jmenných prostorů. (Nette Foundation, 2012b)

3.5. Závěr

Mohlo by se říci, že v tomto srovnání jde o zápas „Davida s Goliášem“. Zend framework patří k nejrozšířenějším frameworkům na světě s ohromným množstvím funkcí, pluginů (zásuvný modul) a API třetích stran. Proti němu stojí Nette, vyvíjené donedávna jedním programátorem, které je rozšířené téměř pouze v česky mluvících zemích s mnoha vylepšeními a „vychytávkami“. Pro Zend mluví stabilní zázemí pro vývoj stabilních aplikací. Nette se snaží zaujmout spoustou originálních funkcí a mnohdy se snaží o vylepšení samotného jazyka PHP.

4. Návrhové vzory (patterns)

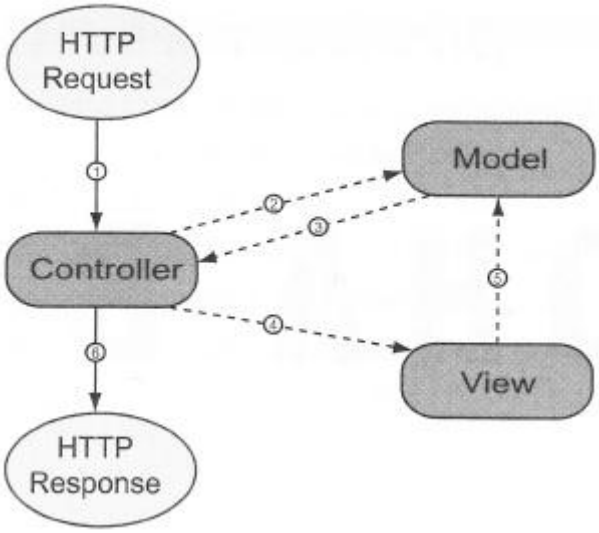
Jak komponenty Nette tak Zend frameworku používají množství návrhových vzorů. Pomocí nich lze řešit problémy, které se pravidelně vyskytují při vývoji aplikace. Jedná se o ustálenou šablonu, která daný problém řeší velmi efektivně. Výhodou je i názvosloví, které usnadňuje komunikaci vývojářů mezi sebou. Níže je popsán je návrhový vzor MVC a jeho derivát MVP.

4.1. Návrhový vzor MVC

MVC je zkratkou pro model, view, controller. Tento návrhový vzor používá drtivá většina webových frameworků. Hlavní myšlenkou tohoto frameworku je rozdělení těchto tří oblastí do třech logických celků (komponent): údaje (model), řadič (controller), pohled (view). Každá z těchto komponent se stará o určitou část aplikace. Protože tento návrhový vzor používají oba frameworky (Nette používá MVP [viz. níže]) a je to vlastně jejich jádro tak bude MVC architektura vysvětlena více do hloubky.

4.1.1. Princip MVC

Celý proces je odstartován HTTP požadavkem (HTTP request), který je směřován na Controller, který daný požadavek zpracuje. Pokud je potřeba vykomunikuje s modelem potřebná data, která uloží nebo načte. Poté kontaktuje view a předá mu data k zobrazení. Pohled nakonec vygeneruje výpis údajů (nejčastěji HTML stránku) a controller ho pošle do prohlížeče uživatele jako HTTP odpověď (HTTP response). (Böhmer, 2010)



Obrázek 2 Průběh požadavku v MVC architektuře [Zdroj: (Böhmer, 2010)]

1. Celý proces je odstartován přijetím HTTP požadavku na server, kde je dále zpracováván.
2. Z požadavku controller zjistí, jestli a který model má být osloven a vyžádá, případně předá mu data.
3. Model, buď odpoví, jestli bylo uložení dat úspěšné, respektive neúspěšné, nebo vrátí data, která si controller vyžádal.
4. Controller zjistí, na který view, má poslat data k vykreslení a pošle je.
5. V závislosti na implementaci může view kontaktovat model a požadovat data nebo informovat o změně dat.
6. Na konec vytvoří view výstup, který pošle controller do prohlížeče v podobě HTTP odpovědi.

Bod 5 je poměrně diskutovanou částí MVC. Existují dva názory na to, jestli může mít view přístup k modelu, byť jen ke čtení nebo nikoli. View by rozhodně nikdy neměl

přistupovat přímo k modelu za účelem zápisu do něj. Může však například obdržet primární klíč (id) tabulky a zbytek dat pomocí něj z modelu natáhnout. Z pravidla se však používá varianta bez přístupu view k modelu. Controller načte veškerá data z modelu a ty předá view k vykreslení. View se potom může plně soustředit na svou hlavní činnost, a sice vykreslování výstupu. (Böhmer, 2010)

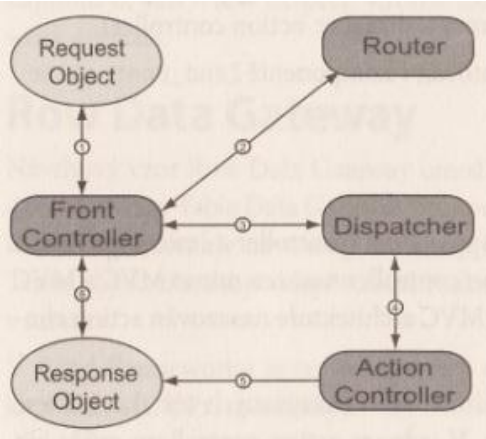
4.1.2. Pohled (view)

Pohled je ta část aplikace, která se stará o zobrazení předaných dat od controlleru uživateli. Nemusí vědět, odkud data pochází, ale musí znát formátování předaných dat. Data, která předává controller view mohou být de facto objekt nebo pole. U objektu musí view znát metody, které vracejí hodnoty atributů. Pokud je předáno view pole, které implementuje návrhový vzor Row Data Gateway, musí view větět jak jsou předaná data přístupná. Jestli jsou přístupná přes jednu metodu, nebo jestli má každý atribut svoji „get“ metodu například `$object->getTitle();`. Některé atributy mohou být přístupné také přímo voláním `$object->title;`. (Böhmer, 2010)

4.1.3. Kontrolor (controller)

Kontrolor je zodpovědný za řídící vrstvu v MVC architektuře. Kontrolor přijme požadavek HTTP (request) od prohlížeče uživatele. Na základě požadavku zjistí, který model má být kontaktován a který view má vygenerovat výstup. Až je vše poskládáno, odešle zpět do prohlížeče odpověď (HTTP response).

Kontrolor se v aplikaci skládá z více dílčích komponent. Tyto komponenty jsou tvořeny dalšími návrhovými vzory jako Action Controller, Front Controller, Command pattern (vzor Příkaz). Nejdůležitější je Front controller, který přijme požadavek, zpracuje ho a komunikuje s dalšími komponentami na základě daného požadavku. (Böhmer, 2010)



Obrázek 3 Průběh požadavku v controlleru
[Zdroj: (Böhmer, 2010)]

Průběh požadavku v Controlleru:

1. HTTP požadavek je zabalen v request objektu. Díky tomu je možné k datům v tomto objektu přistupovat skrze tento centrální objekt. V průběhu zpracování mohou k request objektu přistupovat router, dispečer (dispatcher) i action controller a to jak pro čtení tak pro zápis.
2. V tomto kroku předá controller zpracování routeru. Ten rozhodne, které akce budou vykonány
3. Poté je controller pověřen dispečerem, aby vykonal další zpracování požadavku.
4. V rámci cyklu volá dispečer jednotlivé action controllery, které mají vykonat uživatelem požadované akce. Tyto action controllery mohou případně kontaktovat model.
5. Výsledek je z action controllerů předán objektu odpovědi (response object)
6. Nakonec se dostane ke slovu opět front controller, který vyzve response objekt, aby odeslal odpověď do prohlížeče ve formě HTTP odpovědi (response).

4.1.4. Model

Třetí a poslední část architektury je zodpovědná za datovou vrstvu aplikace. Stará se o zápis a načítání dat. V jaké formě jsou data ukládána a zpracovávána ví jen model. View ani Controller o systému ukládání a načítání dat neví nic a ani vědět nepotřebují, jestliže si mohou být jisti, že model se o to bezpečně postará.

Důležité je si uvědomit, že model !== databáze! Občas to bývá zaměňováno, protože aplikace nejčastěji používá k ukládání dat relační (či objektové) databáze. Model však slouží

k ukládání dat v jakékoli formě, může tedy jít o data uložena do databáze, jako textový soubor, RSS, webová služba (Google, Yahoo!, Flicker, atd.), … (Böhmer, 2010)

4.2. MVP

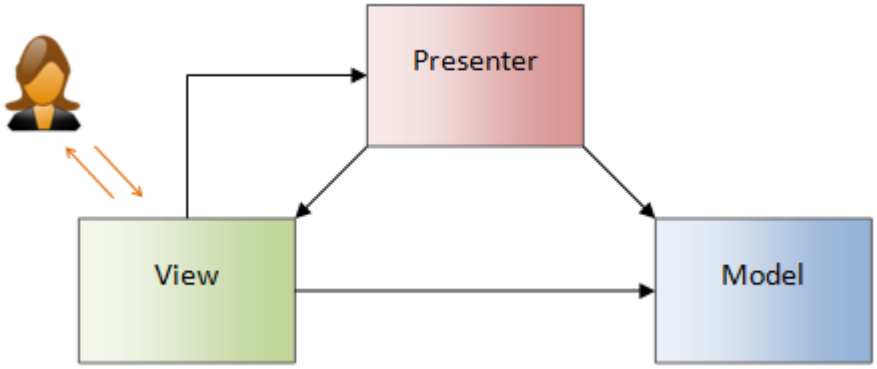
Nette framework nepoužívá MVC přímo, ale jeho variaci. Používá totiž MVP architekturu, která je téměř stejná jako MVC, kde controller je nahrazen presenterem. Presenter má trošku jiné vlastnosti než controller. View v jednotlivých frameworcích má také trochu jiné funkce.

4.2.1. View

Jestliže v MVC zpracovával požadavek controller a view zobrazoval výsledek v MVP to tak není. V MVP view zpracovává jak požadavek, tak odpověď. Uvnitř by neměla být žádná aplikační logika a view pouze deleguje jednotlivé uživatelské akce. V závislosti například na zmáčknutí tlačítka zavolá patřičnou metodu presenteru a předá jí uživatelovu akci. (Zdoják.cz, 2009)

4.2.2. Presenter

Presenter obsahuje jak aplikační tak prezentační logiku. Komunikuje s modelem a pomocí notifikací předává view k zobrazení. To může předat jednoduše, protože má typicky přímou vazbu na view a obráceně view má přímou vazbu na presenter. Vazba mezi presenterem a view je obecně silnější u MVP, než u MVC. (Zdoják.cz, 2009)



Obrázek 4 MVP architektura [Zdroj: (Zdoják.cz, 2009)]

4.3. Závěr

MVC, respektive MVP architektura, která je standardem ve vývoji webových stránek. Je založena na striktním rozdělení aplikace na tři vrstvy, přičemž každá vrstva se stará o něco jiného. Toto členění je důležité zejména kvůli bezpečnosti, kdy pomocí `.htaccess` (konfigurační soubor webového serveru Appache) lze omezit vstup jen do některých adresářů. Případný útočník se tak nedostane přes http protokol do žádného kritického adresáře.

5. Základní srovnání

5.1. Základní srovnání

Oba frameworky si jsou velmi podobné. Svým způsobem se nechá říct, že Nette ze Zendu vychází. Takové základní porovnání je velmi časté, ale jako u porovnání rychlosti, většinou nezohledňuje Nette a Zend framework ve verzi 2. Podobné, avšak neaktuální, srovnání lze nalézt na webové stránce Best web frameworks (Bestframeworks.com, 2011). Toto srovnání jsem vzal jako základ, aktualizoval a rozšířil. Výsledkem je následující tabulka, která shrnuje v bodech jednotlivé rysy obou frameworků. Uvádí pouze, jestli danou technologii/funkcionalitu frameworky obsahují s vysvětlením jednotlivých bodů pod tabulkou.

Jméno	Zend framework	Nette framework
PHP4	Ne	Ne
PHP5	Částečně	Ano
MVC	Ano	Ano
Moduly	Ano	Ano
ORM	Ano	Ano (Nette foundantion, 2012c)
EDP	Ne	Ano
Autentizace	Ano	Ano
Cache	Ano	Ano
Validator	Ano	Ano
Více databází	Ano	Ano
DB objekty	Ano	Ano
Šablony	Částečně	Ano
Licence	New BSD	New BSD, GPL

Tabulka 1 Základní srovnání obou frameworků [Zdroj: (Bestframeworks.com, 2011)]

- PHP4 – Ukazuje, jestli framework podporuje starou verzi php verze 4
- PHP5 – Oba frameworky podporují PHP verze 5.3 se jmennými prostory (namespace). Pouze Nette podporuje PHP 5.2 bez jmenných prostorů.
- MVC – Jestli je framework vystavěn na model-view-controller návrhovém vzoru (respektive MVP)
- Moduly – Jestli framework podporuje modulární tvorbu aplikací. V Nette je tvorba modulů volitelná. Zend 2 je na modulech přímo založen. Předchozí verze podporovala moduly pouze volitelně.
- ORM – ORM (Objektově relační mapování) – jedná se o „programovací techniku, která zajišťuje automatickou konverzi dat mezi relační databází a objektově orientovaným programovacím jazykem.“ (Wikipedie foundantion, 2012e) Nette používá Nette/Database, které přímo vychází z notORM, které jako open source (volně šiřitelný software) napsal Jakub Vrána (2010).

- EDP – EDP (Event-driven programming [událostmi řízené programování]) – programování, které je řízené nějakou uživatelovou akcí, např. kliknutí na objekt, zmáčknutí klávesy, … Zend nepoužívá EDP, ale RDP (Request-driven programming [požadavkem řízené programování]) (Stack exchange inc, nedatováno).
- Autentizace – Jestli framework obsahuje modul/komponentu pro autorizaci a autentizaci uživatele na webové stránce.
- Cache – Jestli framework umožňuje kešování. Po uložení kódu do cache se kód znovu nevykonává a načte se přímo z cache. To zapříčiní velkou změnu v rychlosti načítání stránek. Je to vhodné pro části aplikace, které jsou delší dobu neměnné, jako je menu, acl tabulka, články, …
- Validátor – Jestli má framework vestavěnou validaci nebo filtrování komponent
- Více databází – Pokud framework může používat více databází bez zásahu do kódu. Typicky se jedná o určení databáze pro vývoj (na localhostu) a pro produkční prostředí na serveru.
- Šablony – Jestli má framework implementované šablony. Nette má vlastní šablonovací systém Latte. Zend má také, podle definice webového šablonovacího systému (Wikipedia foundantion, nedatováno), šablonovací systém, avšak já ho za šablonovací systém, tak jak ho chápu já, nepovažuji. Důvodem je, že nepřidává žádnou novou syntaxi a práci v podstatě programátorovi žádným zásadním způsobem neusnadňuje (viz. dále). Podle výše uvedené definice je v podstatě šablonovací systém i samotné PHP.
- Licence – Pod jakou licencí je framework vydáván. Licence jsou vysvětleny výše v 3.3.2 respektive 3.4.2.

5.2. Závěr

Toto srovnání není součástí metodiky porovnávání. Jedná se pouze o bodový výčet vlastností obou frameworků, které sami o sobě o síle či použitelnosti frameworku nic neříkají. Z tohoto porovnání vychází, že se jedná na první pohled o dva velmi podobné frameworky, kterými na pohled druhý nemusí být. Každý má trochu jiné vlastnosti a pravděpodobně i trochu jiné cíle.

6. Bezpečnost

Bezpečnostní stránka aplikace je nejdůležitější ze všech porovnávaných hledisek. Frameworky byly porovnávány v produkčním módu, tzn., nevypisovaly chybové hlášky při chybě.

6.1. Cross site scripting (xss)

Jedná se o útok, kdy se do webové stránky vloží cizí HTML kód. Tento kód se poté interpretuje jako HTML značka a zobrazí se tak, jak má. Toto chování je velmi nebezpečné, protože je poté zejména možnost přesměrovat JavaScriptem uživatele jinam a tam provést útok. Jedná se o vůbec nejznámější útok, pravděpodobně proto, že je velmi snadné ho provést a útočník nepotřebuje téměř žádné znalosti.

6.1.1. Nette

Proti tomuto útoku obstálo Nette bez jediného problému. Velkou výhodou je, že se Nette stará o escapování (vypsání proměnné tak, aby se kód neprovedl) samo. V praxi je tak po napsání pro Nette běžného příkazu pro vypsání proměnné `{$variable}`, která byla předána metodou GET, výraz už escapován. Pokud je potřeba, je možné escapování vypnout '!', tzn. Příkazem `{!$variable}`. Po vypnutí se HTML provede tak, jak má.

6.1.2. Zend

Zend framework byl v tomto horší. Po běžném vypsání proměnné `<?php echo $variable; ?>` se kód neescapoval. Nicméně testovací útok vložením se mě nepodařil také. Na rozdíl od Nette Zend vzal každé zpětné lomítko '/' jako další parametr pro router a snažil se najít další routu za dalším lomítkem. Použití útoku bez zpětného lomítka nicméně proběhlo bez problému. Pro tento případ jsem použil kód `<div style="visibility: hidden">`.

Zend také disponuje escapovací funkcí a sice `$this->escape($variable)`. Nicméně lze na volání této funkce zapomenout a snadno tak vytvořit bezpečnostní díru na webu.

Další místo, kde se může zabránit útoku, je v routeru. Router totiž umožňuje napsat omezení, jaké znaky se mohou použít pro jednotlivé parametry routy jako je controller, akce a hlavně parametry typu id a jiné proměnné, předávané metodou GET. Pokud byly uvedeny podmínky `'[a-zA-Z][a-zA-Z0-9_-]*'`, tak útok nešel provést, protože mezi povolenými

znaky nebyl znak menší/větší ‘<, >’. Těžko říct, zdali se jedná o ochranu, protože stránka se ukáže jako nenalezena a požadavek návštěvníka je tedy nevyřízen. Na druhou stranu nehrozí žádné větší riziko.

6.1.3. Zhodnocení

Nette se zde chová tak, jak bych očekával od dospělého a dobře použitelného frameworku. Sám za programátora escapuje proměnné, takže na žádnou nemůže zapomenout, ale když je potřeba vypsat neescapovanou proměnnou, tak to lze velice jednoduše. Naproti tomu Zend nechává všechnu práci na programátorovi, který musí myslet na to, aby všechny proměnné vyescapoval.

Protože Nette escapuje samo a dovoluje escapování vypnout, což je asi nejběžnější vyžadované řešení, dostává plný počet bodů. Zend je v přesně opačné pozici, sám neescapuje, ale dovoluje escapování zapnout. Jedná se tedy u programátora o práci navíc, na kterou když zapomene, tak udělá aplikaci s bezpečnostní dírou.

Jméno	Zend framework	Nette framework
Automatické escapování	0	4
Možnost escapování	3	3
Vážné hrozby	7	7
Celkem	10	14

Tabulka 2 Porovnání bezpečnosti proti XSS útoku [Zdroj: autor]

- Automatické escapování – jestli framework automaticky escapuje výstup
- Možnost escapování – jestli má framework implementované vlastní metody pro escapování proměnných
- Vážné hrozby – Když při programování programátor nemyslí na bezpečnost, jestli framework ochrání „sám“ aplikaci proti vážným hrozbám typu, přesměrování na jiné servery

- Nette framework: **14 b**
- Zend framework: **10 b**

6.2. SQL Injection

Tento útok se snaží manipulovat přímo s databází. Jedná se o nebezpečnější útok, než útok XSS, protože může získat veškerá data z databáze nebo veškerá data smazat. Zatímco xss je rizikem zejména pro uživatele, kterému může ukrást identitu, SQL Injection může získat přístupové údaje všech uživatelů (i když zašifrované, pokud byly zašifrovaně uloženy v databázi).

6.2.1. Nette/Zend

V tomto testu uspěly oba frameworky skvěle. Ani přímým vložením SQL kódu do ukládací funkce se nepovedlo nabourat do databáze. Oba frameworky se chovaly totožně a sice uložily kód tak, jak byl vložen, čili jako prostý text.

Obrázek 5 Náhled pokusu o útok SQL Injection

6.2.2. Zhodnocení

Zde není moc co hodnotit, oba frameworky toto kritérium zvládly bez problémů, tedy obdržely plný počet bodů.

- Nette framework: **14,5 b**
- Zend framework: **14,5 b**

6.3. Cross Site Request Forgery

Poslední útok, který byl proveden, byl útok CSRF. Jde o asi nejzákeřnější útok v testu. Princip spočívá v tom, že se na nějaké webové stránce vloží odkaz do stránky aplikace, a pak se jen počká, až na ten odkaz někdo klikne nebo se spustí sám.

6.3.1. Nette

Do obrázku byl vložen jako útočný odkaz `/Bakalarka/Nette/kucera/admin.product/detail?productId=43&productListId=1&detailId=43&do=deleteProduct`, který se vykonal v pořádku. Nette tedy není chráněno proti tomuto typu útoku formou GET. Poskytuje však ochranu proti útoku na formuláře, který funguje bezvadně. V presenteru se zavolá metoda třídy **Form** a sice metoda `$form->addProtection('zprava');`. Tato metoda zajistí, že se do formuláře vloží neviditelný input element, který obsahuje token (kus vygenerovaného kódu). Tento token se generuje pro celou aplikaci. Token je možné použít, i pokud je otevřených více záložek v prohlížeči. Takto ochráněný formulář odolal útoku.

6.3.2. Zend

Průběh testu v Zendu měl stejný průběh. Útok na smazání opět prošel, vložením jako hodnoty src atributu ekvivalent odkazu pro Nette. I Zend má metodu pro ochranu formulářů proti CSRF útoku, která se používá téměř stejně jednoduše. Stejně jako Nette i Zend využívá tokeny, které jsou generované, ale na rozdíl od Nette nejsou generované pro celou aplikaci, ale při každém načtení stránky je vytvořen nový token. I tímto způsobem vytvořený token funguje bez problému ve více záložkách. Při vytváření formuláře se zavolá `$this->add(new Element\Csrf('security'));` a při výpisu formuláře uvnitř view se vykreslí: `echo $this->formHidden($form->get('security'));`.

6.3.3. Zhodnocení

Tímto testem prošly oba frameworky opět téměř bezchybně. Otázkou je, jestli není lepší vykreslovat token automaticky a na přání vypnout. Ne všude je to nutné, ale čas pro zpracování tokenu by možná stál za problémy, které mohou být způsobeny úspěšným útokem. Oba obstály stejně úspěšně v testu, proto také dostanou plný počet bodů. Proti útoku z „obrázku“ na GET parametr se brání jen velmi obtížně, proto nebyly strhávány body za úspěšný útok tímto způsobem.

- Nette framework: **14,5 b**
- Zend framework: **14,5 b**

6.4. Závěr

Na závěr lze oba frameworky považovat za přibližně stejně bezpečné. Nette framework je z hlediska pohodlnosti o dost přívětivější nežli Zend, protože automaticky escapuje všechny hodnoty. Na druhou stranu, kromě CSRF útoku na GET metodu předávání proměnných, nebyl žádný z útoků nějakým způsobem fatální a to ani XSS proti Zendu a jeho neescapovaným proměnným. Z hlediska bezpečnosti jsou tedy téměř na stejné úrovni.

- Nette framework: **43 b**
- Zend framework: **39 b**

7. Rychlost aplikace

Rychlost aplikace je hodně důležitá, zejména je-li aplikace velmi rozsáhlá a načítá mnoho souborů ze serveru. Mnou testovaná aplikace je malého rozsahu, ale při porovnání některé opravdu veliké aplikace by se tyto zdánlivě malé rozdíly mnohem zvětšily. Jelikož server běžel na poměrně výkonném stroji a dá se usuzovat, že výkonost serverů je různá, tak nejdůležitějším parametrem výsledku je procentuální rozdíl mezi Nette a Zend frameworkem.

Nette framework si zakládá na tom, že patří mezi nejrychlejší frameworky vůbec. Vzhledem k tomu, že cílem práce je porovnat pouze framework Zend a Nette, tak nemohu rozhodnout, jestli tomu tak je. Mohu ho pouze porovnat se Zend frameworkem. Zend framework si oproti Nette nezakládá na rychlosti, ale spíše na stabilitě.

7.1. Homepage

Prvním testem byl test na úvodní stránku, která obsahovala jak statický text, tak dynamicky generované menu. Toto menu bylo poté na všech stránkách vykresleno taktéž. Výsledek se téměř shoduje s celkovým výsledkem měření, čili se jedná o průměrný výsledek.

Výsledná tabulka Homepage

	vzorky	průměr [ms]	medián[ms]	Min[ms]	Max[ms]	Sm. Odchylka [ms]	Modus[ms]
Nette	90	149,3111111	148,3333333	137,6666667	172	7,860280067	145,6666667
Zend	90	212,2333333	210,6666667	191,6666667	246,3333333	12,82007004	203,6666667
Procentuální rozdíl	**90**	**42,14%**	**42,02%**	**39,23%**	**43,22%**	**63,10%**	**39,82%**

Tabulka 3 Porovnání rychlosti na úvodní obrazovce [Zdoj: autor]

Procentuální rozdíl vždy udává, o kolik procent byl Zend pomalejší než Nette. Tato výsledná tabulka zaznamenává jen průměrné hodnoty, pro detailnější informace jsou všechny hodnoty v příloze 1 a 2. Nejdůležitější parametr je samozřejmě průměr, kdy se jedná o průměrnou hodnotu všech měření na úvodní stránce. Dalším zajímavým jevem je, že Nette nebylo nikdy pomalejší než Zend. I nejvyšší hodnota (187) byla nižší, než nejnižší (188) hodnota Zendu.

7.2. SELECT uživatelů

Druhý test byl proveden po přihlášení na výpis uživatelů, který nebyl omezen. V databázi bylo pro potřeby testu 20 záznamů o fiktivních uživatelích aplikace. Kvůli rozdílné implementaci je možné, že byl tento test zkreslen a znamenalo by to, že Nette je oproti Zendu ještě rychlejší. Nicméně, tento test vykazoval nejmenší rozdíl v testu rychlosti a to i tak poměrně vysokých téměř 20%. Více informací v příloze 3 a 4.

Výsledná tabulka SELECT

	vzorky	průměr [ms]	medián[ms]	Min[ms]	Max[ms]	Sm. Odchylka [ms]	Modus[ms]
Nette	90	182,5111111	180,6666667	168,3333333	209,3333333	8,40795766	179,6666667
Zend	90	217,8666667	215,3333333	197,6666667	247,6666667	12,53790258	212,6666667
Procentuální rozdíl	**90**	**19,37%**	**19,19%**	**17,43%**	**18,31%**	**49,12%**	**18,37%**

Tabulka 4 Porovnání rychlosti na příkaz SELECT [Zdoj: autor]

7.3. EDIT produktu

Poslední test byl zaměřen na editování jednoho produktu. V tomto testu bylo změřeno odesílání na server metodou POST i přijímání a zpracování http souboru. Níže v příloze 5 - 8 jsou uvedeny přesné časy, jak odesílání, tak přijímání a samozřejmě součet obou časů, který je výsledným. Jedná se vlastně o nejnáročnější test, kdy je potřeba rychle zpracovat data poslaná prohlížečem a na jejich základě sestavit a odeslat HTTP odpověď. Tento test také vykazuje největší rozdíl mezi Nette a Zend. Nejhorší čas Nette je dokonce o čtvrtinu lepší než nejlepší čas Zendu.

Výsledná tabulka EDIT

	vzorky	průměr [ms]	medián[ms]	Min[ms]	Max[ms]	Sm. Odchylka [ms]	Modus[ms]
Nette	90	302,3333333	302,6666667	285	322,3333333	9,1466653	301,6666667
Zend	90	482,1555556	480,6666667	462,3333333	516	12,81387772	474,3333333
Procentuální rozdíl	**90**	**59,48%**	**58,81%**	**62,22%**	**60,08%**	**40,09%**	**57,24%**

Tabulka 5 Porovnání rychlosti EDIT [Zdoj: autor]

7.4. Závěr

Na závěr je možné říci, že Nette je o téměř 44% rychlejší než Zend. Pokud by se výsledky oprostily od již výše zmíněného výkyvu Nette frameworku v SELECT testu, tak by byl rozdíl dokonce téměř 54%. Pokud by se tedy programátor rozhodoval, který framework se bude učit, tento faktor hraje jednoznačně pro Nette.

Výsledná tabulka

	vzorky	průměr [ms]	medián[ms]	Min[ms]	Max[ms]	Sm. Odchylka [ms]	Modus[ms]
Nette	270	211,39	210,56	197,00	234,56	8,47	209,00
Zend	270	304,09	302,22	283,89	336,67	12,72	296,89
Procentuální rozdíl	**270**	**43,85%**	**43,54%**	**44,11%**	**43,53%**	**50,19%**	**42,05%**

Tabulka 6 Porovnání rychlosti - výsledná tabulka [Zdroj: autor]

Bodové ohodnocení je následující: Zend získává plný počet bodů, jelikož byl vždy nejrychlejší. Zend získá jen 56,25% bodů (zaokrouhleno na celé body).

- Nette framework: **32/32 b**
- Zend framework: **18/32 b**

8. Dokumentace

Dokumentace je pro začínajícího programátora ve frameworku první místo, které navštíví hned po stažení aplikace. Je tedy poměrně důležité mít přehledně napsanou dokumentaci, která neodradí programátora od použití frameworku, který si vybral.

8.1. První aplikace (Getting started)

První kroky zpravidla vedou na stránku s první aplikací (tutoriálem), kde jsou na několika stránkách představeny hlavní části frameworku na ukázkové aplikaci. Podle mého názoru by tento tutoriál měl jednoduše představovat všechny důležité, a zároveň velmi často používané, části frameworku pro jeho správnou funkčnost.

8.1.1. Nette framework

Po otevření První aplikace (quick start) se objeví na stránce seznam témat, tak jak jdou po sobě. To je užitečné, protože začínající programátor má představu co ho čeká v tutoriálu. Návod je psán pro Nette framework 2.0.5 a PHP 5.3+. Není to sice pro nejaktuálnější verzi Nette, ale látka probíraná v tutoriálu se od verze 2.0.5 nezměnila. Pouze v nejnovější verzi 2.0.7 přibyl adresář router, který obsahuje tvorbu rout.

Tutoriál začíná tradičním stažením a zprovozněním ukázkové kostry (v Nette nazývané sandbox), kde jsou informace o Nette s odkazy na dokumentaci. Poté seznamuje s `Nette/Debuger`em (Laděnka). Laděnka je velmi silný nástroj pro vývojáře, který mu je schopný ukázat srozumitelný chybový kód se zvýrazněním místa, kde k chybě došlo a spoustu nastavení PHP a serveru. Poté tutoriál seznamuje se základními částmi frameworku, kterými jsou: model (a databáze), presenter a šablonovací systém Latte, formuláře, komponenty, přihlašování uživatelů a AJAX. V přípravě je routování a bezpečnost při finálním nasazením aplikace.

Tato skladba je velmi dobrá, nicméně není ideální. Z hlediska důležitosti si myslím, že důležitější je presenter než model, ale v tomto tutoriálu presenter na model navazuje. Dále nepovažuji AJAX za základní funkcionalitu aplikace (ačkoli Nette si zakládá na tom, že je velmi jednoduše zajaxovatelné). Zajaxování aplikace je dle mého názoru v tutoriálu nadbytečné. Místo AJAXu by bylo mnohem přínosnější dokončit a rozšířit tutoriál o routování. Jinak není co vytknout. Model i presenter jsou nepostradatelnými součástmi a dále

následují části, které jsou použité na převážné většině webů, a sice formuláře a přihlašování uživatelů. Tutoriál dále seznamuje s komponentami, což jsou v Nette znovupoužitelné části kódu.

Programování podle tutoriálu proběhlo téměř bez problému. Tutoriál se nevyhnul několika málo překlepům. Zásadním je jen metoda `find()`, se kterou presenter počítá, ale v modelu není naprogramována. Tento problém je nicméně vyřešen v komentářích. Poněkud zarážející je, že Nette a zejména jeho autor David Grudl prosazují Dependenci Injection (DI - vkládané závislosti), ale tutoriál je napsán bez DI a model je předáván presenteru ve formě služby.

8.1.2. Zend framework

Zend má stejně jako Nette seznam po sobě následujících témat. Ačkoli je jaksi nelogické, že je tento seznam vidět ze stránky s dokumentací, ale na stránce tutoriálu (getting started) už ne. Dokumentace je psaná pro Zend framework 2.0.5, což je také nejnovější verze frameworku.

Tutoriál v Zendu začíná požadavky na server, které jsou vyžadovány a následuje popisem instalace a nastavení serveru, tak aby se programátorovi zobrazila kostra aplikace (v Zendu: A skeleton appliaction). Dále následuje podobný popis částí frameworku jako u Nette a sice: jednotkové testy, moduly, routování a controllers, databáze a modely, překlady a na závěr formuláře.

Příjemným, nikoli však nutným bonusem jsou jednotkové testy. Jedná se o stručný popis, jak je implementovat v Zendu. Řekl bych, že jednotkové testy jsou tak na hraně toho, co by v tutoriálu mělo být. Nejsou přímo vyžadované pro běh aplikace, avšak například pro firmy, které používají vývojový proces test-driven development (testy řízené programování), se to více než hodí. Zend je mnohem více modulárně zaměřený, proto je naprosto v pořádku začlenění popisu modulů do tutoriálu. Poslední věcí, kterou bych zmínil, jsou překlady. Nemyslím si, že vývoj vícejazyčné webové aplikace je nutné popisovat v tutoriálu. Chápu, že na tuto funkcionalitu chtěli autoři dokumentace pro Zend framework upozornit, ale podle mého názoru je tato část v tutoriálu nadbytečná, podobně jako AJAX v tutoriálu Nette.

Programování podle tutoriálu u Zendu vykazovalo více chyb/problémů, než u Nette. Zejména část o modelech a databázích. Po projití komentářů bylo jasné, že nejsem sám a že

více a různých problémů měli i ostatní. Nicméně nakonec i zde bylo vše objasněno natolik, že aplikace byla podle tutoriálu spuštěna.

Jméno	Zend framework	Nette framework
Seznam částí	Ano	Ano
Instalace	Ano	Ano
Modely	Ano	Ano
Presentery/Controllers	Ano	Ano
Formuláře	Ano	Ano
Přihlašování uživatelů	Ne	Ano
Routování	Ano	Ne
Bezproblémovost	1b	2b
Celkem bodů	7/10	8/10

Tabulka 7 Porovnání tutoriálu [Zdroj: autor]

Oba frameworky obsahují v tutoriálu poměrně vyčerpávající informace. Ani jeden z nich nicméně neobsahuje veškeré informace, které považuji za nejdůležitější pro vývoj webové dynamické aplikace. Na druhou stranu, oba přidávají něco, co si myslí, že je pro jejich framework velmi důležité a chtějí to zdůraznit.

8.2. Příručka (reference)

Další krok, po naprogramování první aplikace, vede k jednotlivým částem dokumentace a prozkoumávání jejích částí do hloubky.

8.2.1. Nette framework

Nette framework má jednu výhodu i nevýhodu zároveň. Neoddiskutovatelnou výhodou pro české programátory je, že je psán v češtině. Pravděpodobně to je také důvod jeho zdejší rozšířenosti. Na druhou stranu je čeština jeho největší nevýhodou, protože dokumentace anglické verze za českou o něco zaostává a hlavně komunita mnohem raději komunikuje v češtině na českém fóru, než na jeho anglické mutaci. Kompletní dotažení anglické dokumentace a hlavně komunita, která by více komunikovala v angličtině je to, co by framework potřeboval k rozšíření i do zahraničí.

Předností Nette dokumentace je jeho přehlednost. Kdy je dělen na 32 základních částí. Tyto části po rozkliknutí obsahují ucelené informace o právě rozkliklé části. Pro lepší orientaci je po pravé straně ještě rozdělení na dílčí menší celky, které odkazují na konkrétní místa přímo na stránce. Bohužel ne vše, je obsaženo v dokumentaci a dokumentace je tedy nevyčerpávající. Ukazuje zpravidla jeden preferovaný způsob řešení problému a maximálně nastíní další způsoby, které jsou možné použít. Tento preferovaný způsob je naprosto korektní a vede programátora k používání best practices, což je velmi správně. Pokud je ale potřeba problém řešit jiným způsobem, odkazuje programátora už jen na studování API nebo na komunitu okolo Nette.

Za poslední léta se dokumentace Nette velmi zlepšila. Dokumentace psaná k Nette 0.9 byla velmi kritizována, zejména z neaktuálnosti. V této době totiž Nette zažívalo bouřlivý vývoj a dokumentace byla často neaktuální a zpětně nekompatibilní. Dnes je téměř aktuální. Nemění se sice hned s vydanou aktualizací frameworku, ale framework je z velké části zpětně kompatibilní, takže to začínající programátory nevede k nefunkčním kódům.

8.2.2. Zend framework

Zend má kolem sebe jednu z největších, ne-li největší komunitu mezi frameworky. V tom je jeho velká síla. Pokud něco není dostatečně vysvětleno v dokumentaci, je vždy možnost se zeptat na oficiálních nebo i lokálních fórech. Pro Čechy má tu nevýhodu, že je vše v angličtině a pro lidi, kteří jsou méně obratní v anglickém jazyce, je tudíž těžce dostupný.

Výhodou dokumentace Zendu je její vyčerpávající obsah. K mnoha částem frameworku existuje více programátorských přístupů a dokumentace je všechny, nebo alespoň velkou část zmiňuje a představuje. To je určitě správný přístup, ale chybí zde určení nějakého preferovaného řešení, které by bylo nejoptimálnější z nějakého hlediska (ať už by se jednalo o best practices, rychlost nebo bezpečnost). Z hlediska množství obsahu má Zend jednoznačně navrch nad Nette. V přehlednosti už to tak zřejmé není. Ačkoli mají podobné navigační rozložení, orientovalo se mi v dokumentaci Zendu o něco hůře než v Nette. Patrně to bude cena za množství informací, která jsou v dokumentaci obsažena.

Zend framework byl vždy poněkud rigidní. A i ve verzi 2 to potvrzuje a publikuje nové verze frameworku vždy podložené dokumentací, což je rozhodně velmi správný přístup.

Jméno	Zend framework	Nette framework
Přehlednost	1	2
Úplnost	2	1
Aktuálnost	2	1
Velikost komunity	2	1 (2)*
Best practices**	1	2
Celkem bodů	8/10	7(8)/10

Tabulka 8 Porovnání dokumentace [Zdroj: autor]

*Pokud budeme brát Nette framework pouze lokálně na území České republiky, tak je komunita poměrně veliká. V celosvětovém měřítku se ovšem nemůže se Zend frameworkem měřit.

** Je myšleno: Jeden preferovaný přístup, který je nejlepší z určitého pohledu a který by měl být tedy preferován.

API dokumentaci mají oba frameworky na srovnatelné úrovni, proto nebudou předmětem porovnání.

8.2.3. Závěr

V dokumentaci jsou oba frameworky velmi podobné. Kde jeden exceluje, jiný trochu ztrácí a naopak, ale ve výsledku jsou naprosto totožní.

- Nette framework: **15(16)/20**
- Zend framework: **15/20**

9. Programování za pomoci frameworku

Zend má poměrně hodně společného s Nette frameworkem. Tato část práce má za cíl poukázat na některé, zejména nejvýraznější, rozdíly mezi těmito frameworky. Jedná se o subjektivní postřehy z programování aplikace v obou frameworcích.

9.1. Instalace

9.1.1. Zend framework

Požadavky Zend frameworku jsou poměrně benevolentní. Vyžaduje k běhu alespoň PHP 5.3 a webový server s podporou `mode_rewrite`.

Zend framework lze stáhnout ze stránek společnosti Zend (framework.zend.com), kde lze nalézt poslední stabilní verzi, stejně tak jako starší a vývojové verze. Na výběr je několik verzí. Prvně je nutno vybrat mezi ZF 1 nebo 2 a poté stáhnout buď full (plná) verzi, která má zabalená 2,6 MB nebo minimal, která se liší jen tím, že neobsahuje ukázky (dema). Oproti verzi ZF1, je to značný rozdíl. Verze ZF1 má v současné době 30,9 MB.

Další možnosti získání frameworku jsou prostřednictvím verzovacího systému, konkrétně **GitHub**u a balíčkovacího systému **Compouser**u.

Ve verzi ZF1 je oproti ZF2 ještě komponenta Zend_Tool. Tato komponenta se již v ZF2 nevyskytuje.

9.1.2. Nette framework

U Nette frameworku záleží na použité verzi. Nabízí totiž dvě verze, jednu se jmennými prostory a druhou bez nich. Verze bez jmenných prostorů vyžaduje PHP 5.2.0 (tedy postačí i starší verze na rozdíl od Zend frameworku) a několik požadavků na prostředí webového serveru. Pro otestování serveru nabízí i takzvaný `Requirements Checker`, který automaticky otestuje webové prostředí serveru a zvýrazní nastavení, která je potřeba nebo doporučeno změnit. Verze se jmennými prostory potom samozřejmě vyžaduje PHP 5.3.0 (ve starších verzích není podpora pro jmenné prostory).

Nette framework je možné stáhnout ze stránek Nette foundation (nette.org), kde jsou stejně jako u Zend frameworku k dispozici starší i vývojové verze. Po stažení má Nette framework 9,5 MB. Z tohoto pohledu se Nette vydalo na opačnou cestu než Zend. Zend balíček zmenšil, zatímco Nette zvětšil z 3,66 MB na současných 9,5 MB. Z pohledu velikosti je na tom Zend lépe než Nette.

Stejně jako Zend i Nette má stejné další distribuční kanály GitHub a Compouser. Na GitHubu je Nette ke stažení zpravidla v nejnovější (i developerské) verzi. Nette z tohoto

zdroje neobsahuje některé části jako verze stažená ze stránek Nette.org. Zejména chybí sandbox a Requirements Checker.

9.2. Controller/Presenter

Controller, respektive presenter je srdce aplikace. Aplikace by čistě teoreticky mohla fungovat i jen s tímto prvkem. Do databáze by se dotazoval sám, místo toho aby se ptal modelu a data by potom sám vykreslil a poslal na výstup. Tento postup je však silně nedoporučován.

9.2.1. Zend framework

Celá komunikace uživatele se systémem je zahájena tím, že zavolá **index.php**. Ostatně celá komunikace prochází přes **index.php**. Nastaví se zde relativní cesta do kořenového adresáře aplikace, zavolá se **autoloader** a spustí se hlavní konfigurační soubor aplikace.

Nette framework (viz. níže) podporuje poměrně velké množství metod pro práci během zpracování presenteru. Zend framework naproti tomu jde mnohem jednodušší cestou, kdy má v podstatě jen 1 metodu, která je velmi často používána.

Zde bych rád upozornil na poměrně důležitou změnu, kdy z frameworku byla odebrána metoda `init()`, která sloužila místo konstruktoru, který bylo doporučováno nepřepisovat. Nyní je naopak doporučováno přepsat, respektive zdědit konstruktor od předka.

indexAction()

Metoda, která slouží k vypsání určitého pohledu aplikace. Tato metoda vlastně kompletně obsluhuje daný view, kdy komunikuje s databází, zajišťuje přihlašování a ostatní aplikační logiku, která se týká view. Konkrétně u metody `indexAction()` to je view **index.phtml**.

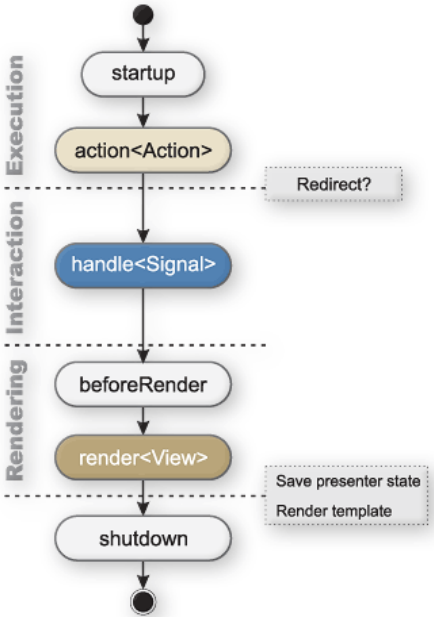
9.2.2. Nette framework

U Nette frameworku to funguje trochu jinak. Všechny požadavky procházejí přes **index.php**, ve lterém se definují konstanty a zavolá se **bootstrap.php**. V bootstrap.php se nakonfiguruje a spustí $application. V bootstrapu setaké definují routy a vůbec celé nastavení aplikace se odehrává zde a v **config.neon**.

Rozdíl je v používání presenteru a layoutu. Nette používá tzv. **BasePresenter**. Tento presenter je předkem pro všechny ostatní presentery. Nette nemá žádnou komponentu jako je

Zend_Navigation, takže BasePresenter je ideální místo pro tahání dat z databáze pro dynamické menu a další dynamické prvky, které mají být na celé webové stránce.

Uvnitř presenterů je poměrně velké množství předdefinovaných metod, na které je rozdělen jeho životní cyklus. Respektive tyto metody jsou uvnitř třídy **Nette\Application\UI\Presenter**, ze které dědí BasePresenter a z něj dědí všechny ostatní presentery. (Nette foundantion, 2012d)



Obrázek 6 Životní cyklus presenteru [Zdorj: (Nette foundantion, 2012d)]

Startup()

Tato metoda se volá ihned po vytvoření presenteru. Slouží k inicializaci proměnných, kontrole přihlášení a všech akcí, které jsou potřeba provést předtím, než se začne provádět ostatní kód presenteru.

action<Action>()

Metoda actionShow se používá ve chvíli, kdy není žádoucí v tuto chvíli něco vykreslit. Typicky může přihlásit uživatele nebo zapsat do databáze a poté může přesměrovat uživatele jinam. Provádí se před renderováním view, takže matoda actionShow může například

zkontrolovat, jestli je užitavel přihlášen, pokud ano tak vypsat data z databáze, pokud ne tak ho přesměrovat na jinou stránku.

Syntaxe je stejná jako v Zend frameworku, kdy actionShow například v ProductPresenteru se zavolá příchodem na stránku `http://mujweb.cz/product/show`.

handle<Signal>()

Handle je metoda, která zpracovává zejména signály (subrequesty). Tyto jsou určeny hlavně pro zpracování AJAXových požadavků. Je to tzv. komunikace pod prahem normálního view.

beforRender()

„Metoda beforeRender, jak už název napovídá, se volá před metodou **render<View>()** a může obsahovat například nastavení šablony, předání proměnných společných pro více view a podobně.“

render<Show>()

Slouží k předání dat view a volá se vždy, pokud se má vykreslit dané view. Má stejnou logiku názvu jako action, tedy aby se vykreslila šablona **show.latte** musí být uživatel na stránce `http://mujweb.cz/product/show`.

shutdown()

Volá se při ukončení životního cyklu presenteru. Presenter může být ukončen více způsoby. Pro přímé ukončení slouží `$presenter->terminate()`, ale existují i jiné metody, které mají vždy trochu jiné využití.

9.3. View

Tato komponenta slouží ke generování výstupů z aplikace. Má na starosti, jak výstup v podobě HTML, tak i dalších alternativ jako je RSS, PDF, tiskárna, atd.

9.3.1. Zend framework

Zend framework nepoužívá žádné šablonovací systémy (template systems). Používá html soubory s koncovkou `*.phtml`, bez nějaké přidané či modifikované syntaxe.

O view se stará komponenta **Zend\View**. Pomocí této komponenty může controller předávat proměnné do view. Controller dědí z **AbstractActionController**u, takže stačí napsat jen: `return array('message' => 'Hello world');`. Další možností je použít **ViewModel**. Celé použití vypadá následovně.

```
$view = new Zend\View\Model\ViewModel;ViewModel(array(
     'message' => 'Hello world',
));
$view->setTemplate('cesta/k/view');
return $view;
```

`message` je proměnná, která se potom může použít uvnitř view. Tuto proměnnou lze velice snadno zobrazit v samotném pohledu příkazem `$this->message`. Z toho je vidět, že se opravdu jedná o jednu instanci třídy, kdy se v controlleru nadefinují atributy a uvnitř view se k nim může přistupovat.

Komponenta dovoluje i volání jiné akce než ve které je právě vykonáván požadavek. Například pokud je uživatel na stránce `http://webpage.com/` znamená to, že se vykoná indexAction() v IndexControlleru. Je to proto, že stránka je vlastně zkrácenou formou `http://webpage.com/index/index` a zpracování této stránky náleží právě index controlleru a index metodě. Pokud je to však žádoucí může být z této indexAction() zavolána jiná šablona, která se poté postará o vykreslení. To lze provést právě pomocí `$view->setTemplate('cesta/k/view');`

Odkazy se nepíší v normálním tvaru, ale vždy se určí, jaká routa je bude zpracovávat. Příklad takového příkazu je: `<a href="<?php echo $this->url('eshop', array('action'=>'product', 'id' => $product->id));?>"><?= $product->title; ?></a>` Kde `$this->url` je helper pro URL (Uniform Resource Locator) adresy, `'eshop'` je název routy, která daný požadavek má zpracovávat a poté pole jednotlivých parametrů, zde předány parametry action a id.

9.3.2. Nette framework

Nette framework používá vlastní šablonovací systém, který byl vyvinut Nette foundation s názvem Latte. Myslím si, že to je velká přidaná hodnota oproti Zend frameworku. Vývoj za použití frameworku se tím stává o hodně rychlejší a přehlednější. Po nainstalování Nette

pluginu do IDE (Vývojové prostředí – Integrated Development Environment) prostředí se syntaxe latte zvýrazňuje.

Tato výhoda nepřekáží ani ve chvíli, kdy chce programátor použít některý jiný šablonovací systém nebo Latte syntaxi prostě vypnout. Pro vypnutí šablonovacího systému existuje zápis ve tvaru `syntax off`, který vypne makra na kterých je Latte založeno.

Latte používá dva způsoby zadávání tzv. **maker** (speciální výrazy). Buďto se může jednat o makra uvozená a uzavřená ve složených závorkách (např. `{$variable}`) nebo o takzvaná **n:makra**. Závorková makra v podstatě nahrazují uvození a ukončení párovým tagem spolu s escapováním `<?php echo ?>`. N:makra jsou vkládána přímo dovnitř html elementu jako jeho atribut a mají vliv na element, ve kterém jsou vloženy plus jeho podelementy. N:makra se zapisují jen přidáním „n:“ před běžný atribut nebo php výraz (např. `<li n:if="$product->published">Obsah buňky</li>`). Tento krátký zápis vypíše něco jako:

```
<?php if($product->published): ?>
<li>Obsah buňky</li>
<?php endif ?>
```

Jak je vidět z příkladů výše Latte přistupuje k proměnným přímo, jakoby byli inicializovány lokálně přímo uvnitř view. Pro práci se šablonou to je přehlednější a rychlejší než práce v Zend frameworku bez šablon. Volání stejné proměnné v Zend a Nette frameworku: `<?php echo escape($this->variable) ?>` oproti `{$variable}`. Latte totiž automaticky volá escape() na proměnné uvnitř maker. Příjemné pro práci je, že tento escape se liší v závislosti na tom, kde je volán. Použije se jiná escape syntaxe uvnitř javascriptu a html.

Jednodušší je i tvorba odkazů. S pomocí Latte jsou odkazy kratší než u Zend frameworku. Latte využívá dvojtečku k oddělení presenteru a akce. Příklad takového odkazu může vypadat například takto: `<a n:href=":Eshop:Product:default $product->id">{$product->title}</a>`. První dvojtečka říká, že se odkazuje absolutně. `Eshop` je název modulu, ve kterém se má hledat (pokud aplikace nepoužívá moduly, může se vynechat jak úvodní dvojtečka, tak název modulu). `Product` je název presenteru ve kterém se zavolá akce `default`. Této akci se předá parametr (proměnná) `$product->id`, která je oddělena od akce mezerou.

Předání proměnné šabloně je velmi podobné jako u Zend frameworku. Předává se v presenteru příkazem: `$this->template->variable`, kde `variable` je proměnná. Vypsání uvnitř view už bylo popsáno výše, je to (s použitím Latte) jen proměnná ve složených závorkách (`{$variable}`). Přesměrování na jiné view je také velmi podobné a sice `$this->redirect('Presenter:action);`. Rozdíl je v tom, že v Zend frameworku se volá view, které se má vykreslit a v návaznosti na to se použije controller. V Nette se volá presenter, který má inicializovat vykreslení a v závislosti na to se volá view.

9.4. Komponenty

Komponenty jsou jednotlivé části aplikace, které řeší ucelený problém. Například Zend\Form je komponenta pro generování formulářů, Zend\Paginator generuje stránkování, atd. Oba frameworky využívají množství komponent. Zde bude z důvodu rozsahu práce uvedena a popsána jen komponenta formuláře.

9.4.1. Formuláře

Formuláře jsou jedním z nejčastějších prvků na stránce. Jedná-li se o dynamický web, jde prakticky o nutnost použít na stránce nějaké formulářové prvky.

Zend framework

Formulář podporuje HTML5 validaci, tudíž lze k jednotlivým elementům navěsit i HTML5 atributy. Zend framework má pro formuláře vlastní třídu Zend\Form. Tato třída je hlavním rozhraním při práci s formulářem. Framework dále disponuje množstvím tříd pro jednotlivé elementy formuláře a mnoho tříd pro úpravu vzhledu. Základní použití Zend\Form spočívá ve vytvoření instance Zend\Form a nastavení atributů elementu form. Poté je třeba vytvořit instanci pro každý objekt, který se bude vykreslovat. Ten se vytvoří buď standardně `new Element\Email('jmenoElementu');` nebo metodou Zend\Form\Form: `$this->add(array('name' => 'name'));`. Na vytvořený objekt elementu je možné navěsit další informace jako je popisek (label), validace, či jestli má být vyplnění elementu vyžadováno. Poté se všechny takto vytvořené objekty postupně předají třídě Zend\Form a její metodě `add();`. Nakonec je třeba předat view třídu Zend\Form k vykreslení. Základní vykreslení je prosté zavolání `echo $this->form;`.

```
$name = new Element('name');
$name->setLabel('Vaše jméno');
```

```
$name->setAttributes(array('type'  => 'text'));

$email = new Element\Email('email');
$email->setLabel('Váš e-mail');

$send = new Element('send');
$send->setValue('Odeslat');
$send->setAttributes(array('type'  => 'submit'));

$form = new Form();
$form->add($name);
$form->add($email);
$form->add($send);
return array('form' => $form);
```

Po odeslání je třeba ověřit, zda se formulář odeslal a zda je validní. Některé skutečnosti je třeba ověřovat až po odeslání formuláře (pokud není použit AJAX). Například jestli zvolené jméno v databázi již existuje. Zend formuláře dovolují i po úspěšném odeslání dodatečně vypsat chybu. Pokud je vše validní, tak po zpracování je třeba přesměrovat uživatele (třeba na tu samou stránku) aby se vyprázdnili hodnoty a hlavně aby při znovunačtení stránky nedošlo k opakovanému odeslání formuláře.

Nette framework

Formuláře jsou velkou zbraní Nette frameworku. Jejich použití je poměrně intuitivní a mají i pěkně vyřešené zpracování. Stejně jako Zend i Nette podporuje HTML5 validaci. Ve spolupráci s netteForms, které je napsané v jQuery (JavaScriptová knihovna) formuláře validují data už na straně klienta přesně podle validačních pravidel, která se nastaví v presenteru. Pokud tato knihovna není přístupná nebo má uživatel vypnutý JavaScript, tak samozřejmě funguje i PHP validace.

Pro každý formulář, který se vkládá do stránky, se musí vytvořit objekt **Nette\Forms\Form()**. Pomocí fluent interface (plynulé rozhraní) se poté vytvoří prvky formuláře i s dalším množství nastavení, které lze pomocí metod nastavit.

Celé vykreslení formuláře vypadá následovně:

```
protected function createComponentNewUserForm(){
    $form = new Nette\Application\UI\Form;
```

```
        $form->addText('name', 'Jméno:');
        $form->addPassword('password', 'Heslo:');
            ->setRequired('Zvolte si heslo')
            ->addRule(Form::MIN_LENGTH, 'Heslo musí mít alespoň %d znaky', 6);
        $form->addPassword('passwordVerify', 'Heslo pro kontrolu:')
            ->setRequired('Zadejte prosím heslo ještě jednou pro kontrolu')
            ->addRule(Form::EQUAL, 'Hesla se neshodují', $form['password']);
        $form->addSubmit('login', 'Přihlásit se');
        $form->onSuccess[] = callback($this, 'newUserFormSubmitted');
        return $form;
    }
```

Takto napsaný kód je v podstatě samo vysvětlující. Formulář je komponenta a proto se vytváří zavoláním metody `createComponent<JménoKomponenty>()`, která vytvoří komponentu se zadaným názvem. Poté se musí vytvořit instance třídy Form. Pomocí metod třídy Form se poté přidávají jednotlivé položky formuláře, které se mohou zřetězit s pravidly. Například u položky passwordVerify je vyžadováno neprázdné pole a kontroluje se, jestli se heslo shoduje s položkou password. Po kliknutí na tlačítko submit se zavolá atribut `onSuccess[]`, kterému se předá callback v aktuálním presenteru a metodou `newUserFormSubmitted`.

V šabloně se poté volá formulář, ačkoli se jedná o komponentu, trochu jinak. Buď se může využít defaultního vykreslení, které vykreslí formulář v předpřipraveném rozložení. To se provede, stejně jako u Zendu, prostě vypsáním formuláře `echo $form;`. Druhou možností je vykreslit formulář ručně. To může vypadat například jako následující příklad. Zavolá se makro form, kterému se předá název komponenty. Struktura je normální HTML a místo elementu label a input se použije jejich makro ekvivalent, kterému se předá název, který se vygeneroval v presenteru při vytváření jednotlivých prvků formuláře. Volitelně se můžou předat další parametry, jako jsou defaultní hodnota, velikost pole a pokud je formulář napsaný v HTML5 tak i jeho atributy jako je autofocus.

```
{form newUserForm}
        {label name /}{input name autofocus => TRUE}
        {label password /}{input password}
    {label passwordVerify /}{input passwordVerify} {input login}
{/form}
```

Zpracování formuláře se provádí v metodě, která je předána callbacku (zpětné volání) a může se tedy jmenovat naprosto libovolně. V této metodě už není potřeba ověřovat, jestli se vše odeslalo v pořádku. Pokud nejsou dodrženy podmínky, které jsou navěšeny na jednotlivých prvcích formuláře, tak se formulář neodešle a vypíše uživateli, co má udělat, pro úspěšné odeslání. Stejně jako Zend framework i Nette má možnost po úspěšném odeslání vypsat chybu, jestliže se zjistí skutečnosti, které nebyly ošetřeny u jednotlivých prvků. Pakliže je vše v pořádku provede se zpracování formuláře a na závěr se přesměruje stránka. Kdyby se nepřesměrovala stránka jinam (nebo na sebe samou), tak by zůstaly pole ve formuláři vyplněny tak, jak byly při posledním odeslání a při znovunačtení stránky by prohlížeč odeslal data znovu. Opět stejné chování jako Zend.

9.5. Moduly

Obě aplikace jsou modulární. Zde je poměrně velký rozdíl v tom, že Zend je přísně modulární a i samotná nemodulární aplikace je vlastně modul. Každý modul má vlastní nastavení je tedy možné aby si každý z modulů žil vlastním životem.

Naproti tomu Nette má modulární systém jen jako dobrovolný a není nutné aplikaci do modulů členit. Základní využití modulů může a nemusí sdílet nastavení pro celou aplikaci, v základu však sdílí. Zde nejsem schopen rozhodnout, který z přístupů je lepší. V tomto bodě je Zend, podle mého názoru, lépe připraven na opravdu rozsáhlé modulární aplikace o něco lépe než Nette.

9.6. Závěr

Tato kapitola byla více méně o pocitu z používání. Byly zde jen v krátkosti představeny některé prvky a rozdíly mezi oběma frameworky. Jak jsem psal už výše, Nette framework je na používání ze začátku jednodušší. Je to dáno i tím, že v Zendu se musí vše někde explicitně napsat, zatímco Nette má „inteligentní“ rozpoznávání a procházení adresářů. Například vytvořený presenter se nikam nemusí registrovat a zaregistruje se sám, to samé s moduly. Musí mít sice speciální tvar, ale funguje to naprosto automaticky a to hodně usnadní práci. Zejména to eliminuje chyby ze zapomnětlivosti, kterých jsem se v Zendu dopouštěl poměrně často (deklarovat controller, předat model, …)

Ke konci programování v Zendu jsem již alespoň částečně pronikl do frameworku a nezdál se mi tak špatný jako na začátku. To jen podtrhuje, že křivka učení je opravdu

exponenciální. Bohužel, alespoň pro mě, začínala výrazně níž než křivka Nette. Jelikož se jedná o porovnání jen dvou frameworků, Nette získá 100% bodů z použitelnosti a Zend relativní odstup od Nette tak jak ho vnímám já, tzn. 60%.

- Nette framework: **5 b**
- Zend framework: **3 b**

10. Závěrečné shrnutí

Každá kapitola má svůj závěr, ke kterému jsem dospěl, proto zde je jen celkové shrnutí obou frameworků. Oba frameworky si jsou velmi podobné až na jeden rozdíl. Ten je ovšem na tolik markantní, že dělá celkový rozdíl téměř 20% mezi oběma frameworky. Tímto rozdílem je rychlost. Nette je téměř o polovinu rychlejší. V tak malé aplikaci, jaká je součástí práce, to nevynikne tolik, ale v rozsáhlé a složité aplikaci bude rozdíl o hodně výraznější.

Další věcí, kterou bych rád podotknul, je učící se křivka. Křivka, která udává v čase, jak rychle je programátor schopen se nový framework naučit. Zend uvádí, že má progresivní učící se křivku, s čímž mohu naprosto souhlasit. Věci mi šly čím dál rychleji se učit. Nette na svých stránkách zase proklamuje, že má strmou učící se křivku. I toto tvrzení mohu potvrdit. Subjektivně se mi zdá, že křivka Nette začíná výš než křivka Zendu, proto se obě křivky protnou až za velmi dlouhou dobu (pokud vůbec).

Oblast	Zend framework	Nette framework
Bezpečnost	39	43
Rychlost	18	32
Dokumentace	15	15(16)
Používání	3	5
Celkem	**75**	**95(96)**

Tabulka 9 Závěrečné hodnocení [Zdroj: autor]

Výsledné hodnocení je pouze relativní, nikoli absolutní. Nelze usuzovat, že Nette framework je téměř dokonalý - to jistě není. Prostor pro zlepšení tu je vždy. Podařilo se mi však prokázat, že pro aplikace malého až středního rozsahu je Nette lepším frameworkem než Zend. Je o necelých 44% rychlejší, bezpečnější a lépe se učí. Naproti tomu se Zend jeví jako velmi stabilní a je možná o něco více připravený na rozsáhlé projekty než Nette. Na otázku, který framework zvolit, bych ve zvolených verzích odpověděl jednoznačně: Nette.

Bibliografie

Bestframeworks.com, 2011. *Compare PHP Frameworks »Zend Framework« vs. »Nette«.* [Online]
Available at: http://www.bestwebframeworks.com/compare-web-frameworks/php/1-zend-framework-vs-120-nette/
[Přístup získán 2 12 2012].

Böhmer, M., 2010. *Zend framework Programujeme webové aplikace v PHP.* Brno: Computer Press.

Codex-m, 2012. *Most Used PHP Framework-The Popular Top 7 List in year 2011.* [Online]
Available at: http://www.php-developer.org/most-used-php-framework-the-popular-top-7-list-in-year-2011/
[Přístup získán 20 8 2012].

Daněk, P., 2008. *Velký test PHP frameworků.* [Online]
Available at: http://www.root.cz/clanky/velky-test-php-frameworku-2008/
[Přístup získán 2012].

Knesl, J., 2012. *Webexpo 2012: Souboj frameworků.* [Online]
Available at: http://www.knesl.com/articles/view/webexpo-2012-souboj-frameworku
[Přístup získán 20 11 2012].

Nette foundantion, 2012a. *Licenční politika.* [Online]
Available at: http://nette.org/cs/license
[Přístup získán 29 11 2012].

Nette foundantion, 2012c. *Databáze & ORM.* [Online]
Available at: http://doc.nette.org/cs/database
[Přístup získán 4 12 2012].

Nette foundantion, 2012d. *MVC aplikace & presentery.* [Online]
Available at: http://doc.nette.org/cs/presenters
[Přístup získán 1 10 2012].

Nette Foundation, 2012b. *Download.* [Online]
Available at: http://nette.org/cs/download
[Přístup získán 28 12 2012].

Olišar, P., 2012b. *Nette framework - Kučera.* [Online]
Available at: https://github.com/Olicek/nette
[Přístup získán 17 12 2012].

Olišar, P., 2012. *Nette framework - Kučera.* [Online]
Available at: https://github.com/Olicek/nette
[Přístup získán 17 12 2012].

PHPFrameworks.com, 2012. *PHP Frameworks.* [Online]
Available at: http://www.phpframeworks.com/
[Accessed 1 12 2012].

Stack exchange inc, nedatováno *Stack overflow.* [Online]
Available at: http://stackoverflow.com/questions/12232874/php-am-i-mixing-up-event-driven-programming-with-signals-aware-interfaces-sing
[Přístup získán 2 12 2012].

Vrána, J., 2010. *notORM.* [Online]
Available at: http://www.notorm.com/
[Přístup získán 2 12 2012].

Vrána, J., 2012. *Souboj frameworků nevyhrál framework.* [Online]
Available at: http://php.vrana.cz/souboj-frameworku-nevyhral-framework.php
[Přístup získán 1 12 2012].

Website Optimization, 2008. *The Psychology of Web Performance.* [Online]
Available at: http://www.websiteoptimization.com/speed/tweak/psychology-web-performance/
[Přístup získán 1 10 2012].

Wikipedia foundantion, 2012a. *Framework.* [Online]
Available at: http://cs.wikipedia.org/wiki/Framework
[Přístup získán 15 11 2012].

Wikipedia foundantion, 2012b. *BSD licence.* [Online]
Available at: http://cs.wikipedia.org/wiki/BSD_licence
[Přístup získán 20 11 2012].

Wikipedia foundantion, 2012c. *Zend Framework.* [Online]
Available at: http://en.wikipedia.org/wiki/Zend_Framework
[Accessed 28 9 2012].

Wikipedia foundantion, 2012d. *Copyleft.* [Online]
Available at: http://cs.wikipedia.org/wiki/Copyleft
[Přístup získán 19 11 2012].

Wikipedia foundantion, nedatováno *Web template system.* [Online]
Available at: http://en.wikipedia.org/wiki/Web_template_system
[Přístup získán 2 12 2012].

Wikipedie foundantion, 2012e. *Objektově relační mapování.* [Online]
Available at:
http://cs.wikipedia.org/wiki/Objektov%C4%9B_rela%C4%8Dn%C3%AD_mapov%C3%A1n%C3%AD
[Přístup získán 2 12 2012].

Zdoják.cz, 2009. *Prezentační vzory z rodiny MVC.* [Online]
Available at: http://www.zdrojak.cz/clanky/prezentacni-vzory-zrodiny-mvc/
[Přístup získán 19 10 2012].

Zend Technologies Ltd., 2010. *Zend - The PHP Company.* [Online]
Available at: http://www.zend.com/en/company/
[Accessed 29 10 2012].

Zend Technologies Ltd., 2012a. *About.* [Online]
Available at: http://framework.zend.com/about/
[Accessed 12 12 2012].

Zend Technologies Ltd., 2012b. *ZF Blog*. [Online]
Available at: http://framework.zend.com/blog//zend-framework-2-0-5-released.html
[Přístup získán 30 11 2012].

Přílohy

Následující přílohy obsahují podrobné výsledky měření rychlosti Nette a Zend frameworku.

Příloha 1 – Podrobná data z měření Homepage

Příloha 2 – Výsledky měření Homepage

Příloha 3 – Podrobná data z měření SELECT

Příloha 4 – Výsledky měření SELECT

Příloha 5 – Podrobná data z měření EDIT – 1. Měření

Příloha 6 – Podrobná data z měření EDIT – 2. Měření

Příloha 7 – Podrobná data z měření EDIT – 3. Měření

Příloha 8 – Výsledky měření EDTI

Příloha 9 – Výsledná tabulka Nette a Zend

Příloha 10 – Výsledná tabulka

Příloha 11 – Aplikace naprogramovaná v Nette (Olišar, 2012)

Příloha 12 – Aplikace naprogramovaná v Zend (Olišar, 2012b)

Příloha 1

Podrobná data z měření Homepage

	1. měření		2. měření		3. měření	
číslo měření	Nette	Zend	Nette	Zend	Nette	Zend
1	140	188	151	209	141	205
2	151	209	135	216	151	224
3	148	201	141	202	152	217
4	144	226	149	195	147	208
5	145	201	138	206	151	225
6	145	224	150	212	145	206
7	154	204	137	226	149	215
8	157	230	135	188	153	208
9	159	203	163	214	155	202
10	144	237	155	243	148	239
11	159	202	145	202	141	210
12	151	204	159	209	151	233
13	166	201	145	198	159	215
14	150	235	152	210	142	207
15	169	203	133	193	153	212
16	156	236	141	227	151	199
17	153	221	136	204	145	201
18	187	213	149	226	144	222
19	145	216	143	203	160	213
20	149	211	141	233	149	213
21	152	227	136	215	148	218
22	155	212	142	197	155	218
23	142	207	142	209	156	217
24	151	202	145	228	148	201
25	149	215	150	195	152	201
26	160	217	169	197	154	208
27	148	230	132	190	151	215
28	148	207	154	205	149	201
29	171	192	135	210	151	217
30	141	257	143	203	152	235
průměr	152,9667	214,3667	144,8667	208,8333	150,1	213,5
maximum	187	257	169	243	160	239
minimum	140	188	132	188	141	199
rozdíl		40,14%		44,16%		42,24%
medián	151	211,5	143	207,5	151	213
modus	151	201	135	209	151	201
sm.odchylka	9,954842	15,07865	8,954453	13,15823	4,671545	10,22334

Příloha 2

Výsledná tabulka Nette

	vzorky	průměr [ms]	medián[ms]	Min[ms]	Max[ms]	Sm. Odchylka [ms]	Modus[ms]
I. měření	30	152,97	151	140	187	9,95	151
II. Měření	30	144,87	143	132	169	8,95	135
III měření	30	150,1	151	141	160	4,67	151
CELKEM	**90**	**149,31**	**148,33**	**137,67**	**172**	**7,86**	**145,67**

Výsledná tabulka Zend

	vzorky	průměr [ms]	medián[ms]	Min[ms]	Max[ms]	Sm. Odchylka [ms]	Modus[ms]
I. měření	30	214,37	211,5	188	257	15,08	201
II. Měření	30	208,83	207,5	188	243	13,16	209
III měření	30	213,5	213	199	239	10,22	201
CELKEM	**90**	**212,23**	**210,67**	**191,67**	**246,33**	**12,82**	**203,67**

Výsledná tabulka Homepage

	vzorky	průměr [ms]	medián[ms]	Min[ms]	Max[ms]	Sm. Odchylka [ms]	Modus[ms]
Nette	270	149,31	148,33	137,67	172	7,86	145,67
Zend	270	212,23	210,67	191,67	246,33	12,82	203,67
Procentuální rozdíl	**270**	**42,14%**	**42,02%**	**39,23%**	**43,22%**	**63,10%**	**39,82%**

Příloha 3

Podrobná data z měření SELECT

	1. měření		2. měření		3. měření	
číslo měření	Nette	Zend	Nette	Zend	Nette	Zend
1	171	227	199	204	192	210
2	165	220	193	225	185	246
3	169	217	201	215	195	204
4	175	216	197	190	190	206
5	180	217	190	213	225	213
6	169	217	185	227	191	230
7	170	238	189	199	201	234
8	174	259	190	235	181	211
9	171	213	193	236	202	206
10	173	212	183	214	179	222
11	157	229	187	224	169	232
12	166	207	189	216	185	206
13	171	223	222	214	189	215
14	153	257	198	201	178	205
15	177	253	199	224	181	223
16	175	201	194	205	175	211
17	171	210	189	188	178	225
18	173	218	188	216	173	221
19	179	220	189	238	181	216
20	165	215	190	201	175	215
21	179	213	195	213	175	223
22	174	220	190	198	178	213
23	176	237	190	220	199	207
24	169	243	202	225	176	207
25	160	208	196	205	183	204
26	181	234	186	215	178	212
27	167	213	191	220	176	217
28	171	222	184	214	191	229
29	165	210	195	212	174	231
30	165	216	186	218	180	204
průměr	170,3667	222,8333	192,6667	214,1667	184,5	216,6
maximum	181	259	222	238	225	246
minimum	153	201	183	188	169	204
rozdíl		0,307963		0,111592		0,173984
medián	171	217,5	190	214,5	181	214
modus	171	220	190	214	178	204
sm.odchylka	6,467783	14,59699	7,372622	12,31282	11,38347	10,70389

Výsledná tabulka Nette

	vzorky	průměr [ms]	medián[ms]	Min[ms]	Max[ms]	Sm. Odchylka [ms]	Modus[ms]
I. měření	30	170,37	171	153	181	6,47	171
II. Měření	30	192,67	190	183	222	7,37	190
III měření	30	184,5	181	169	225	11,38	178
CELKEM	**90**	**182,51**	**180,67**	**168,33**	**209,33**	**8,41**	**179,67**

Výsledná tabulka Zend

	vzorky	průměr [ms]	medián[ms]	Min[ms]	Max[ms]	Sm. Odchylka [ms]	Modus[ms]
I. měření	30	222,83	217,5	201	259	14,60	220
II. Měření	30	214,17	214,5	188	238	12,31	214
III měření	30	216,6	214	204	246	10,70	204
CELKEM	**90**	**217,87**	**215,33**	**197,67**	**247,67**	**12,54**	**212,67**

Výsledná tabulka SELECT

	vzorky	průměr [ms]	medián[ms]	Min[ms]	Max[ms]	Sm. Odchylka [ms]	Modus[ms]
Nette	270	182,51	180,67	168,33	209,33	8,41	179,67
Zend	270	217,87	215,33	197,67	247,67	12,54	212,67
Procentuální rozdíl	**270**	**19,37%**	**19,19%**	**17,43%**	**18,31%**	**49,12%**	**18,37%**

Příloha 5

Podrobná data z měření EDIT – 1. měření

1. měření

	Nette			Zend		
	POST	GET	suma	POST	GET	suma
1	134	155	289	246	230	476
2	143	152	295	247	233	480
3	144	161	305	235	238	473
4	137	167	304	270	240	510
5	135	157	292	249	237	486
6	133	162	295	256	234	490
7	144	161	305	270	233	503
8	136	163	299	271	234	505
9	138	161	299	252	221	473
10	135	163	298	260	235	495
11	127	160	287	257	233	490
12	148	151	299	252	229	481
13	141	161	302	249	235	484
14	159	159	318	268	226	494
15	149	160	309	258	238	496
16	142	164	306	246	237	483
17	140	152	292	252	233	485
18	144	159	303	258	244	502
19	133	152	285	259	224	483
20	143	160	303	273	268	541
21	132	161	293	268	236	504
22	132	151	283	257	232	489
23	138	160	298	267	245	512
24	130	161	291	245	236	481
25	166	163	329	267	249	516
26	138	161	299	242	237	479
27	146	163	309	251	233	484
28	135	153	288	268	237	505
29	139	163	302	257	238	495
30	139	160	299	252	233	485
průměr	140	159,2	299,2	256,7333	235,9333	492,6667
maximum	166	167	329	273	268	541
minimum	127	151	283	235	221	473
rozdíl				83,38%	48,20%	64,66%
medián	138,5	160,5	299	257	235	489,5
modus	144	161	299	252	233	473
sm.odchylka	8,029114	4,245782	9,495261976	9,705439	8,172855	14,60898

Příloha 6

Podrobná data z měření EDIT – 2. měření

2. měření

	Nette			Zend		
	POST	GET	suma	POST	GET	suma
1	145	165	310	269	234	503
2	146	168	314	253	233	486
3	145	166	311	236	223	459
4	145	167	312	254	230	484
5	132	161	293	248	235	483
6	135	168	303	242	229	471
7	141	164	305	235	230	465
8	145	166	311	247	231	478
9	152	156	308	245	234	479
10	145	168	313	252	222	474
11	144	163	307	246	230	476
12	139	164	303	241	227	468
13	150	168	318	266	233	499
14	139	154	293	250	230	480
15	137	157	294	248	227	475
16	132	157	289	254	229	483
17	142	155	297	229	232	461
18	141	156	297	235	231	466
19	136	165	301	253	230	483
20	141	157	298	268	230	498
21	137	169	306	233	229	462
22	149	159	308	230	226	456
23	156	157	313	237	231	468
24	135	162	297	262	234	496
25	149	167	316	232	233	465
26	134	159	293	251	232	483
27	135	158	293	235	231	466
28	149	170	319	247	229	476
29	153	167	320	236	234	470
30	142	158	300	238	233	471
průměr	142,3667	162,3667	304,7333333	245,7333	230,4	476,1333
maximum	156	170	320	269	235	503
minimum	132	154	289	229	222	456
rozdíl				72,61%	41,90%	56,25%
medián	142	163,5	305,5	246,5	230,5	475,5
modus	145	168	293	235	230	483
sm.odchylka	6,342888	4,909062	8,812617218	11,01494	3,072458	11,97423

Příloha 7

Podrobná data z měření EDIT – 3. měření

3. měření

	Nette			Zend		
	POST	GET	suma	POST	GET	suma
1	149	168	317	251	229	480
2	146	167	313	235	232	467
3	138	169	307	243	227	470
4	146	168	314	246	235	481
5	141	172	313	238	229	467
6	135	168	303	243	232	475
7	138	163	301	239	239	478
8	129	156	285	239	230	469
9	146	166	312	253	223	476
10	149	166	315	257	227	484
11	148	170	318	231	229	460
12	135	159	294	231	232	463
13	145	161	306	233	228	461
14	137	169	306	230	231	461
15	143	157	300	260	230	490
16	145	159	304	245	228	473
17	138	157	295	228	230	458
18	144	165	309	268	236	504
19	139	154	293	254	231	485
20	139	163	302	241	231	472
21	143	161	304	267	224	491
22	143	151	294	246	241	487
23	140	157	297	267	230	497
24	138	161	299	252	236	488
25	141	167	308	241	233	474
26	138	164	302	254	237	491
27	132	151	283	256	223	479
28	149	164	313	244	227	471
29	141	152	293	260	237	497
30	131	161	292	246	235	481
průměr	140,8667	162,2	303,0666667	246,6	231,0667	477,6667
maximum	149	172	318	268	241	504
minimum	129	151	283	228	223	458
rozdíl				75,06%	42,46%	57,61%
medián	141	163	303,5	245,5	230,5	477
modus	138	161	313	246	230	467
sm.odchylka	5,277205	5,79885	9,132116707	11,19702	4,411601	11,85842

Příloha 8

Výsledná tabulka Nette

	vzorky	průměr [ms]	medián[ms]	Min[ms]	Max[ms]	Sm. Odchylka [ms]	Modus[ms]
I. měření	30	299,2	299	283	329	9,50	299
II. Měření	30	304,73	305,5	289	320	8,81	293
III měření	30	303,07	303,5	283	318	9,13	313
CELKEM	**90**	**302,33**	**302,67**	**285**	**322,33**	**9,15**	**301,67**

Výsledná tabulka Zend

	vzorky	průměr [ms]	medián[ms]	Min[ms]	Max[ms]	Sm. Odchylka [ms]	Modus[ms]
I. měření	30	492,67	489,5	473	541	14,61	473
II. Měření	30	476,13	475,5	456	503	11,97	483
III měření	30	477,67	477	458	504	11,86	467
CELKEM	**90**	**482,16**	**480,67**	**462,33**	**516**	**12,81**	**474,33**

Výsledná tabulka EDIT

	vzorky	průměr [ms]	medián[ms]	Min[ms]	Max[ms]	Sm. Odchylka [ms]	Modus[ms]
Nette	270	302,33	302,67	285	322,33	9,15	301,67
Zend	270	482,16	480,67	462,33	516	12,81	474,33
Procentuální rozdíl	**270**	**59,48%**	**58,81%**	**62,22%**	**60,08%**	**40,09%**	**57,24%**

Příloha 9

Výsledná tabulka Nette

	vzorky	průměr [ms]	medián[ms]	Min[ms]	Max[ms]	Sm. Odchylka [ms]	Modus[ms]
I. Měření							
Homepage	30	152,97	151	140	187	9,95	151
SELECT	30	170,37	171	153	181	6,47	171
EDIT	30	299,2	299	283	329	9,50	299
II. Měření							
Homepage	30	144,87	143	132	169	8,95	135
SELECT	30	192,67	190	183	222	7,37	190
EDIT	30	304,73	305,5	289	320	8,81	293
III. Měření							
Homepage	30	150,1	151	141	160	4,67	151
SELECT	30	184,5	181	169	225	11,38	178
EDIT	30	303,07	303,5	283	318	9,13	313
CELKEM	**270**	**211,39**	**210,56**	**197**	**234,56**	**8,47**	**209**

Výsledná tabulka Zend

	vzorky	průměr [ms]	medián[ms]	Min[ms]	Max[ms]	Sm. Odchylka [ms]	Modus[ms]
I. Měření							
Homepage	30	214,37	211,5	188	257	15,08	201
SELECT	30	222,83	217,5	201	259	14,60	220
EDIT	30	492,67	489,5	473	541	14,61	473
II. Měření							
Homepage	30	208,83	207,5	188	243	13,16	209
SELECT	30	214,17	214,5	188	238	12,31	214
EDIT	30	476,13	475,5	456	503	11,97	483
III. Měření							
Homepage	30	213,5	213	199	239	10,22	201
SELECT	30	216,6	214	204	246	10,70	204
EDIT	30	477,67	477	458	504	11,86	467
CELKEM	**270**	**304,09**	**302,22**	**283,89**	**336,67**	**12,72**	**296,89**

Výsledná tabulka

	vzorky	průměr [ms]	medián[ms]	Min[ms]	Max[ms]	Sm. Odchylka [ms]	Modus[ms]
Nette	270	211,39	210,56	197	234,56	8,47	209
Zend	270	304,09	302,22	283,89	336,67	12,72	296,89
Procentuální rozdíl	**270**	**43,85%**	**43,54%**	**44,11%**	**43,53%**	**50,19%**	**42,05%**