Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Studijní obor Informatika

Bakalářská práce

Frameworky pro tvorbu webových aplikací

Tomáš Badin

Vedoucí práce: Ing. Jiří Brožek

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Frameworky pro tvorbu webových aplikací" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne _____

Poděkování

Rád bych touto cestou poděkoval Ing. Jiřímu Brožkovi za metodické vedení práce, připomínky a veškerý čas, který mi věnoval.

Frameworky pro tvorbu webových aplikací

Frameworks for web development

Souhrn

Ve své bakalářské práci se zabývám problematikou tvorby webových aplikací v jazyce PHP. Stručně popisuji historický vývoj tohoto skriptovacího jazyka, dále se věnuji jeho objektovým možnostem. Hlavní část práce je věnována softwarovým frameworkům, které umožňují snadnější a efektivnější vývoj webových aplikací. V praktické části testuji rychlost vybraných frameworků a upozorňuji na jejich výhody a nevýhody, se kterými se může začínající programátor setkat.

Summary

In my bachelor thesis I focus on web application development in PHP programming language. I briefly describe historical progress of this scripting language, and then present its object capabilities. The main part is dedicated to software frameworks, which are enabling easier and more efficient way to develop web applications. In the practical part I am testing the speed of selected frameworks and pointing out their advantages and disadvantages, which novice may encounter.

Klíčová slova: web, aplikace, PHP, framework, vývoj, OOP, testování

Keywords: web, application, PHP, framework, development, OOP, testing

Obsah

1 Úvod

Webové prezentace se vyvíjejí společně s rozmachem celosvětové sítě Internet. Zprvu byly webové stránky psány ve značkovacím jazyce HTML, který umožnil zobrazení statického obsahu, elektronických dokumentů provázaných odkazy. Postupem času však pouze statické weby nedostačovaly potřebám uživatelů a začaly být nahrazovány dynamickým a interaktivním obsahem. Jedním z programovacích jazyků, který umožnil generovat výstup pro webové stránky, se stal serverový skriptovací jazyk PHP.

Od vytvoření systému evidence přístupů k webu, kterou započal Rasmus Lerdorf vývoj PHP, uběhlo již mnoho let. V současné době na webu používáme sofistikované aplikace umožňující správu obsahu a dat, prodej zboží nebo komunikaci mezi uživateli. Takové aplikace kladou vysoké požadavky na jejich vývojáře. Každý vývojář se snaží ulehčit si opakující se činnosti a ušetřit čas. Novým trendem se tak stávají frameworky, které poskytují programátorovi dostatečnou oporu při jeho práci a umožňují mu efektivněji vytvářet propracované webové aplikace.

2 Cíl práce a metodika

Cílem této bakalářské práce je seznámení se softwarovými strukturami, které je možné využít při tvorbě webových aplikací. Softwarové frameworky jsou užitečnými pomocníky při vývoji webových projektů, které především usnadňují programátorům práci a oproti použití čistého programovacího jazyka zaručují vyšší produktivitu a znovupoužitelnost kódu.

V úvodu rešeršní části práce je představen samotný programovací jazyk PHP[1], jenž slouží především pro programování dynamických internetových stránek a je, vzhledem ke své jednoduchosti a zároveň velkému množství funkcí, nejrozšířenějším skriptovacím jazykem pro web. Dále jsou zmíněny důvody použití, architektura, základní komponenty a funkční vztahy frameworků. Druhá polovina teoretické části práce je pak zaměřena na bližší popis jednotlivých PHP frameworků, představení a srovnání jejich hlavních výhod a nedostatků. Následuje praktický test vybraných frameworků, ve kterém jsou testovány s ohledem na rychlost. Pro tento test bude vytvořeno jednoduché databázové schéma naplněné smyšlenými daty, na kterém bude možné otestovat typickou komunikaci podobnou reálnému provozu. Závěr testu je věnován zhodnocení naměřených hodnot, subjektivním pocitům programátora aplikace a získaným zkušenostem funkční stránky frameworků.

[1] rekurzivní zkratka PHP = Hypertext Preprocessor, historicky Personal Home Page

3 Úvod do problematiky webových aplikací

V této kapitole popisuji základ webových aplikací, konkrétně jazyk PHP, který je pro tvorbu internetových stránek hojně rozšířen. Zmíním také principy objektově orientovaného programování a základní architekturu aplikací.

Webovou aplikací rozumíme podle organizace QWASP[2] software, komunikující s uživatelem či jiným systémem přes počítačovou síť prostřednictvím HTTP[3] protokolu.

3.1 Skriptovací jazyk PHP a jeho historický vývoj

PHP je skriptovací programovací jazyk navržený speciálně pro interakci s HTML[4]. Používá se především pro vývoj dynamických webových stránek a klient-server aplikací (programování na straně serveru). Je velmi rozšířený pro svou jednoduchost, obsáhlost hotových knihoven a snadné nasazení na web. Kombinuje v sobě vlastnosti více programovacích jazyků, můžeme ho využít k malé osobní prezentaci i velkému projektu s obsáhlou relační databází. Je samozřejmě možné v něm programovat procedurálně, ale mnohem větší potenciál tkví v objektovém přístupu ve vývoji. PHP samozřejmě není čistě objektovým jazykem, ale je velmi vhodný pro základní pochopení principů objektového programování a následné využití objektových přístupů usnadňuje a zpříjemňuje vývoj webových aplikací.

Vznik programovací jazyka jako systému datujeme do roku 1994, kdy dánský programátor Rasmus Lerdorf vytvořil jednoduchý nástroj pro sledování návštěvnosti na osobních webových stránkách (Personal Home Page) v programovacím jazyce C, který

[2] The Open Web Application Security Project, http://www.qwasp.org/
[3] HTTP = Hypertext Transfer Protocol
[4] HTML = HyperText Markup Language

posléze uvolnil i pro své známé. Těm se nápad zalíbil a především díky nim Lerdorf systém stále vyvíjel a snažil se ho funkčně přizpůsobit novým požadavkům. Tento kód velmi brzy obohatil novým programem Form Interpreter, který umožnil komunikaci s databází a tím pádem i tvorbu dynamických webových stránek.[5]

Od roku 1998 byla přístupná verze 3.0, která byla podstatně rychlejší, obohacená například o objekty nebo cookies. Tou dobou už nebyl jazyk PHP doménou malých osobních stránek uživatelů, ale především nástrojem pro velké servery zpřístupňující velké množství dat.

V roce 2000 byla uvolněna verze PHP4, která obsahovala zcela nové jádro Zend Engine, což znamenalo rapidní zvýšení výkonu, ale také mnoho nových funkcí a vlastností. I přes tyto novinky byla zachována kompatibilita s předchozí verzí PHP3.[6]

Roku 2004 byla představena nová generace jazyka, označená jako PHP5. Nová verze opět doznala vylepšení samotného jádra, nově Zend Engine 2 a přinesla mnoho zajímavých novinek. Ve starších verzích bylo s objekty nakládáno jako s primitivními typy, nově se s nimi zachází pomocí odkazu. PHP5 podporuje privátní a chráněné proměnné a metody v definici tříd, nechybí abstraktní metody a rozhraní.[7] Nově má zabudovaný mechanismus výjimek, tedy i snazší obsluhu chyb, podporu jmenných prostorů, vícenásobné dědičnosti nebo drobné změny v zápisu řetězců.

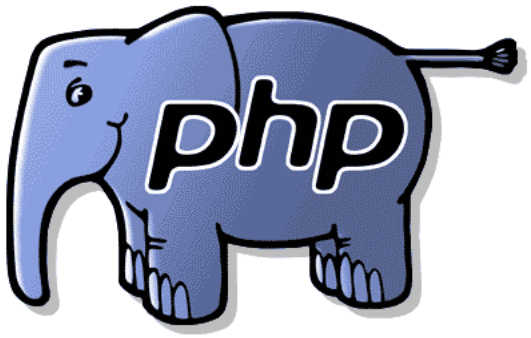
---

[5] THE PHP GROUP. *History of PHP* [online]. 2011 [cit. 2011-01-29]. Dostupné z WWW: <http://www.php.net/manual/en/history.php.php

[6] Zpracováno podle: Jiří Bráza, PHP 4 – učebnice základů jazyka, GRADA, 2002, str. 14

[7] Zpracováno podle: Oliver Leiss, Jasmin Schmidt, PHP v praxi, Grada Publishing, 2010, str. 13-14

Od PHP verze 5 se očekává, že napomůže ještě většímu rozšíření a především upevnění pozice tohoto jazyka ve vývoji webu. Tyto naděje nejsou podloženy pouze novými objektovými možnostmi, ale také snadným zacházením s XML strukturami (díky rozšíření SimpleXML), databázemi a dalšími webovými technologiemi.



Obrázek 1 – elePHPant – maskot projektu PHP

(Zdroj: http://www.cs.cas.cz/)

3.2 Objektově orientované programování v PHP

Objektové schopnosti jazyka PHP byly až do verze PHP4 poměrně omezené. Jak píše Peter Lavin[8]: „PHP4 vypadalo jako odfláknutý pokus o naskočení do rychlíku objektově orientovaného programování (dále jen OOP). Jelikož mu chyběly některé z hlavních prvků asociovaných s OOP, bylo pak jednoduché odepřít PHP jeho snahu stát se objektově orientovaným (dále jen OO) jazykem.“

Od verze PHP5 jsou možnosti jazyka mnohem větší, zarytí příznivci o něm dokonce hovoří jako o plnohodnotném OO jazyku, s čímž objektoví puristé v žádném případě nesouhlasí. To ale není vada na kráse, protože PHP si neklade za cíl být stoprocentním objektovým jazykem. Jak jsem již zmínil, jeho síla je především v jeho rozšířenosti a jednoduchosti, jazyk jde tedy, co se objektovosti týče, zlatou střední cestou. PHP můžeme nazvat hybridním jazykem, který má objektové možnosti, které však nevylučují procedurální programování. [8]

Osobně si myslím, že použité řešení, to jest objektový či procedurální princip, je závislé na rozsahu projektu a počtu programátorů. Obecně platí, že na malý projekt není OOP zapotřebí, naopak by mohlo vývoj zdržovat, zatímco při práci na větší aplikaci je vhodnější modulární přístup.

[8] Peter Lavin, PHP objektově orientované, GRADA, 2009, str. 22

3.2.1 **Objekt**

Objektem rozumíme stavební jednotku modelující část reality. Každý objekt obsahuje data a jeho vlastnosti chování. Je základem objektově orientovaného programování.

3.2.2 **Třída**

Třída je základním prvkem objektově orientovaného programování, můžeme si ji představit jako šablonu objektu. Třída objektu definuje určité vlastnosti a metody. Zvláštním druhem třídy je potom třída abstraktní. Abstraktní třída je virtuální, není možné z ní vytvořit instanci objektu. Deklaruje společné metody, které je posléze možné dědit. Použití abstraktních tříd je velmi vhodné a je základem dobrého objektového návrhu.

3.2.3 **Metoda**

Metoda je posloupnost kódu, která vykonává nějakou činnost, obecně to jsou funkce a procedury. Metody pracují s daty tříd nebo objektů a navenek jsou nepřístupné, mluvíme o zapouzdřenosti metod. V PHP často používaným typem metody je metoda statická (značíme ji *static*), kterou je možné použít, aniž by byla vytvořena nová instance třídy.[9]

[9] Vojtěch Merunka, Objektové modelování, Alfa, 2008, str.25-28

3.3 MVC architektura webových aplikací

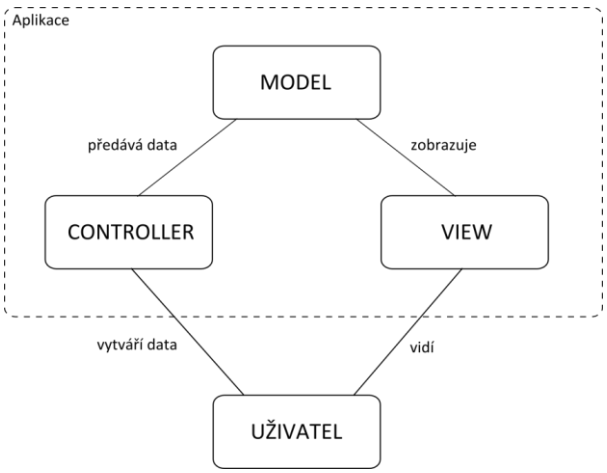
Model-View-Controller (dále jen MVC) je softwarová struktura skládající se ze tří základních komponent, již z názvu je patrné, že to jsou tyto:

- Model – hlavní logika aplikace
- View – zobrazení dat z Modelu uživateli
- Controller – řídící logika, zajišťuje manipulaci s daty

MVC je poslední dobou nejčastěji zmiňovaný návrhový vzor, jehož autorem je Trygve Reenskaug. První návrh sepsal již v roce 1979 v rámci svého působení v Xerox Palo Alto Research Center. Jeho práce popisuje problém uživatelské kontroly velkých a složitých datových struktur a okamžitě po zveřejnění vzbudila velký zájem. Zprvu T. Reenskaug nazval tuto architekturu Thing-Model-View-Editor. O půl roku později a po obsáhlých diskuzích, které vedl především s Adele Goldberg, matkou jazyka Smalltalk, překřtil Thing-Model-View-Editor na Model-View-Controller. Reenskaugova práce[10] je hlavním zdrojem teoretických informací v této podkapitole. Roku 1987 byl návrhový vzor MVC poprvé implementován, a to do jazyka Smalltalk, kde Model zajišťoval modelování vnějšího světa, View prezentovalo data a Controller rozděloval uživatelský vstup. Tou dobou se už T. Reenskaug na vývoji nepodílel. Touto implementací se postupně inspirovalo několik dalších projektů a došlo i na webové aplikace.

[10] REENSKAUG, Trygve. *Models-Views-Controllers* [online]. 2011 [cit. 2011-02-23]. Dostupné z WWW: <http://heim.ifi.uio.no/~trygver/1979/mvc-2/1979-12-MVC.pdf>.

Webové aplikace mají, na rozdíl od jiných implementací vzoru MVC, jedno specifikum. Vzhledem k tomu, že je veškerá aktivita realizována na základě klient-server komunikace, musí být aplikace schopna vytvořit výstup z URL [11] a HTTP hlaviček. Veškeré požadavky od uživatele prochází skrz Controller, na který jsou tím pádem kladeny vyšší nároky.



Obrázek 2 - Model-View-Controller

Jak je patrné z Obrázku 1, komponenty aplikace jsou od sebe zřetelně oddělené, což umožňuje samostatně vytvářet a spravovat každou komponentu zvlášť, tím pádem bez nežádoucích efektů.

Představme si MVC aplikaci jako model reálného světa – automobil. Automobil má navenek dva pohledy (Views) – interiér a exteriér. Modelem rozumějme volant, brzdy a další ovládací prvky. Řidič je v našem případě logickým prvkem, tedy

[11] URL = Uniform Resource Locator

Controllerem, který určuje, jak se bude automobil projevovat, a který vybírá Model, jenž tuto změnu zajistí a provede.

Jakákoliv událost typicky vyvolává změnu View, kterou obsluhuje Controller. Ideou MVC návrhu při vývoji webové aplikaci je mít jeden Controller, který zajišťuje spuštění akce založené na parametrech žádosti. Tato žádost má vždy minimálně tyto parametry – definuje, který Model volat, událost, která má nastat a obvykle i předávané GET[12] argumenty. Pokud je žádost validní, Model a požadovaný View existuje, bude spuštěna požadovaná akce.

O uživatelské rozhraní, které je u webových aplikací hlavním prezentačním nástrojem, se stará převážně View v součinnosti s Controllerem. Je tedy patrné, že od sebe oddělujeme model a prezentaci, což je vhodné z několika důvodů. Za prvé považuji za vhodné soustředit se při vývoji buď na zpracování uživatelského rozhraní, nebo na aplikační logiku. To s sebou přináší možnost programovat v týmu, přičemž se lze specializovat na tu či onu část projektu. Také se mnohem snadněji zapracovávají změny interface aplikace. Oddělení je dále vhodné při nutnosti přístupu k datům modelu zvenčí, například pomocí API[13].

[12] HTTP metoda GET zajišťuje předání formulářových dat jako součást URL – například parametry stránky jako je název článku či jeho ID
[13] API = Application Programming Interface

V úvodu této podkapitoly je uvedeno, které prvky MVC struktura obsahuje, nyní se s každým z nich seznámíme blíže.

3.3.1 **Model**

Model je hlavní logickou komponentou celé aplikace. Určuje jak pravidla chování modelovaného problému, tak i jeho data. Aplikaci poskytuje prostředky pro obsluhu dat (jejich ukládání, aktualizaci) a přístup ke stavům aplikace.

3.3.2 **View**

View je odpovědný za formátování a výstup dat, která přebírá od Modelu a dále je prezentuje uživateli aplikace pomocí, obvykle grafického, výstupu. Z povahy této komponenty je zřejmé, že View může být pro jeden model hned několik, záleží na tom, která data a jak mají být prezentována.

Kdykoliv se změní stav Modelu, dojde k aktualizaci zobrazení. Přístup k těmto změnám je možno realizovat dvěma způsoby, buď si View načítá aktuální data z Modelu sám, nebo Model posílá zprávy o změnách na základě registrace View.[14]

V případě webových aplikací je View úzce svázán s Controllerem, kterému předává události vyvolané uživatelem, například kliknutí myší do určitého pole.

[14] MICROSOFT. *Model-View-Controller* [online]. 2011 [cit. 2011-02-23]. Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/ff649643.aspx>.

3.3.3 **Controller**

Controller definuje, jak bude aplikace reagovat na určité požadavky vytvořené především interakcí s uživatelem. Zpracovává tak veškeré vstupy a události, na základě kterých poté volá příslušné metody Modelu. Podle událostí a výsledků metod vybírá vhodný View pro zobrazení výstupu.

Speciálním typem Controlleru je takzvaný Front Controller, který je odpovědný za směrování všech požadavků na příslušný Controller a vrácení výsledku.

Jak již bylo zmíněno, událostí je například kliknutí myší nebo jiná interakce uživatele s aplikací. Hlavním vstupem webových aplikací jsou GET a POST požadavky odeslané prohlížečem.

4 Aplikační frameworky

Ač jsou v dnešní době programovací jazyky pro tvorbu webových aplikací velmi vyspělé a poskytují vývojáři spoustu možností, není prakticky možné, aby dokázali uspokojit veškeré funkční požadavky projektu. Rozsah služeb aplikace se může velmi lišit. Záleží na tom, zda řešíme jednoduchou úlohu nebo vytváříme komplexní aplikaci, například on-line obchod, která obsahuje velké množství dat a funkcí.

4.1 Softwarové knihovny

Při vývoji pokročilé webové aplikace, mající velké množství funkčních požadavků, si musí vývojář potřebné funkce doprogramovat sám. Často se opakující úlohy je možné řešit pomocí hotových knihoven, které bývají volně k dispozici. Softwarová knihovna obsahuje zobecněné funkce, procedury, datové typy a zdroje a může být dobře sdílena napříč celou aplikací.

Knihovny pro PHP začaly vznikat už pro verzi 4 a to právě z důvodů výše uvedených. Za první vlaštovku se považuje repositář PHP Classes[15], který funguje do roku 1999 dodnes. Velmi důležitým projektem se stal balíčkovací systém PEAR[16], který je vyvíjen od listopadu roku 1999 a je oficiální součástí PHP od verze 4.3. Často používané jsou také šablonovací systémy, například Smarty[17], které umožňují oddělení aplikační a zobrazovací logiky.[18]

[15] PHP Classes Repository, http://www.phpclasses.org/
[16] PEAR = PHP Extension and Application Repository, http://pear.php.net/
[17] Smarty Template Engine, http://www.smarty.net/
[18] STOUPA, Václav. *Přehled a vývoj PHP frameworků* [online]. 2008 [cit. 2011-02-26]. Dostupné z WWW: <http://www.root.cz/clanky/prehled-a-vyvoj-php-frameworku/>.

Vzhledem ke vzrůstající oblíbeností knihoven pro PHP se začaly vytvářet i první frameworky. Problém ovšem byl ve slabé podpoře objektového programování v PHP4, největší oblíbenost si proto frameworky získávají až s příchodem nové verze jazyka.

4.2 Co je framework a proč ho používat

Technologická firma Maxxess ve svých materiálech [19] uvádí, že framework je znovupoužitelná softwarová struktura sloužící jako podpora při programování aplikací. Může obsahovat podpůrné programy, softwarové knihovny, šablonovací jazyky, které společně pokrývají funkční požadavky aplikací. Cílem frameworku je zajistit funkčnost systému, zatímco se vývojář soustředí jen na své zadání. Při samotném programování pak v případě potřeby pouze načítá vybrané knihovny.

Důvodů, proč používat aplikační frameworky, bychom mohli najít velké množství. Pokusím se zmínit ty největší výhody, kvůli kterým jsem já začal jeden z testovaných frameworků používat.

Vzhledem k tomu, že PHP nemá výrazná omezení, co se struktury kódu týče, je velmi snadné napsat špatný kód, což je samozřejmě nežádoucí. Použitím vhodného frameworku se chyb snáze vyvarujeme, vede nás k tomu totiž samotná jeho struktura a také proto, že dobré frameworky často obsahují nástroj pro report chyb (tady bych vyzdvihl „Laděnku“ Nette Frameworku, díky které je hledání a oprava chyb snadnou a sympatickou záležitostí).

[19] MAXXESS SYSTEMS. *What is a Software Framework?* [online]. 2011 [cit. 2011-03-12]. <http://www.maxxess-systems.com/whitepapers/what_is_a_software_framework.pdf>.

Obrázek 3 – Výpis chyby v třídě Nette/Debug, „Laděnce“

(Zdroj: http://zdrojak.root.cz/clanky/nette-framework-odvsivujeme/)

Dalším příjemným prvkem je fakt, že se programátor téměř nemusí starat o načítání dat z databáze, pouze si je vyžádá a framework už ostatní zařídí sám (pod podmínkou, že máme funkční Model).

V posledních letech se vývoj webu neopírá pouze o samotný obsah, ale jeho nedílnou součástí je i marketing. Web musí být tím pádem dobře přístupný vyhledávačům, proto se webové prezentace píší s ohledem na SEO[20]. Není náplní této práce popisovat metody optimalizace pro vyhledávače, ale je vhodné zmínit alespoň zásadu krátkých a nejlépe neměnných URL adres (v angličtině se užívá příhodnějších

[20] SEO = Search Engine Optimization

názvů Pretty URLs nebo také User-Friendly URL). Dodržení této zásady výrazně zjednodušuje Router. Jeho úkolem je generovat URL z interních požadavků a také URL adresu rozkládat (parsovat) a tím požadavek získat.
Pro zachování zpětné kompatibility slouží routery jednosměrné, které zpracovávají již dříve existující odkazy a tím zachovávají jejich funkčnost. Navíc automaticky přesměrovávají na nový tvar URL adresy, odkaz takzvaně kanonizují (přesměrovávají na výchozí formu, kterou generuje Router).[21]

[21] NETTE FOUNDATION. *Nette Framework* [online]. 2011 [cit. 2011-02-04]. Routování. Dostupné z WWW: <http://doc.nette.org/cs/routovani>.

4.3 Zend Framework

Zend Framework [22] je pravděpodobně nejznámější a zároveň nejpoužívanější PHP framework. Má jednu velmi podstatnou výhodu, co se vývoje týče a to, že je od samého začátku, tedy roku 2005, podporován přímo vývojáři samotného PHP. Na Zendu jsou typické balíčky, které jsou samostatně použitelné a třídy mají striktně dané pojmenování. Záporem se zdá být přílišná složitost zdrojového kódu, která však vychází ze samotné syntaxe PHP. Také rychlost zaostává, výraznějšímu zpomalení je však možno zabránit použitím Zend Optimizeru či Zend Acceleratoru na straně serveru. V každém případě je připravena cache, kterou disponuje většina lepších frameworků.

Obrázek 4 – Logo Zend Framework

(Zdroj: http://framework.zend.com/)

Z mého pohledu je Zend Framework vhodný spíše pro velké a náročné projekty. Použití Zendu na menší a jednodušší projekt nedoporučuji a souhlasím s přirovnáním, že psát v Zendu malou aplikaci je jako lovit komára s tankem. Početná komunita programátorů (celosvětově asi největší) okolo něj však upozorňuje na možnost použití balíčků

[22] Zend Framework, http://framework.zend.com/

samostatně a Zend doporučuje i pro malé projekty. Našel oblibu i u českých vývojářů, proto je možné najít řešení obvyklých problémů i v češtině. V případech, kdy nestačí databáze již vyřešených problémů, tu je propracovaná dokumentace.

Základní vlastnosti Zend Frameworku uvedu pouze heslovitě:

- Velké množství modulů, například pro administraci (Zend_Acl), export PDF dokumentů (Zend_Pdf) nebo podpora e-mailů (Zend_Mail)
- Snadná konfigurace pomocí souborů .ini nebo .xml
- Šablonovací systém Smarty nebo kombinace HTML a PHP ve formě .phtml
- Široká podpora databází (typicky MySQL, MSSQL, Oracle a další)

4.4 CakePHP

CakePHP[23] je na rozdíl od Zendu velmi jednoduchý, má přehledný kód, který je velmi vhodně členěn na kratší celky. To dle mého názoru pramení z inspirace v jazyce Ruby on Rails. CakePHP je obecně vhodný pro řešení standardních problémů, v případě složitějších aplikací je možná i integrace se Zendem, což umožňuje zkombinovat robustnost Zendu s jednoduchostí CakePHP.

[23] CakePHP, http://cakephp.org/

Obrázek 5 – Logo CakePHP Framework

(Zdroj: http://www.ydeveloper.com/)

Vlastnosti CakePHP ve zkratce:

- Databázová vrstva typu active record
- Propracované View helpery
- Krátký a přehledný zdrojový kód

4.5 **Symfony**

První verzi Symfony vydal roku 2005 Fabien Potencier, výkonný ředitel webové agentury Sensio. Ačkoli se v prostředí vývoje webových aplikací pohyboval delší dobu, stále nemohl najít vhodný open-source nástroj. S vydáním PHP5 se rozhodl vytvořit vlastní framework. Při vývoji jádra vycházel z MVC frameworku Mojavi, ORM[24] nástroje Propel a z šablonovacích helperů z Ruby on Rails. Tento framework původně

[24] ORM = Object-relational mapping

vytvářel pro potřeby firmy Sensio, po jeho několikerém použití se však rozhodl ho uvolnit pod open-source licencí.[25]

Obrázek 6 – Logo Symfony Framework

(Zdroj: http://symfony-project.org/)

Velmi brzy po spuštění projektu se o framework začali zajímat programátoři celého světa a k dobře zpracované dokumentaci tak přibyla i poměrně obsáhlá komunita. Symfony má veřejný repozitář, čímž je otevřen možnosti podílet se na projektu. Nad hlavním kmenem však stále vládne Fabien Potencier a zajišťuje tak kvalitu kódu.

Vlastnosti Symfony v bodech:

- Magické metody zastiňující chování tříd bez nutnosti jejich úprav
- Objektově relační mapování – překlad objektové logiky na relační
- Díky ORM snadný přechod na jiný databázový systém
- Možnost využití formátu YAML[26]
- Integrovaná podpora pro AJAX

[25] PLATINA DESIGNS. *What is Symfony?* [online]. 2011 [cit. 2011-03-24]. Dostupné z WWW: <http://www.platinadesigns.nl/en/what-is-symfony>.
[26] YAML, http://www.yaml.org/

4.6 **Nette framework**

Nette Framework [27] je dalším z výkonných nástrojů pro tvorbu webových aplikací v PHP5. Vznikl už v roce 2004, ale uvolněn byl až roku 2008 jako open-source. Od roku 2009 vývoj Nette zajišťuje a komunitu zastřešuje Nette Foundation[28]

Jeho autorem je David Grudl, český programátor a publicista, který stojí nejen za vznikem a vývojem tohoto MVC frameworku. Mezi jeho další open-source projekty patří Texy![29], PHP knihovna pro převod čistého textu na validní HTML či XHTML kód. Je založena na důležitosti logické struktury v (X)HTML a na čitelnosti zdrojového kódu i pro obyčejného člověka (neprogramátora). Druhým velkým projektem je databázový layer Dibi [30] , ten umožňuje maximálně zjednodušit a zpřehlednit zápis SQL příkazů. Velkou výhodou je jeho přenositelnost mezi databázovými systémy a snadný přístup k metodám. Používat kombinaci Nette+Dibi+Texy! je tedy velmi efektivní a díky ní je možno dosáhnout dobrých výsledků.

Největší výhodou Nette pro českého programátora je početná česká komunita. Ta samozřejmě ani zdaleka nedosahuje velikosti komunity okolo Zendu, ale právě přítomnost mateřského jazyka v příkladech a dokumentaci a především možnost řešení problémů na aktivním fóru je k nezaplacení.

[27] Nette Framework, http://www.nette.org/
[28] Nette Foundation, http://www.nettefoundation.com/
[29] Texy!, http://www.texy.info/
[30] Dibi, http://www.dibiphp.com/

Obrázek 7 – Logo Nette Framework

(Zdroj: http://nette.org/)

Nette využívá vlastní šablonovací systém, je řízené událostmi a z velké části využívá komponenty. Mezi jeho nejsilnější vlastnosti osobně řadím výše zmíněnou „Laděnku“, tedy třídu Nette/Debug, dále pohodlnou tvorbu webových formulářů pomocí třídy Nette/Forms, zabezpečení aplikací před XSS (Cross-site scripting), CSRF (Cross-site request forgery) a zneužití Sessions (Session hijacking, Session stealing). Pro bezproblémové nastavení serveru a především otestování požadavků obsahuje distribuční balíček Nette nástroj Requirements Checker. Ten ověří prostředí serveru a podá informaci o tom, zda lze framework používat.

Obrázek 8 - Nette Framework Requirements Checker

5 Praktický test vybraných frameworků

5.1 Úvod k testování

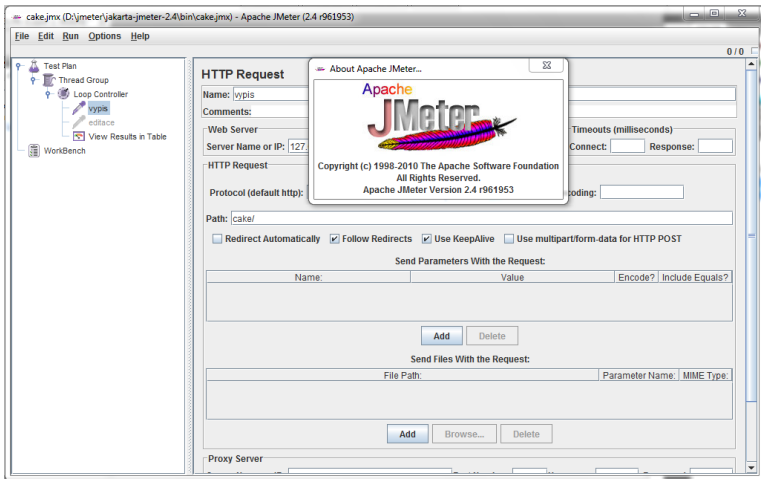
V současné době existuje velké množství PHP frameworků, které navenek umožňují prakticky totéž – oddělení aplikační logiky od samotného zobrazení, řeší nejčastější problémy a výrazně zkracují čas potřebný k napsání aplikace. Můžeme je rozdělit podle několika kritérií, například podle objektovosti, použití návrhového vzoru či velikosti. Není však možné je objektivně hodnotit a srovnávat, ve většině případů totiž nenajdeme více společných prvků tak, aby mělo číselné srovnání smysl. Každý framework nabízí jiné knihovny, liší se jejich množství, záleží i na použitých databázových vrstvách a podobně. Další vlastnosti nejsou kvantifikovatelné. Okolo snad každého z frameworků existuje komunita, některé jsou větší, jiné zase aktivnější. Srovnávat můžeme i kvalitu dokumentace a příkladů nebo křivku učení. Na těchto faktorech totiž většinou záleží doba sžití se s frameworkem, případně doba práce na projektu v případě nedostatečných znalostí.

Pro tuto práci jsem zvolil jednoduché testování rychlosti vybraných frameworků, konkrétně Zend Frameworku, Nette, CakePHP a Symfony. Pro každý z nich jsem se pokusil napsat testovací aplikaci, která načítá data z databáze a následně je zobrazuje. Nemělo by smysl zjišťovat jejich rychlost na statických stránkách, jelikož při ostrém použití aplikace je databáze v současné době nezbytností. Kromě výsledků testování a jejich vyhodnocení se zmíním i o praktických zkušenostech z tvorby aplikací.

5.1.1 **Prostředí pro testování, použitý software**

Pro testování bylo vzhledem k nárokům programu Apache JMeter[31] použito dvou počítačů vzájemně propojených pomocí místní sítě.

Open-source software Apache JMeter je vhodný nástroj pro měření výkonu webových aplikací (HTTP/HTTPs), relačních databází a FTP serverů. JMeter je součástí projektu Apache Jakarta Project, který se zabývá vývojem open-source software pro platformu JAVA. Jeho pomocí si můžeme vytvořit test pro naši aplikaci tak, abychom simulovali reálnou zátěž serveru.



Obrázek 9 - Prostředí programu Apache JMeter

[31] Apache JMeter, http://jakarta.apache.org/jmeter/

Pro úspěšné testování webové aplikace je nezbytné vložit element ThreadGroup, který určuje počet vláken (reálně uživatelů), které aplikaci navštíví, kolikrát se bude návštěva opakovat a v jakém časovém intervalu se budou spouštět. Dalším elementem je logická komponenta Loop Controller, která zajišťuje možnost opakování dotazů. V Loop Controlleru se v mém případě nachází ještě dva dotazy typu HTTP Request, první se dotazuje na stránku s tabulkou uživatelů, druhý pak načítá data konkrétního uživatele k úpravě. Podstatnou součástí, bez které by nemohl být test vyhodnocen, je minimálně jeden Listener. Listener zajišťuje odchytávání výsledků a jejich zobrazení v tabulce, grafu a dalších reprezentace získaných dat. Ve svém testu využívám zobrazení v tabulce (View Results in Table), který pro potřeby získání požadovaných hodnot postačuje.

5.1.2 **Praktický test Zend Frameworku**

K prvnímu testu jsem záměrně vybral Zend Framework. Z testovaných frameworků je nejrobustnější a očekávám od něj spíše horší výsledky ve srovnání s ostatními. Pomalejší zpracování je obecně možné přičítat množství a rozsáhlosti knihoven, které obsahuje.

Při prvním setkání se Zend Frameworkem jsem byl přesvědčen, že má velmi dobrou a propracovanou dokumentaci. Největší důležitost však v začátcích přikládám spíše zpracovaným tutoriálům a hotovým řešením a těch je, k mému překvapení, velmi málo. Téměř jediným použitelným začátkem pro orientaci v Zendu je tutoriál Akrabat[32]

[32] Tutoriál Akrabat, http://akrabat.com/zend-framework-tutorial/

od Roba Allena, který je často napodobován a převáděn i do dalších frameworků. Při tvorbě aplikace se začínající programátor beztak neobejde bez hledání v dokumentaci. Četnost zdrojů a jejich dohledatelnost hodnotím jako průměrnou.

ZENDF	DOTAZŮ	PRŮMĚR(MS)	MIN(MS)	MAX(MS)	MEDIÁN	MODUS
1. test						
zobrazení	50	228,38	204	245	229	228
úprava	50	281,22	263	302	281,5	285
celkem	100	509,6	467	547	510,5	
2. test						
zobrazení	50	222,68	208	299	220,5	218
úprava	50	260,06	234	382	258,5	258
celkem	100	482,74	442	681	479	
3. test						
zobrazení	50	215,84	201	230	215	214
úprava	50	258,5	246	271	258	258
celkem	100	474,34	447	501	473	

Tabulka 1 – Zend Framework - výsledky testu

Výsledky testu vyzněly pro Zend Framework překvapivě dobře. Není totiž obecně řazen mezi ty rychlejší, ale je jak vidno vhodný i pro jednodušší úkoly. Testovacím serverem nebyl v tomto případě Zend Server, který se pro použití tohoto frameworku doporučuje, ani nebyl nainstalován plugin Zend Optimizer. Předpokládám, že výsledky by v takovém případě byly ještě příznivější. Celkově se Zend Framework umístil na druhém místě.

5.1.3 **Praktický test CakePHP**

Dalším testovaným zástupcem je CakePHP, framework, který sám nejčastěji používám. Vybral jsem si ho s ohledem na dobrou dokumentaci, velké množství praktických ukázek a jednoduchost. Na CakePHP se mi líbí, že nevnucuje programátorovi jeden jediný způsob psaní kódu, když není frameworku potřeba obejdete se bez jeho funkcí. Hodí se i pro tvorbu statických stránek, jelikož je možné ho používat bez přístupu k databázi, která u malých prezentací není často zapotřebí. Vadou na kráse se zdá být striktní pojmenování souborů, tříd a adresářů, které je pro správný chod aplikace nutností.

CakePHP je v tomto testu mým favoritem, co se dokumentace a tutoriálů týče. Nejobsáhlejším zdrojem návodů je CakePHP Cookbook[33]. Dobře se v něm orientuje a velkou výhodou je především použitý systém tvorby článků – wiki. Najdeme zde jak podrobně popsané komponenty, tak jednoduché ukázkové aplikace, bez nichž si proniknutí do daného frameworku nedokážu dobře představit. Zajímavostí je použití kuchařských termínů v komunitním prostředí, což vychází už z názvu samotného frameworku. Aplikace se tedy neprogramují, ale pečou, dokumentaci se říká Kuchařka nebo Pekárna, která je založena spíše na tvorbě obsahu uživateli.

Stažení a zprovoznění CakePHP je velmi jednoduché a rychlé, základní kostra aplikace se pouze kopíruje do kýženého adresáře. Jediný problém nastane při použití na PHP5.3, kde je nutné ještě nastavit výchozí časovou zónu.

[33] CookBook, the CakePHP documentation, http://book.cakephp.org/

CAKEPHP	DOTAZŮ	PRŮMĚR(MS)	MIN(MS)	MAX(MS)	MEDIÁN	MODUS
1.test						
zobrazení	50	242,98	218	360	236	236
úprava	50	252,3	231	351	244	244
celkem	100	495,28	449	711	480	
2.test						
zobrazení	50	242,66	220	321	233,5	230
úprava	50	249,94	227	350	242,5	244
celkem	100	492,6	447	671	476	
3.test						
zobrazení	50	240,7	217	311	235	233
úprava	50	255,3	231	525	243,5	242
celkem	100	496	448	836	478,5	

Tabulka 2 – CakePHP – výsledky testu

CakePHP je známý svou komplexností a strmou křivkou učení. I úplný začátečník (se základními znalostmi jazyka PHP) by měl být schopný po chvíli zprovoznit svou první aplikaci. Stejně tak je známé, že generování stránek není právě nejsvižnější. V testu obsadil CakePHP třetí místo, v těsném závěsu za Zendem, přesto ho pro začátky s PHP frameworky doporučuji.

5.1.4 Praktický test Symfony

Symfony je mi z testovaných webových frameworků nejméně sympatický, ač se v současnosti používá ve velké míře. Důvodem mé averze vůči němu je nemožnost spravovat projekt jinak než přes příkazovou řádku. Nevýhodou pro začínajícího programátora je nemožnost použití Symfony na sdíleném webovém hostingu, právě kvůli zmíněné příkazové řádce.

Po problémech s instalací přichází další zklamání – dokumentace. Ač se Symfony tváří, že má dokumentaci dostatečně obsáhlou (*„You want to learn more about Symfony? We have plenty of documentation for you.“)*, zdání klame. Instalace je sice dobře rozebrána, ale s dalšími kroky při vývoji začínají návody pokulhávat. Ani konfigurace pomocí YAML souborů, která je často vychvalována, mi k srdci nepřirostla, v některých případech trvá pochopení delší dobu.

Vzhledem k použití automaticky generovaného administračního rozhraní jsem v tomto testu neodděloval zobrazení tabulky a editaci uživatelů. Výsledky jsou proto souhrnné a mohou být zkreslené.

SYMFONY	DOTAZŮ	PRŮMĚR(MS)	MIN(MS)	MAX(MS)	MEDIÁN	MODUS
1.test						
celkem	100	847,48	827	896	845	850
2.test						
celkem	100	843,48	809	925	833,5	827
3.test						
celkem	100	824	801	901	820	804

Tabulka 3 – Symfony – výsledky testu

Symfony dopadlo v praktickém testu nejhůře, mezi hojně používanými frameworky patří k nejpomalejším. I přes možné zkreslení výsledků jsou naměřené hodnoty velmi špatné a pro malou webovou prezentaci bych Symfony právě kvůli rychlosti nedoporučil. U velkých projektů však může být vhodné, vzhledem k produktivitě a rychlosti vývoje.

5.1.5 **Praktický test Nette Framework**

Důvodů, proč používat Nette Framework je hned několik. Nacházíme u něj skvělé řešení věcí vývojáři hojně používaných – jednoduchá tvorba formulářů včetně potřebného ošetření, obsluha databáze je snadná díky vrstvě Dibi, skvěle zvládnuté ladění, šablonovací systém a cachování.

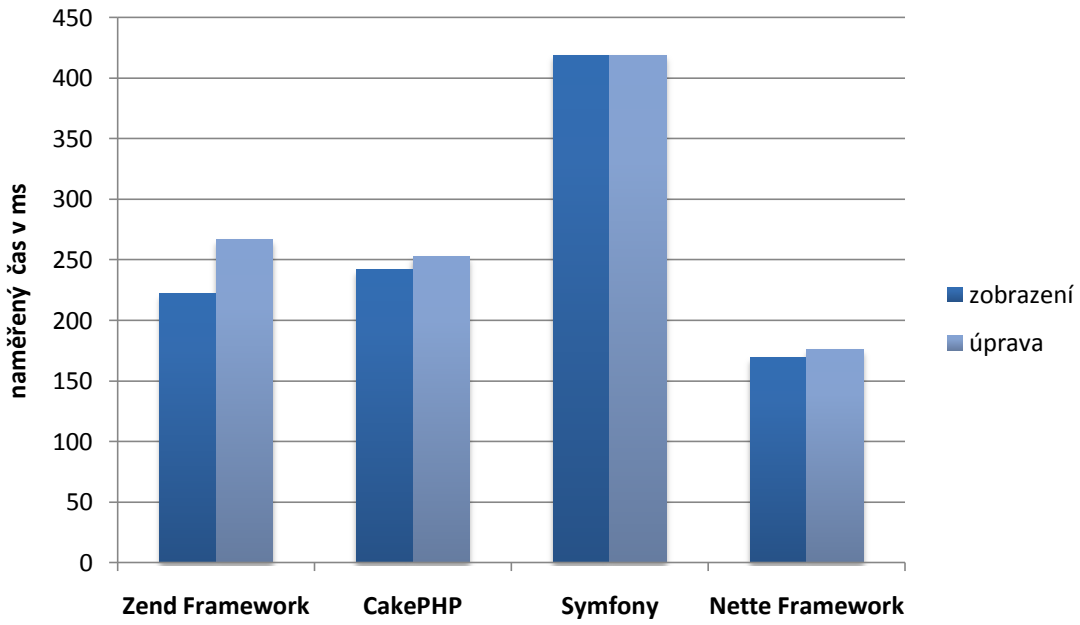
Dovolte mi citovat autora projektu Davida Grudla: „*Jak úžasné věci jsem mohl udělat, kdybych nebyl tak línej…“*. Myslím, že právě lenost se nejvíce podepsala (v dobrém slova smyslu) na úspěchu Nette. Žádné zbytečné opakování kódu, snadná obsluha a vysoká efektivita práce.

NETTE	DOTAZŮ	PRŮMĚR(MS)	MIN(MS)	MAX(MS)	MEDIÁN	MODUS
1.test						
zobrazení	170,98	162	228	170	171	170,98
úprava	181,78	172	242	178,8	178	181,78
celkem	352,76	334	470	348,8		352,76
2.test						
zobrazení	169,42	162	185	167	165	169,42
úprava	174,65	167	193	174	175	174,65
celkem	344,07	329	378	341		344,07
3.test						
zobrazení	168,74	150	186	167	172	168,74
úprava	172,77	154	191	170,8	174	172,77
celkem	341,51	304	377	337,8		341,51

Tabulka 4 – Nette Framework – výsledky testu

Nette Framework, jediný český zástupce, se stal posledním testovaným a zároveň vítězným frameworkem celého testu. V aplikaci byla použita jednosouborová verze frameworku, která podle autora minimalizuje nároky pro načtení knihoven. Tento fakt ale naměřené hodnoty nezkreslil, neboť rozdíly nejsou oproti klasické verzi velké a především je zvykem na produkční servery nasazovat právě tuto jednosouborovou

verzi. Velkou měrou se na dobrém výsledku podílí i fakt, že na optimalizaci se u Nette opravdu hledí, důležitá je i podpora cachování.



Obrázek 10 – Porovnání rychlostí frameworků v grafu

V závěru vyhodnocení testu je třeba upozornit na fakt, že jsem s dvěma testovanými frameworky (Zend Framework a Symfony) neměl žádnou předchozí zkušenost a jejich hodnocení je založeno pouze na krátkodobých poznatcích.

6 Závěr

Cílem této práce bylo seznámení se softwarovými frameworky, které je vhodné používat při tvorbě webových aplikací. Programátor se tak snáze vyvaruje opakovanému psaní stejné části kódu napříč aplikací a výrazně zvýší svoji produktivitu. V úvodu rešeršní části jsem proto popsal základ tvorby webových aplikací, od historického vývoje jazyka PHP přes jeho objektové možnosti v nových verzích po jednu z nejpoužívanějších softwarových struktur – MVC architekturu. Hlavní část práce pak byla věnována PHP frameworkům, jejichž používání je podle mého názoru v současné době nezbytností. Pokusil jsem se toto stanovisko podpořit výčtem výhod aplikace frameworku na webový projekt a představil čtyři vybrané zástupce: Zend Framework, CakePHP, Symfony a Nette Framework. V těchto jsem posléze napsal ukázkové aplikace, které jsou přiloženy na CD, a otestoval je s ohledem na rychlost. Tento postup je popsán v praktické části práce, která je věnována především testování aplikací a hodnocení frameworků. Z toho testu vyšel nejlépe český Nette Framework, a to nejen z hlediska rychlosti, ale také díky tomu, že s jeho pomocí lze pohodlně a rychle tvořit pokročilé webové aplikace.

7 Seznam použitých zdrojů

BRÁZA, Jiří. *PHP4 učebnice základů jazyka*.: Grada, 2002. 224 s. ISBN 80-247-0442-0.

BRÁZA, Jiří. *PHP5 začínáme programovat*.: Grada, 2005. 244 s. ISBN 80-247-1146-X.

CAKE SOFTWARE FOUNDATION. *CakePHP Cookbook* [online]. 2011 [cit. 2011-01-20]. Dostupné z WWW: <http://book.cakephp.org>.

DLOUHÝ, Radek. *PHP v příkladech*.: Computer Media, 2007. 180 s. ISBN 80-86686-83-3.

LAVIN, Peter. *PHP objektově orientované*.: Grada, 2009. 224 s. ISBN 978-80-247-2137-8.

LEISS, Oliver; SCHMIDT, Jasmin. *PHP v praxi*.: Grada Publishing, 2010.242 s. ISBN 978-80-247-3060-8.

MAXXESS SYSTEMS. *What is a Software Framework?* [online]. 2011 [cit. 2011-03-12]. Dostupné z WWW: <http://www.maxxess-systems.com/whitepapers/what_is_a_software_framework.pdf>.

MCARTHUR, Kevin. *Pro PHP: Patterns, Frameworks, Testing and More*.: Apress, 2008. 349 s. ISBN 978-1-59059-819-1.

MERUNKA, Vojtěch. *Objektové modelování.:* Alfa, 2008. 197 s. ISBN 978-80-87197-04-2.

MICROSOFT. *Model-View-Controller* [online]. 2011 [cit. 2011-02-23]. Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/ff649643.aspx>.

NETTE FOUNDATION. *Nette Framework* [online]. 2011 [cit. 2011-02-04]. Dokumentace. Dostupné z WWW: <http://nette.org/cs/dokumentace>.

PLATINA DESIGNS. *What is Symfony?* [online]. 2011 [cit. 2011-03-24]. Dostupné z WWW: <http://www.platinadesigns.nl/en/what-is-symfony>.

REENSKAUG, Trygve. *Models-Views-Controllers* [online]. 2011 [cit. 2011-02-23]. Dostupné z WWW: <http://heim.ifi.uio.no/~trygver/1979/mvc-2/1979-12-MVC.pdf>.

STOUPA, Václav. *Přehled a vývoj PHP frameworků* [online]. 2008 [cit. 2011-02-26]. Dostupné z WWW: <http://www.root.cz/clanky/prehled-a-vyvoj-php-frameworku/>.

SYMFONY PROJECT. *Symfony Documentation* [online]. 2011 [cit. 2011-03-08]. Dostupné z WWW: <http://www.symfony-project.org/doc/1_4/>.

THE PHP GROUP. *History of PHP* [online]. 2011 [cit. 2011-01-29]. Dostupné z WWW: <http://www.php.net/manual/en/history.php.php>.

TICHÝ, Jan. *Programová podpora tvorby webových aplikací* [online]. 2004. 66 s. Diplomová práce. Vysoká škola ekonomická.

ZEND TECHNOLOGIES. *Zend Framework Documentation* [online]. 2011 [cit. 2011-03-08]. Dostupné z WWW: <http://framework.zend.com/docs/>.

8 Seznam zkratek

API = Application Programming Interface

HTML = HyperText Markup Language

HTTP = Hypertext Transfer Protocol

ORM = Object-relational mapping

PHP = Hypertext Preprocessor, historicky Personal Home Page

SEO = Search Engine Optimization

URL = Uniform Resource Locator

9 Seznam obrázků a tabulek