Univerzita Karlova
Pedagogická fakulta

BAKALÁŘSKÁ PRÁCE

2017 Anton Hatala

Univerzita Karlova

Pedagogická fakulta

Katedra informačních technologií a technické výchovy

BAKALÁŘSKÁ PRÁCE

Porovnání webových frameworků

Web frameworks comparison

Anton Hatala

Vedoucí práce:	PhDr. Josef Procházka, Ph.D.
Studijní program:	Specializace v pedagogice
Studijní obor:	Informační technologie se zaměřením na vzdělávání

2017

Prohlašuji, že jsem bakalářskou práci na téma Porovnání webových frameworků vypracoval pod vedením vedoucího práce samostatně za použití v práci uvedených pramenů a literatury. Dále prohlašuji, že tato práce nebyla využita k získání jiného nebo stejného titulu.

Praha 21.4.2017

.....

podpis

Chtěl bych poděkovat vedoucímu práce PhDr. Josefu Procházkovi, Ph.D. za cenné rady, ochotu a trpělivost, kterou mi věnoval.

ANOTACE

Bakalářská práce představuje a porovnává tři webové frameworky - Nette, Zend a Symfony. Představuje frameworky za pomoci vybrané literatury a porovnává samotné webové frameworky pomocí třech identických aplikací. Modelové aplikace jsou vytvořeny třemi výše zmíněnými webovými frameworky. Podle metodiky, která je popsána ve třetí kapitole bakalářské práce, se provádí měření. Výsledné hodnoty se porovnají a určuje se tím nejvhodnější framework pro vývojáře. Cílem práce je pomoci vývojáři vybrat si správný webový framework s ohledem na jeho potřeby. V první části práce je definováno, co se rozumí pod pojmem webový framework a podrobně jsou zde popsány vybrané frameworky. V druhé části je specifikováno, jakou metodikou a metrikou se hodnotí webové frameworky. V poslední části je vybraná metrika aplikována na jednotlivé frameworky a pomocí modelových aplikací testována.

KLÍČOVÁ SLOVA

Nette Framework, Zend Framework, Sympfony, PHP, framework, MVC, MVP model, pohled, kontrolor, presenter, bezpečnost, SQL Injection, Cross Site Request Forgery, Cross Site Scripting.

ANNOTATION

Bachelor thesis presents and compares three web frames - Nette, Zend and Symphons. It frames using selected literature and compares the web framework itself with three identical applications. Model applications are created by three of the above web frames. According to the methodology described in the third chapter of the bachelor thesis, measurements are made. The resulting values are compared and determined by the most appropriate development framework. The aim of this diploma thesis is to help developers choose the right web framework for their needs. The first part of the work defines what the web framework means and detailed frames are described in detail here. The second part specifies web framework methodology and metrics. In the last part, the selected metric is applied to individual frames and tested using model applications.

KEYWORDS

Nette Framework, Zend Framework, Sympfony, PHP, framework, MVC, MVP model, pohled, kontrolor, presenter, bezpečnost, SQL Injection, Cross Site Request Forgery, Cross Site Scripting.

Obsah

1 Úvod a cíle práce

1.1 Úvod

Od první webové stránky, kterou vytvořil roku 1991 Tim Berners-Lee (Daven Hiskey, 2010), uběhlo už několik let a webové technologie se posunuly mílovými kroky kupředu. Vyjmenuji jen několik technologií z mnoha, které jsou aktuálně nezbytnou součástí webových aplikací, a které se neustále vyvíjejí. Značkovací jazyk HyperText Markup Language (HTML), který umožňuje vytvářet a pomocí hypertextových odkazů i spojovat webové stránky. Cascading Style Sheets (CSS) neboli kaskádové styly, které webovým stránkám určují vzhled. Skriptovací jazyk PHP, díky kterému se webové stránky stávají dynamické. V dnešní době ale neexistují pouze tyto technologie. Je jich nespočet a proto je třeba mezi nimi pozorně vybírat.

Programátoři, kteří vyvíjejí již několikátou aplikaci, by měli narazit na části kódu, které třeba již dříve napsali. K tomu, aby si programátor ulehčil život a zefektivnil práci je zapotřebí začít psát určité předlohy. Některým to trvá déle, některý se do toho pustí hned. Díky předlohám vývojář nemusí zdlouhavě opisovat kód, který kdysi již použil, ale postačí pouze odkázat na danou předlohu. Framework je soubor těchto šablon, které pracují nad určitým programovacím jazykem a pomáhají se programátorovi soustředit na logiku, místo psaní rutinních úloh.

Jenomže stejně jako to platí u technologií, tak i webových frameworků je mnoho. Pro bakalářskou práci jsem proto vybral tři frameworky, na které s vysokou pravděpodobností může vývojář narazit, když bude hledat vhodný webový frameworku pro svojí aplikaci. Ty pak podrobím testu, zhodnotím a určím jejich výhody a nevýhody.

1.2 Cíle práce

Jak jsem již zmínil, existuje mnoho webových frameworků a žádný vývojář by tedy neměl na internetu stáhnout framework, na který odkazoval první odkaz ve vyhledávači. Je třeba řádně prostudovat, jak daný framework funguje a zjistit jaké jsou jeho výhody a nevýhody. Může se stát, že se vývojář rozhodne pro jeden webový framework, ale po určité době zjistí, že mu nevyhovuje, například z důvodu nedostatečného zabezpečení. Pokud se tak stane,

může vzniknout mnoho problémů a vývojář tím ztratí mnoho času. Takové situaci bych chtěl výsledným hodnocením webových frameworků zabránit.

Touto bakalářskou prací bych chtěl především ověřit deklarované vlastnosti frameworku, vyhodnotit jejich zásadní vlastnosti a tím ulehčit práci všem vývojářům, kteří chtějí vytvářet svoji aplikaci pomocí webového frameworku, ale nemůžou se rozhodnout který zvolit. Jen málo lidí má čas na procházení webových stránek a studování vlastností jednotlivých frameworků. Vývojáři, kteří váhají se můžou podívat do závěru této bakalářské práce, kde najdou odpověď na svoji otázku, který framework je pro jejich webovou aplikaci nejvhodnější.

V práci se zabývám detailně vybranými frameworky, jejich návrhovými vzory, ale popisuji i to, co takový webový framework vlastně je. Cílem práce je porovnat webové frameworky těmi nejzákladnějšími, ale přitom nejdůležitějšími vlastnostmi, které řeší vývojář při jejich výběru. Ve třetí kapitole se věnuji seznamu všech těchto základních aspektů. Aspekty budou hlavními body pro porovnávání v mnou vytvořených aplikacích. Vytvořené aplikace vybranými frameworky budou co nejvíce identické, aby porovnávání mezi nimi bylo relevantní.

2 Framework

2.1 Webový Framework

"Framework je softwarová struktura, která slouží jako podpora při programování a vývoji a organizaci jiných softwarových projektů. Může obsahovat podpůrné programy, knihovnu API[1], návrhové vzory nebo doporučené postupy při vývoji." (Škrášek, 2008)

Vývojáři se mohou začít soustředit na jádro problému projektu a nezabývat se běžnými problémy. O ty se postará právě framework. Mluvíme tu například o rutinách jako připojování k databázi, změny databází, zpracování šablon, validace stránek a formulářů, bezpečnosti nebo vytváření SEO URL[2] adres, které nám pomáhají dosahovat lepšího umístění ve vyhledávačích. Funkčnost a úroveň zmíněných rozšíření se u každého frameworku liší a každý z nich se zaměřuje na něco jiného. "Frameworků je v dnešní době jako máku" (Böhmer, 2010, s. 23) a vývojář se musí rozhodnout, jaký je pro jeho projekt nebo systém nejvhodnější.

Každý framework s sebou přináší nové názvy funkcí, tříd a metod. Nejsou to proměnné, které patří do daného programovacího jazyka (například PHP), ale názvy pro framework specifické. Při správném zvolení názvu to pro nás znamená volání konkrétní akce. Pro vývojáře to znamená, že se musí seznámit s frameworkem a pochopit jeho princip fungování.

2.2 K čemu slouží

Hlavním cílem frameworku je usnadnit práci webovým programátorům tím, že je zbaví zabýváním se rutinních prací a pomocí knihovny API poskytne prostor pro jednoduchá a efektivní řešení daného problému.

---

[1] Application Programming Interface označuje v informatice rozhraní pro programování aplikací. Jde o sbírku procedur, funkcí, tříd či protokolů nějaké knihovny (ale třeba i jiného programu nebo jádra operačního systému), které může programátor využívat. API určuje, jakým způsobem jsou funkce knihovny volány ze zdrojového kódu programu. (ITBiz) Dostupné z: http://www.itbiz.cz/slovnik/informacni-technologie-it/api

[2] Zkratka pro Search Engine Optimization Uniform Resource Locator. Znamená to vytvořit takové webové adresy, kterou jsou "použitelnější a zapamatovatelnější a pozitivně přispívají k SEO." (Nette Foundation, 2008-2017) Dostupné z: https://doc.nette.org/cs/2.4/routing

Framework musí řešit několik desítek problémů, které mají právě usnadnit programátorovi práci, ale tím celá struktura a složitost frameworku roste. Je proto otázkou, zda na malé a střední projekty využívat framework. Většinou jsou v takových případech projekty s frameworkem vnímány negativně. Programátor se musí učit novým příkazům, kódování a zpracování je pomalejší a ve většině případech se ani plně nevyužijí všechny výhody frameworku.

Naopak velice pozitivně jsou vnímány větší projekty, které výhody frameworku využijí naplno. Rozsáhlý projekt využije celý potenciál daného frameworku a tím jen málo jeho funkcí zůstane nevyužitých. Framework nabídne programátorovi řešení pro rutinní operace a tím ušetří jeho čas, který on může využít jinde. Opakované činnosti jsou mnohdy zjednodušené a efektivnější.

2.3 Návrhové vzory

Každého vývojáře musí dřív nebo později dostihnout složitost aplikací a on si už nebude moci dovolit psát vše dohromady. Aplikace je pak nepřehledná, složitá a pozdější úpravy jsou časově náročné. Je třeba rozdělit logické funkce a výstupní funkce. Návrhové vzory slouží k tomu, aby rozdělily kód a daly mu určitou strukturu - rozdělení na komponenty. Výhodou bude bezesporu lepší přehlednost kódu, udržitelnost a zlepšení komunikaci mezi programátory. Podle Čápka se tím vyřeší tzv. "špagetový kód", kdy máme v jednom souboru jak logické funkce, tak i renderování výstupu - nesrozumitelný a neudržovatelný programovací kód.

2.3.1 MVC

MVC (Model-View-Controller) vznikl na desktopech, ale nyní je jedním z oblíbených návrhových vzorů, který se využívá při vytváření webových stránek. Používá ho drtivá většina webových frameworků, mezi kterými jsou Symfony a Zend framework. MVC architektura je rozdělena do třech "komponent, hovoříme o modelu, view (pohledu) a controlleru (kontroleru)." (Čápka)

Model

V návrhovém modelu MVC je model logickou částí. Obsahuje logiku, takže to jsou výpočty, databázové dotazy, informace o tom, jestli je uživatel přihlášený nebo třeba obsah košíku na

e-shopu. “Model neví, odkud data v parametrech přišla a ani jak budou výstupní data zformátována a vypsána.” (Čápka) Jeho úkolem je přijmout vstupní data, zpracovat je a poslat je zpět.

Důležitou vlastností modelu je validace přijatých dat. Samozřejmě může programátor provádět validaci ještě na straně uživatele například pomocí JavaScriptu, ale to spíše pro zvýšení komfortu.

Do rozhraní může vstupovat v libovolnou dobu jakýkoliv uživatel, tudíž by měl být model plně konzistentní a to ve všech svých stavech. Nemělo by se stát, že ho nějaký požadavek dostane do nepovolaného stavu. Proto je třeba provádět validaci právě v modelu a snažit se, aby model byl dostatečně odolný a poskytoval ven data, která by nenarušila jeho integritu. (Tichý, 2007)

To, že model poskytuje funkce na čtení a zapisování údajů neznamená, že model se rovná databázi. “Ve většině MVC architektur ukládá model údaje v relační databázi, jako je například MySQL, PostgreSQL nebo Oracle.” (Böhmer, 2010, s. 399) Model však není omezen jen na databáze, může také používat různá úložiště údajů jako jsou textový soubor, RSS nebo jiné webové služby.

View (pohled)

Pohled vyobrazuje výstup uživateli. Stejně jako model, tak ani pohled nepotřebuje vědět, odkud data přišla. Stará se jen o jejich zobrazení uživateli. Důležité ale je, aby poznal jejich typ a formát, kterým může být například objekt nebo pole. V případě Zend frameworku se jedná o phtml šablonu, kde jsou pomocí značkovacího programovacího jazyka vloženy proměnné. Logika v pohledu by měla být, jak píše ve své dokumentaci Nette Foundation (2008 - 2017a)**,** omezena pouze na iterace (cykly) a podmínky.

Pohledů lze mít v aplikaci mnoho a jeden pohled může být společný všem uživatelům (*ProfilView*), záleží jaká data jsou do něj zaslána. Šablony mohou být vkládány do sebe, abychom zamezili opakování, a tím případných chyb. (*Menu*, *Footer*)

Controller (kontroler)

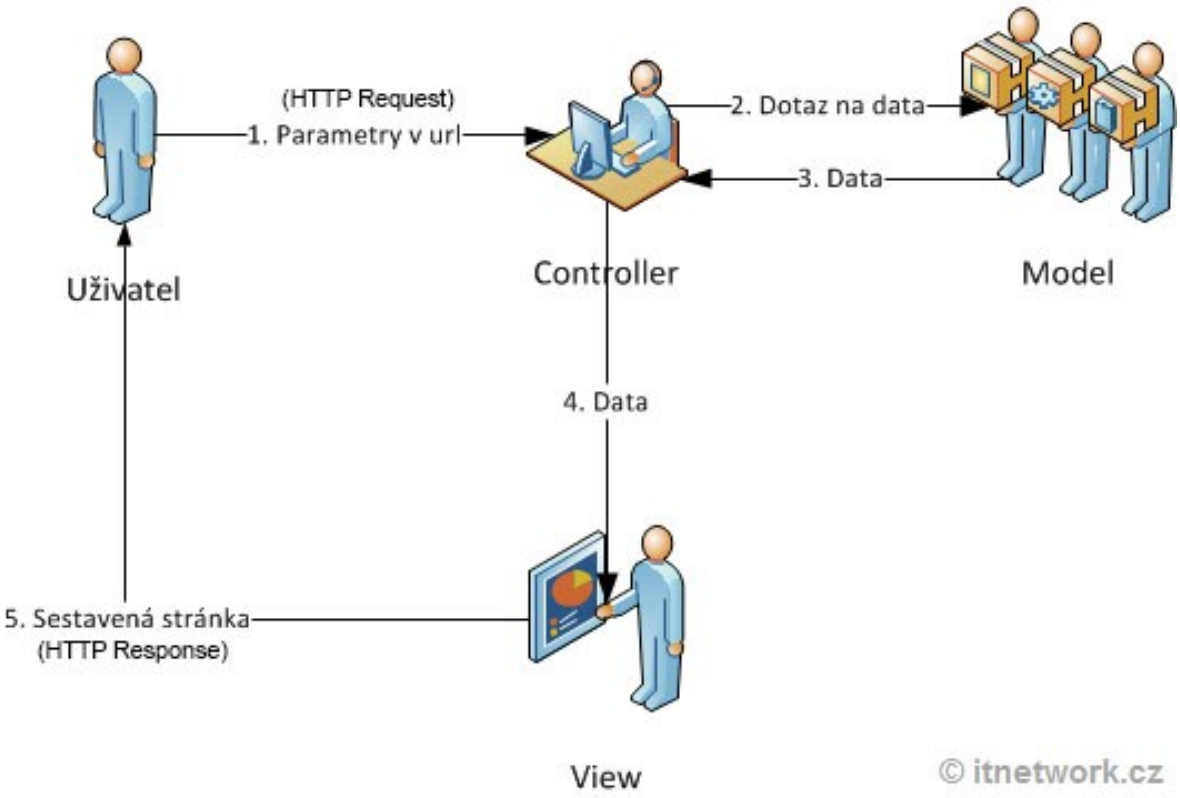
K tomu, aby celá struktura fungovala a vzájemně všechno fungovalo je zapotřebí kontroleru. Kontroler se stará o funkčnost celého návrhového vzoru MVC a je jakýmsi řadičem.

Teoreticky by programátor mohl vytvořit aplikaci bez přítomnosti modelů a pohledů. Řadič by pak tvořil jak logickou tak i zobrazovací část. V praxi bychom se tomu ale měli vyhýbat. Kontroler komunikuje s uživatelem, modelem a pohledem. Od uživatele dostává požadavky, které patřičně vyhodnotí a rozhodne, který model je zpracuje. Z modelu se vrátí odpověď, která se vygeneruje do příslušného pohledu.

Cyklus v MVC

Obrázek zobrazuje HTTP požadavek na server a průběhu jeho zpracování. "Kromě už zmiňovaných tří komponent (Model, View a Controller) je obrázek doplněn o HTTP Request (požadavek), HTTP response (odpověď) a propojení mezi jednotlivými elementy." (Böhmer, 2010, s. 397)

Obrázek 1: Cyklus MVC [Zdroj: (Čápek, 2017)]



1. Každý požadavek začíná tím, že se odešle HTTP požadavek na server. Uživatel zadá do prohlížeče adresu webové stránky s parametry a podle parametrů se pozná jaký kontroler má být volán. Pokud uživatel chce zobrazit stránku s článkem, volá se například kontroler *ClankyController*.

2. Z požadavku dále musí kontroler zjistit, jaký model musí být v případě potřeby osloven, a požaduje od něho údaje.
3. V některých případech předá kontroler modelu údaje získané od uživatele a požaduje jejich uložení. “V tomto případě obdrží kontroler od modelu odpověď o úspěchu, respektive neúspěchu, vykonané operace.
4. Kontroler zjistí pohled potřebný na výpis údajů a předá mu údaje, které mají být vypsány.
5. Na závěr vytvoří pohled výpis údajů (například ve formě HTML stránky) a řadič ho pošle zpět do prohlížeče uživatele ve formě HTTP odpovědi.

V závislosti na implementaci se může stát, že se pohled dožaduje údajů od modelu, respektive model informuje pohled o změně údajů. To vytváří mezi vývojáři dvě skupiny. Někteří jsou toho názoru, že pohled nesmí nikdy přímo přistupovat k modelu ani jen číst. Jiní to berou volněji a jsou toho názoru, že pohled může čtením přistupovat k modelu. Toto může být například tehdy, jestliže řadič předá pohledu primární klíč knihy, která má být vypsána, a pohled tehdy zavolá metodu modelu, která vrátí všechny informace o knize. V každém případě však pohled nikdy nesmí přistupovat k modelu za účelem zapisování údajů.” (Böhmer, 2010, s. 398)

2.3.2 MVP

Dalším návrhovým vzorem je MVP (Model-View-Presenter), který využívá Nette framework. Architektura MVP je jako MVC rozdělena do třech komponent, s tím rozdílem, že místo kontroleru zaujal místo presenter. Na první pohled by se mohlo zdát, že se změnilo jen poslední písmeno, ale presenter má jiné vlastnosti, než kontroler. Také pohled se může lišit u mnoha frameworků na principu MVP.

Model

Model je komponent, který se od návrhové vzoru MVC nijak neliší. Jeho úkole je stejný - logická část, která se stará o přístup k datům a celkovou manipulaci s nimi.

View (pohled)

Pohled v návrhovém vzoru MVC neměl žádnou vazbu na uživatelské akce (například reakce na kliknutí), ale to se právě v MVP mění. Kontroler jako takový již nezpracovává

uživatelský požadavek, ale dělá to právě pohled. Pohled čeká na uživatelský vstup, který může být třeba na webovém tlačítku, a poté zavolá příslušnou metodu v presenteru. Neznamená to ale, že v pohledu vzniká jiná logika, než výpis parametrů, iterace (cykly) a podmínky. Pohled má pouze zaznamenat uživatelský požadavek a postarat se o zobrazení výstupu. V Nette frameworku je pohled vykreslen pomocí tzv. Latte šablony[3].

Při vývoji webových aplikací může nastat situace, kdy je zapotřebí složitější logiky v pohledu. Spravovat tuto logiku může být velice obtížné, stejně tak jako ji testovat. Proto Fowler (2006) rozdělil dále návrhový vzor MVP na dvě hlavní části a to podle míry logické zodpovědnosti pohledu.

1. Supervising Controller
2. Passive View

Ve variantě Supervising Controller je pohledu "svěřena zodpovědnost za zobrazení dat z modelu, nejčastěji pomocí vzoru Observer[4] nebo v modernějších technologiích pomocí data bindingu[5]. Jakákoliv složitější prezentační logika je potom přesunuta do Presenteru, který má dovoleno s View libovolně manipulovat." (Bernard, 2009)

Passive View má jediný, avšak podstatný rozdíl od varianty Supervising Controller. Zmizela vazba pohledu na model. "Presenter má nyní daleko větší zodpovědnost, protože kromě prezentační logiky musí i správně synchronizovat View s Modelem. Primární motivací pro tento vzor jsou automatické testy, které se pro View píší velmi těžko. Veškerá logika se proto přesouvá do Presenteru, který lze pokrýt testy výrazně snadněji. " (Bernard, 2009)

[3] "Latte je šablonovací systém pro PHP, který vám ušetří a zpříjemní vám práci a zabezpečí výstup před zranitelnostmi jako je XSS" (Nette Foundation, 2008-2017) Dostupné z: https://latte.nette.org

[4] "Návrhový vzor Observer umožňuje objektu spravovat řadu pozorovatelů, kteří reagují na změnu jeho stavu voláním svých metod" (Čápka) Dostupné z: http://www.itnetwork.cz/navrhove-vzory/observer-pozorovatel-navrhovy-vzor

[5] Data binding je proces, který reaguje a mění obsah, podle uživatelské akce. Příkladem může být změna *checkboxu*. (Rouse, 2005) Dostupné z: http://searchmicroservices.techtarget.com/definition/data-binding

Presenter

Presenter pracuje podobně jako kontroler s tím rozdílem, že manipuluje s požadavky, které mu pošle pohled. Mění se jeho přístup ke vstupním datům, ale jeho úkol prostředníka mezi modelem a pohledem je stejný. Pohled po požadavku uživatele volá určitou metodu presenteru, která dále může pracovat s modelem. Logika presenteru by se měla vymezit na úplnou změnu pohledu, změnu stavu aktuálního pohledu nebo příkazu pro model.

2.4 Nette Framework

Nette Framework je český projekt, který byl v roce 2008 publikován a do roku 2014 vyvíjen autorem Davidem Grudlem. (Hassman, 2014) Nyní se o další rozvoj stará Nette Foundation a původní autor pouze dohlíží na jeho pokračování. Jelikož je to český projekt, má v České republice velikou popularitu. Na jednu stranu vzniká rozsáhlá komunita s kvalitní českou dokumentací a se zdroji, na stranu druhou to brání v rozvoji frameworku na světovou úroveň. Výhodou vzniku projektu v Čechách může být i akce jménem Poslední sobota, kde se pravidelně každý měsíc, setkávají fanoušci Nette frameworku a vzájemně si vyměňují své zkušenosti. O tom, že Nette framework nezůstává pozadu a neustále se vyvíjí, vypovídá i to, že se umístil na třetím místě v anketě "Nejlepší PHP framework roku 2015". (SitePoint, 2015)

Aktuální stabilní verze Nette frameworku je označená číslem 2.4 a vyžaduje PHP verzi alespoň 5.6 nebo vyšší. Tato verze je kompatibilní s s PHP 7.1, Nette 2.1 a 2.2 s PHP 7.0. (Nette Foundation, 2017b)

2.4.1 Licence

Nette framework je open source projekt, což pro vývojáře znamená, že mohou volně pracovat s jeho otevřeným zdrojovým kódem. Mohou ho přepisovat, opravovat, kopírovat nebo vyvářet nové návrhy pro zlepšení. V případě Nette frameworku si ale musí vybrat jednu

ze dvou licencí, kterou je třeba respektovat. Na výběr mají buď new BSD[6] licenci nebo GNU GPL[7] ve verzi 2 nebo 3.

New BSD licenci využije většina vývojářů, jelikož je snadná pro porozumění a umožňuje využívat program prakticky bez omezení. Podmínkou pro tuto licenci je uvádět v programu jména autorů a vzdát se odpovědnosti. Využívá se i v komerčních projektech, kdy se nemusí zveřejňovat ani zdrojové kódy aplikace.

Druhá licence GNU, respektive GPL, je velice populární licence, která říká, že odvozené dílo patří pod stejnou licenci jako dílo původní.

2.5 Zend Framework

V říjnu 2005 firma Zend Technologies Inc. oznámila, že pracuje na novém PHP frameworku. Začátkem roku 2006 byla zveřejněna první verze Zend frameworku, Pre Alpha Version 0.1.1, která sice nebyla označena jako produkční verze, ale díky komponentům, které již tehdy nabízela se s pomocí několika vývojářů razantně začala rozvíjet. Polovina roku 2007 byla pro Zend framework zásadní, jelikož Zend vývojáři vydali první produkční verzi 1.0. Pokračovalo se ve vývoji a po několika letech přišla verze 2.0, která se chlubí 15 milióny stažení. Aktuální verze Zend framework 3 si bere z obou předchozích verzí to nejlepší a celkové číslo stáhnutí již překročilo 116 miliónů. Verze vyžaduje alespoň PHP 5.6 nebo vyšší. Vývojáři kladli velký důraz na problematiku zpětné kompatibility, takže je migrace na novější verzi zcela bezproblémová. (Böhmer, 2010, s. 23)

Hlavním přispěvatelem frameworku je společnost Rogue Wave, která se stará o to, aby byl Zend framework nadále zlepšován a rozšiřován. Jsou zde ale i společnosti jako Google, Microsoft nebo StrikeIron, které jsou partnery projektu a na přání vývojářů se snaží implementovat nové služby a funkce.

---

[6] "BSD (Berkeley Software Distribution) je svobodná licence, která ale může být použita i ke komerčním účelům." (ITSlovnik) Dostupné z: https://it-slovnik.cz/pojem/bsd-licence

[7] Původně napsaná licence pro projekt GNU. GPL je všeobecně veřejná licence a "software s touto licencí je k dispozici se zdrojovými kódy a zdarma. Zdrojové kódy s touto licencí mohou být svobodně upravovány a používány, šířeny však musí být opět pod GPL." (ITSlovnik) Dostupné z: https://it-slovnik.cz/pojem/gpl

Čísla stáhnutí, která jsou vypsána v prvním odstavci o Zend frameworku značí, že framework musí mít velkou základnu uživatelů. Velká komunita má výhody v tom, že na internetu existuje několik diskuzních fór či skupin, kde si vývojáři píšou ohledně frameworku. Je velká šance, že pokud vývojář narazí na problém, na internetu najde odpověď. Všechny nové komponenty, které vznikají, musí nejdříve mít svojí příručku a až poté se publikují. Příručky mohou být přepsány do několika jazyků. (Zend, 2006-2017a)

2.5.1 Licence

Zend je stejně jako Nette framework, distribuován pod new BSD licencí, která opět umožňuje vývojáři využívat program prakticky bez omezení.

Pokud bude vývojář chtít přispět do projektu Zend framework 1 verze svým kódem, musí podepsat CLA[8], čímž potvrdí, že jeho přispívající část neporušuje práva třetích osob. U nejnovější verze Zend frameworku 2, podepisování licence již není požadavkem. (Block, 2014)

2.6 Symfony Framework

Dalším velice známým webovým frameworkem je Symfony, který byl poprvé zveřejněn roku 2005 a to ve verzi 1.0. Myšlenka vzniku frameworku patří webovým designérům ze společnosti SensioLabs, ale za vznikem stojí jejich webový vývojáři. Framework je open source projekt vyvíjený již pět let nejenom zmíněnými lidmi ze SensioLabs, ale tisícem dalších vývojářů po celé zemi. Po několika letech se Symfony dostalo na číslo 3.2.4, které značí aktuální verzi frameworku.

Vývojáři ze SensioLabs si zakládají na pravidelném vydávání verzí frameworku, ať už nových nebo revizních. Vývojáři mohou počítat s tím, že pokud se v jejich verzi najde chyba, v příští verzi bude opravena. V hlavních verzích se devět měsíců starají o chyby a až čtrnáct měsíců o bezpečnostní chyby. K tomu každé dva roky vyjdou takzvané LTS[9] verze, kde se

[8] Contributor License Agreement licenci využívají projekty, kteří si chtějí být jistí, že jejich přispěvatelé neporušují práv třetích osob a že přispívající část vlastní oni. (Morrison, 2012) Dostupné z: https://www.clahub.com/pages/why_cla

[9] Long Term Support Version jsou označené takové verze, které mají delší podporu, než je standardní doba. (Symfony, 2017b) Dostupné z: http://symfony.com/doc/current/contributing/community/releases.html

o chyby starají tři roky a o bezpečnostní chyby čtyři roky. Zpětná kompatibilita je samozřejmostí.

Bezpečností strategie frameworku Symfony nabádá vývojáře, kteří narazí na chybu v programu, aby jí nezveřejňovali, ale poslali na příslušný email, kde se o ni postará tým, který má na starosti bezpečnost. Oni vytvoří sdílenou složku s příslušným oprávněním a pracují na tom tak, aby mohla být chyba opravena v další verzi.

Symfony slouží také jako velká platforma pro vytváření projektů, které si berou některé komponenty právě z frameworku. Drupal, phpBB, Laravel, Joomla!, Composer nebo Doctrine jsou všechno projekty, které vznikly na bázi Symfony. Vývojář, který se nějaký čas pohybuje ve webovém odvětví musí určitě jeden z nich poznat. (Symfony, 2017a)

2.6.1 Licence

Na rozdíl od přecházejících frameworků je Symfony distribuován pod licencí MIT[10]. Tato licence může být využívaná i u programů, u kterých je uzavřený kód. U open source projektů je plná kompatibilita s GNU GPL licencí, která dovoluje kombinaci s MIT. Pro vývojáře to znamená jen to, že text licence MIT musí být uveřejněn spolu se zveřejněným dílem.

[10] Massachusetts Institute of Technology je licence, která "umožňuje použití v komerčních projektech bez nutnosti zveřejňování zdrojových kódů, jen vyžaduje přidávání textu licence MIT do projektu." (Mickey, 2011) Dostupné z: http://mickey.pise.cz/117-licence-gnu-gpl-gnu-lgpl-mit.html

3 Metodika porovnávání frameworků

3.1 Základní aspekty

Vývojář, který nemá mnoho zkušeností s programováním ve webovém frameworku, musí na začátcích narazit na jeho základní aspekty. Bez předchozích zkušeností se tolik nebude zabývat programováním samotným, jakožto výhodami a nevýhodami samotného frameworku, který si vybral. Může to být například komunita, která mu na začátku jeho učení rychle odpoví na problém, který napsal na diskuzní fórum. Framework si tím u něj získá plusové body za komunitu. Následující kapitoly budou zaměřeny právě na tyto základní aspekty.

Zaměřuji se na aspekty, které podle vlastních zkušeností považuji za základní a důležité. U každého frameworku by se dalo vymyslet ještě několik dalších aspektů, které by se daly porovnávat. Cílem ovšem je pomoct vývojářům vybrat si přívětivý webový framework, který jim ulehčí a zefektivní práci na jejich webové aplikaci. K tomu není potřeba do detailu zkoumat každou komponentu a každou funkci frameworků.

Všechny aspekty, které se budou porovnávat, jsem na základě vlastních zkušeností seřadil podle důležitosti při vývoji webové aplikace. První dva aspekty budou testovány a hodnoceny zcela objektivně - například měření rychlosti aplikace. Ostatní aspekty budou hodnoceny zcela subjektivním názorem - například velikost komunity a její aktivita.

Hodnocené vlastnosti frameworku jsou tyto:

- Bezpečnost
- Rychlost
- Komunita
- Dokumentace
- Vývoj

Hodnocení webových frameworků bude prováděno formou slovního hodnocení, i když v prvních dvou případech půjde o objektivní hodnocení. Podle výsledků bude evidentní, jaký webový framework by mohl být nejrychlejší a jaký nejpomalejší. Slovní hodnocení bude nejvhodnější formou, jelikož umožňuje popsat webové frameworky výstižněji

a objektivněji, než například hodnocení pomocí známkování. Výsledkem bude výpovědní hodnota, která bude říkat, jak webový framework obstál v daném aspektu.

V závěru této kapitoly bude vytvořen souhrn ze všech hodnocených aspektů a navzájem budou porovnány. Výsledné hodnocení bude jako v případě aspektů, hodnoceno slovní formou. Cílem této části práce je sestavit souhrn, který bude obsahovat silné a slabé stránky daného webového frameworku. Ty budou ukazatelem toho, pro které vývojáře je daný framework vhodnější a pro které naopak ne.

Jelikož všechny vytvořené aplikace budou testované v jednotném prostředí (localhostu), aby nedošlo ke zkreslení při měření, doplním informace o konfiguraci počítače, verze technologií a verze frameworků.

Konfigurace počítače:

- Počítač, OS: MacBook Pro Retina 2014, macOS Sierra
- Procesor: 2,6 Ghz Intel Core i5
- Paměť: 8 GB 1600 Mhz DDR3
- Grafika: Intel Iris 1536 MB

Verze technologií:

- Apache: 2.2.31
- PHP: 7.0.4
- MySQL: 5.6.33
- Safari: 10.0.3

Verze frameworků:

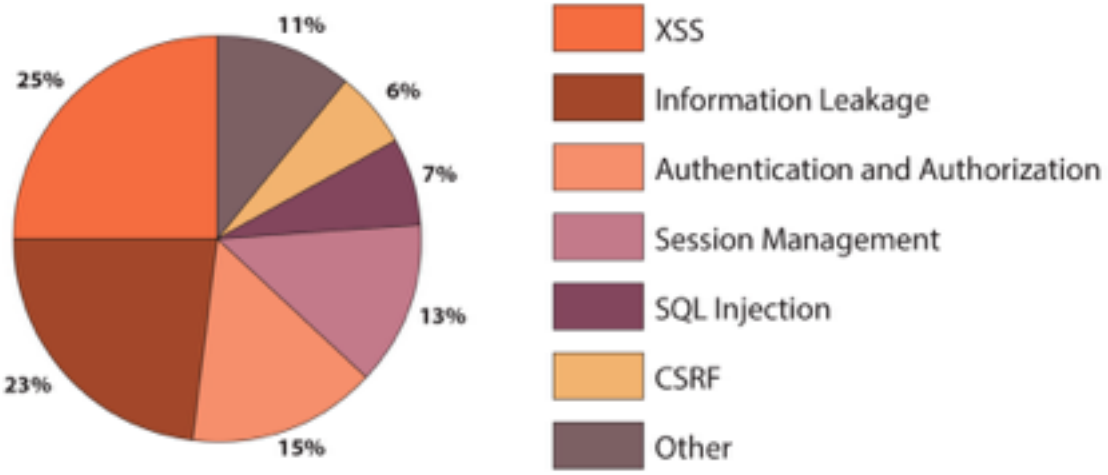
- Nette: 2.4
- Zend: 3.0
- Symfony: 3.2.4

3.2 Bezpečnost

Bezpečnost bude v modelových aplikacích testována webovými útoky - XSS, CSRF a Session krádež. Ty se podle Cenzic (2014) nacházejí mezi šesti nepoužívanějšími útoky.

Vývojář nemusí vědět nic o těchto bezpečnostních rizikách, ale framework se o všechno postará na pozadí. Vývojáři samotného frameworku jsou zkušení programátoři, kteří o bezpečnosti vědí hodně a tak do jádra projektu napíšou několik řádků, které se o všechno postarají. V celkovém důsledku je o bezpečnostní rizika postaráno automaticky nebo stačí přidat jeden řádek s funkcí, která se o to postará. Jak prosté. Je tomu tak, ale u všech frameworků?

Obrázek 2: Nejpoužívanější útoky roku 2013 [Zdroj: (Cenzic, 2014)]



Vybrané typy bezpečnostních útoků jsou jedny z nejpoužívanějších a budou testovány v té nejzákladnější podobě. Jistě by se našel útok, který by vybraný framework nezvládl a našla by se v něm bezpečnostní chyba. To však není cílem testu bezpečnosti. Vývojáři webových frameworků většinou prohlašují, že v jejich prostředích jsou základní bezpečnostní útoky eliminovány. Testováním vybraných typů útoků se prověří, jestli tomu tak opravdu je. Bylo by přeci velice nevhodné, kdyby webový framework podlehl základnímu bezpečnostnímu útoku. Pro další analýzu bezpečnosti může vývojář využít dostupné testy na portálech jednotlivých projektů.

K online ověření bezpečnosti webových aplikací jsou na internetu k dispozici desítky nástrojů. Od nástrojů, které jsou zdarma a poskytují základní výpis informací o zranitelnosti, až po ty placené, které vývojáři pečlivě prověří webovou aplikaci a výsledný výpis je brán do detailu. Důvod proč se při testování bezpečnosti nevyužije online nástroj je ten, že nástroj sice vypíše podrobný seznam nalezených chyb, ale je jen na vývojáři, aby je odstranil.

V modelové aplikaci se bude testovat jak bezpečnostní útok, tak i obrana proti němu a tím v závěru vznikne objektivnější vyhodnocení.

U každého z útoků se pokusím v jednom odstavci vysvětlit princip, kterým je útok prováděn a čeho všeho může být útokem docíleno. V druhém odstavci uvedu přesnou techniku, kterou se budu snažit ověřit bezpečnost webových frameworků.

3.2.1 Cross-Site Scripting (XSS)

Jedním z bezpečnostních rizik je XSS, který nastane tehdy, když útočník podstrčí do stránky svůj vlastní kód a tím může stránku zcela změnit nebo dokonce získat citlivé údaje. Především díky neošetřeným vstupům je možné provést XSS útok. Základním principem, jak se proti tomuto napadení bránit, je převádět každý vstupní řetězec na entitní vyjádření (tzv. escapování[11]), aby byl korektně vyjádřen výstup. Pokud bude řetězec obsahovat například < a >, musí být takové znaky převedeny na < a >. Takových případů, kdy se pracuje se vstupem od uživatele je mnoho a stačí jen jedinkrát opomenout a celá webová aplikace může být ohrožena.

V modelové aplikaci tento útok provedu tak, že do parametrů, které se předávají pomocí metody GET, vložím script *<script>alert('JavaScript XSS útok!');</script>*. Tento útok by měl zapříčinit pouze to, že v prohlížeči vyskočí okénko s textem: "JavaScript XSS útok!".

3.2.2 Cross-Site Scripting Request Forgery (CSRF)

Tento druh útoku souvisí, stejně jako předchozí s tím, že se útočník snaží podvrhnout svůj vlastní kód. Aby se mohl vývojář vyhnout CSRF útoku je bezpodmínečně nutná ochrana proti XSS. Kdyby tomu tak nebylo, útočník by mohl snadno přečíst pomocné kódy, nebo přečíst kontrolní token. Princip je takový, že útočník vykonává útok na webovou stránku přes právě přihlášeného uživatele. Takto může manipulovat s funkcemi, ke kterým má uživatel povolení, bez jeho vědomí. Bránit se lze tím, že se bude generovat autentizační

[11] "Escapování je převod znaků majících v daném kontextu speciální význam na jiné odpovídající sekvence. Příklad: do řetězce ohraničeného uvozovkami chceme zapsat uvozovky. Jelikož uvozovky mají v kontextu řetězce speciální význam a jejich prosté zapsání by bylo chápáno jako ukončení řetězce, je potřeba je zapsat jinou odpovídající sekvencí. Jakou přesně určují pravidla kontextu." (Grudl, 2009) Dostupné z: https://phpfashion.com/escapovani-definitivni-prirucka

token a při operaci, ověřovat. Představit si lze i řešení pomocí CAPTCHA[12] nebo SMS, kdy nelze operace uživatele nijak podvrhnout. (Ferschmann, 2008)

Jako u předchozího útoku využiji metody GET, která je u toho typu útoku tou jednoduší variantou, ale pro hodnocení dostačující. V případě POST je zapotřebí, aby měla oběť útoku zapnutý JavaScript a postarat se o to, aby se po načtení stránky potvrdil formulář. Formulář bude prvek v modelové aplikaci, který využiji při pokusu o útok. Bude obsahovat textové vstupní pole, které pojmenuji email, predmet a obsah. Útočníkova webová stránka bude obsahovat obrázek, ale jeho URL adresa bude odkazovat na vyplněný formulář, na který útočím. Obrázek, který bude obsahovat právě tento odkaz, zapíšu jako *<img src="bc/CSRF/email=jan.xxx@gmail.com&predmet=utok&obsah=bezpecnost">*.
Předpokladem takového útoku je, že oběť vstupující na útočníkovy stránky je právě přihlášená do webové aplikace, kterou chce útočník napadnout.

3.2.3 Session krádež

Každá webová stránka, která má nějaké přihlášení, pravděpodobně využívá session nebo cookie, a to k tomu, aby dokázala určit, jaký uživatel je přihlášen a co se má danému uživateli zobrazit. Při přihlášení do aplikace se uživateli vytvoří session identifikátor, který útočník může podvrhnout a získat tím přístup do aplikace bez toho aby znal heslo uživatele. Tím může v aplikaci provádět cokoliv, aniž by si toho samotný uživatel všiml. K tomu, aby se vývojář tomuto problému vyhnul, je třeba správné konfigurace serveru a PHP, kde se musí řešit třeba náhodné generování právě zmiňovaného session identifikátoru.

Session identifikátor se budu snažit odcizit pomocí Javascriptu. Jestliže bude pokus úspěšný, získám tím přístup do webové aplikace, aniž bych musel znát přihlašovací údaje. Vytvořím webovou aplikaci, do které budu mít přístup jen s předem vytvořenými přihlašovacími údaji. Spolu s tím vytvořím i útočníkovu webovou stránku, která bude obsahovat JavaScript příkaz *document.cookie*. Pokud se tedy přihlásím do webové aplikace a poté navštívím útočníkovu

[12] CAPTCHA je program, který ochraňuje webové stránky proti webovým robotům (program, který automaticky spouští scripty) tím, že generuje testy, které roboti nedokážou splnit, ale lidem nedělají problémy. Příkladem může být přepsat zakřivený text. (Captcha, 2000-2010) Dostupné z: http://www.captcha.net

stránku, měl bych získat session identifikátor mého přihlášení. V druhém webovém prohlížeči použiji odcizený session identifikátor a tím se přihlásím.

3.3 Rychlost

V dnešní moderní době je rychlost jednou z nejdůležitějších vlastností aplikace, jelikož každý se snaží hospodařit s časem. Byť by to bylo jen o pár sekund navíc. Pár sekund navíc a uživatel prohlížející webové stránky hned může ztratit zájem. Weboví vývojáři se proto musí přizpůsobit a snažit se vytvářet aplikace, které s rychlostí nebudou mít problém. Záleží na více aspektech, které mohou ovlivnit rychlost načítání webové stránky - výkonnost hardwaru zařízení, prohlížeč webových stránek, systematičnost a kvalita napsaného zdrojového kódu. Pro webového vývojáře je důležitá optimalizace zdrojového kódu, který pracuje na daném frameworku.

Čím složitější projekt je, tím větší má režii. Webová aplikace může například stahovat několik externích souborů najednou nebo načítat objemné obrázky. To může zapříčinit to, že načítání bude velice zdlouhavé. Pokud bude ale jen jednoduchá webová stránka bez složitějších elementů, mohu měřit opravdovou rychlost, která se odráží od kvality zdrojového kódu webového frameworku.

Pro testování rychlosti vytvořím PHP soubor, který bude obsahovat tři základní dotazy na databázi - INSERT, UPDATE a DELETE. Každý příkaz budu testovat zvlášť a několikrát, abych zaručil přesnější výsledky. Příkazy budou pracovat s tabulkou, ve které bude 100 řádků, aby při měření došlo k nějaké diferenciaci. Vycházím totiž z předpokladu, že vkládat, upravovat nebo mazat jeden řádek v databázi by nepřineslo v měření žádné značné rozdíly. Měření rychlosti budu provádět 20x za sebou (opětovné načtení stránky). Z naměřených hodnot udělám průměr, který bude brán jako konečný výsledek. Všechny tři frameworky budu měřit stejným způsobem a popořadě.

Nástroj, který při měření použiji je inspektor webu v prohlížeči Safari. Ten nabízí takové funkce, které mi pomohou změřit rychlost dotazů na databázi v milisekundách. Funkce časová osa, dokáže rozdělit měření načítání webové stránky na síťové požadavky, CSS vykreslování a JavaScript události. To mi zajistí ještě přesnější měření.

3.4 Komunita

Komunita je jednou z nejdůležitějších aspektů při výběru frameworku, ať už je vývojář začátečník nebo pokročilý. Pokud webový vývojář tápe v začátcích a dokumentace není natolik dobře sepsaná, aby mu pomohla, využije pomoci u komunity frameworku. Podle mých zkušeností je aktivní komunita podstatnější, než dokumentace. Do komunity zahrnuji jakékoliv informační zdroje, které nejsou brány jako dokumentace - diskuzní fóra, skupiny, blogy, články nebo společenské akce spojené s daným frameworkem. Při hodnocení tohoto aspektu budu zkoumat právě tyto oblasti. Komunita je to, co vývojáři dává stéle nové a nové informace o frameworku.

U komunity budu uvádět čísla přibližného počtu výsledků při vyhledávání ve webovém vyhledávači Google. Při vyhledávání dám hledané výrazy do uvozovek, a tím se budou vyhledávat jen ty webové stránky, které obsahují pouze toto sousloví. Hledaná sousloví budou tato - Nette framework, Zend framework a Symfony framework. Výsledná čísla budu uvádět pro každý framework zvlášť.

3.5 Dokumentace

Kdyby neznámý vývojář vymyslel svůj vlastní framework, publikoval ho a nepřiložil k němu dokumentaci, jeho projekt by se neujal. Nikdo kromě autora by danému frameworku nerozuměl v takové míře, aby s ním začal pracovat. Dokumentace je nedílnou součástí frameworku. Každý, kdo si stáhne poprvé jakýkoliv framework, musí nahlédnout do dokumentace. Tam by mělo být uvedeno, jak daný framework pracuje a jak plně využívat jeho potenciálu. Postupem času se ale dokumentace pro pokročilejšího vývojáře stává méně důležitou. Nápovědy k věcem, které v dokumentaci dříve vyhledával, jsou pro něj nyní automatické. Samozřejmě se může stát, že i pokročilý vývojář nahlédne do dokumentace, jelikož něco zapomněl. Pokud si ale vývojář osvojí ve větší míře framework, hledá přednosti v aspektech, které jsem popisoval výše - bezpečnost, rychlost a komunita. S tím, aby se framework dostal do světa a dále se šířil, bude určitě souviset i to, jestli bude dokumentace sepsána ve více jazykových mutacích.

Dokumentaci bych rozdělil do třech kategorií:

- Příručka
- Vzorová aplikace
- API reference

Příručka

Nejdůležitější částí dokumentace je příručka. V příručce může vývojář najít základní informace o tom, jak funguje framework a všechny jeho komponenty. Hodnotit na uživatelských příručkách budu jak samotnou informační hodnotu, tak i přítomnost obrázků a ukázek kódů. To by nemělo v dobré příručce chybět, jelikož vývojář tím získá lepší přehled v dané problematice. Výhoda ukázek kódu je, že programátor nemusí zdlouhavě přepisovat kód, ale může ho třeba jen zkopírovat do své aplikace a vyzkoušet ho v praxi.

První aplikace

Začít psát aplikaci ve frameworku, ve které vývojář teprve začíná, i když si mohl přečíst dokumentaci, může být někdy docela složité. Proto by neměla chybět série článků, které na sebe navazují a pomáhají vývojáři ke spuštění první aplikace. Cílem není hned vytvořit webové stránky se složitými funkcemi, ale dát programátorovi krok po kroku šanci pochopit jednotlivé funkce frameworku. Přítomnost odkazu na stažení celé aplikace považuji za dobrý nápad a budu na to nahlížet.

API reference

Přítomnost této části v dokumentaci je nezbytná. Ať už bude vývojář pracovat s daným frameworkem jakkoliv dlouho, vždy se může najít příležitost k tomu, aby se podíval na API dokumentaci. Jde o kolekci všech tříd, funkcí, metod, procedur a jejich podrobný popis, kde se ve struktuře frameworku nachází, do jakého jmenného prostoru patří nebo se kterými parametry pracují.

3.6 Vývoj

Pravděpodobně žádný z vývojářů si nevybere takový framework, který se nebude dále vyvíjet a zlepšovat. Vývojáři většinou chtějí, aby jejich verze byly stabilní a bezchybné.

Sémantické verzování *x.y.z* se dělí, jak píše Preston-Werner (2016) do třech hlavních kategorií:

- Verze major
- Verze minor
- Verze patch

Verze major přinášejí zásadní změny. Něco, co v předchozí verzi nebylo a třeba i scházelo. Ve verzích major se manipuluje s celým zdrojovým kódem a přihlíží se k novým technologiím, proto vzniká určité riziko, že dojde k problému tam, kde dříve nebyl. Dochází také k celkové změně API, takže major verze není zpětně kompatibilní s předchozí verzí. (Preston-Werner, 2016) Pro mnoho vývojářů ale převažují výhody major verzí, takže spokojeně migrují všechny své aplikace na nejnovější verzi frameworku.

I přes mnoho testů, které udělají vývojáři frameworku, se může stát, že se vyskytne v praxi po oficiálním vydání nějaký problém. K tomu slouží verze minor, které opravují bezpečností chyby a chyby obecně. Mohou přidat několik funkcí navíc, ale jejich vydání by nemělo změnit API a měla by se zachovat zpětná kompatibilita.

Jestliže se nepřidávají žádné funkcionality a je potřeba pouze opravit chyby, využije se patch verze. Ta musí být zpětně kompatibilní a nesmí změnit API. (Preston-Werner, 2016)

Budu hodnotit pravidelnost vydávání jednotlivých verzí frameworků. Zaměřím se nejenom na verze major, které v mnoha případech vychází po uplynutí mnoha let, ale i na minor verze, které udrží framework stále aktuální.

4 Modelová aplikace

4.1 Nette Framework

4.1.1 Bezpečnost

Před zranitelností XSS se vývojáři v Nette frameworku nemusí bát. "Nette Framework přichází s revoluční technologií Context-Aware Escaping, která vás provždy zbaví rizika Cross-Site Scriptingu. Všechny výstupy totiž ošetřuje automaticky a tak se nemůže stát, že by kodér na něco zapomněl." (Nette Foundation, 2008-2017a) V praxi to znamená, že vývojáři stačí proměnné vložit do složených závorek {*$promena}* a výstup je ošetřen. Pokud by chtěl, aby byl výstup bez escapování (např. název článku s HTML značky), přidal by k proměnné filtr *noescape*. Context-Aware Escaping usnadňuje práci zároveň tak, že vývojář nemusí přemýšlet jaké escapování použije, jelikož technologie to pozná sama. Escapování může totiž probíhat například v CSS, PHP nebo JavaScriptu.

K tomu, aby webový vývojář v Nette frameworku zabránil Cross-Site Request Forgery (CSRF) útoku, stačí aby k vytvářenému formuláři vložil řádek s *$form->addProtection();*. Vytvoří tím vygenerovaný token, který se bude při odeslání formuláře ověřovat. Token má určitou časovou platnost a při jeho vypršení je uživatel požádán, aby znovu odeslal formulář. První parametr funkce je text chybové hlášky, druhý je nastavení času platnosti tokenu.

Získat session identifikátor se mi při testování bezpečnosti Nette frameworku nepodařilo. Framework ve svém zdrojovém kódu využívá funkci *init_set*, která umožňuje přímo konfigurovat PHP direktivy. Tato funkce musí být ovšem povolená, jinak by při webhostingu mohl vzniknout problém. Vývojář se poté musí domluvit se správcem serveru, aby mu server nakonfiguroval. Nette a změna nastavení PHP zajistí, aby "Session ID přenášel pouze v cookie, znepřístupnil jej JavaScriptu a případné identifikátory v URL ignoroval. Navíc v kritických chvílích, jako je třeba přihlášení uživatele, vygeneruje Session ID nové." (Nette Foundation, 2008-2017a)

Verdikt

Ani jedno bezpečnostní riziko, které jsem ve webové aplikaci testoval, neprošlo. Je vidět, že se Nette bezpečností určitě zabývá a snaží se, aby webové aplikace pod frameworkem byly zabezpečené. Z bezpečnostního hlediska Nette framework obstál na výbornou.

4.1.2 Rychlost

Obrázek 3: Výsledné hodnoty rychlosti pro Nette [Zdroj: autor]

NETTE FRAMEWORK

Pokus	INSERT - Trvání [ms]	UPDATE - Trvání [ms]	DELETE - Trvání [ms]
1	66,0	82,5	94,5
2	77,6	89,0	96,9
3	79,2	90,1	86,1
4	83,5	68,2	81,4
5	91,7	91,9	75,7
6	67,0	87,8	80,4
7	70,8	88,7	94,5
8	91,5	73,2	65,6
9	72,1	67,1	81,6
10	55,6	60,3	95,6
11	81,1	65,8	85,7
12	66,4	55,3	82,5
13	74,4	63,6	76,9
14	74,0	52,6	61,2
15	54,0	54,0	74,0
16	62,8	80,2	56,2
17	76,5	84,9	94,2
18	58,0	84,1	93,4
19	75,2	80,3	71,4
20	75,3	88,7	60,5
Průměr	72,6	75,4	80,4
Min	54,0	52,6	56,2
Max	91,7	91,9	96,9

Databázového spojení v modelové aplikaci bylo vytvořeno pomocí aplikační konfigurace. To znamená, že se do konfiguračního souboru jménem *config.neon* vložil následují kód.

```
database:
        dsn: 'mysql:host=127.0.0.1;dbname=Bakalarka'
        user:   user
        password:   pass
```

Tímto zápisem se kromě nastavení připojení k databázi také vytvořilo spojení se službou Nette\Database\Context, která je nutná pro práci s vrstvou Database\Table. „Vrstva Database\Table zjednodušuje a optimalizuje výběr dat z databáze.“ (Nette Foundation, 2008-2017a) V presenteru jménem *DatabasePresenter.php* se s využítím konstruktoru modelová aplikace přihlásí do databáze.

```
private $database;
   public function __construct(Nette\Database\Context $database) {
      $this->database = $database;
}
```

Jednotlivé dotazy na databázi jsou v modelové aplikaci tvořeny pomocí vrsty Database\Table a rozděleny do samostatných metod. Metody se na základě průběhu testování postupně volají. Zdrojový kód takové metody vypadá následovně.

```
public function insertDatabase() {
      for($i = 1; $i <= 100; $i++) {
         $this->database->table('posts')->insert([
            'id' => NULL,
            'title' => "Titulek",
            'content' => "Obsah"
         ]);
      }
}
```

Metoda *insertDatabase()* se volá pomocí *$this->insertDatabase()* a zajistí, aby se do tabulky jménem *posts* vložilo sto řádků. Do sloupců *title* a *content* se vloží již předem deklarovaný text, zatímco *id* sloupec má auto_increment a hodnotu si databáze sama vygeneruje.

SQL příkaz pro vytvoření tabulky *post* byl použit pouze jednou a to při prvním testováním.

```
CREATE TABLE `posts` (
  `id` int(11) NOT NULL,
  `title` varchar(255) NOT NULL,
  `content` text NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Verdikt

Při testu rychlosti Nette frameworku bylo zapotřebí vymazat veškeré scripty, které by mohly ovlivnit výsledné hodnoty měření (načítání externích souborů). Bylo ovšem třeba vypnout také debugovací nástroj Tracy, který je standartě zapnutý a který je pro měření nepotřebný. Bohužel i přes všechny tyto kroky má framework v průměru ty největší hodnoty a stává se nejpomalejším z testovaných frameworků. Hodnoty se přesto od ostatních frameworků liší maximálně v řádu deseti milisekund, proto bych Nette jen kvůli rychlosti zpracování dotazů na databázi nezavrhoval. Výslednou rychlost frameworku bych považoval za dostačující.

4.1.3 Komunita

V České republice není pravděpodobně webového frameworku, který by měl větší komunitu, než Nette framework. Na oficiálním diskuzním fóru se nachází přes 23 000 témat s 160 000 příspěvky. Existuje i oficiální anglické diskuzní fórum, kde je přes 1 300 témat a 6 000 příspěvků. (Nette Foundation, 2008-2017d) Založení anglického diskuzního fóra značí, že se Nette snaží dostat do povědomí i v jiných zemích, než jen v zemích České a Slovenské republiky. Výše uvedená čísla jen potvrzují to, že framework nezastarává, vývojáři se o něj starají a snaží se ho neustále vylepšovat.

Pokud něčemu vývojář v Nette frameworku nerozumí nebo se jen chce sejít s dalšími fanoušky vytváření webových aplikací, tak se každý měsíc může zúčastnit akce jménem Poslední Sobota, která se koná v České republice a většinou v Praze. Místo konání ale není pravidlem a pražští vývojáři občas cestují přes celou republiku. Pro vývojáře se nabízí ještě komornější setkání, které se jmenuje #NetteFwPivo. (Posobota, 2012) Veškerá diskuze je u piva a probírá se, co se nestihlo na Poslední Sobotě. Vstup na obě dvě akce je zdarma, jen je pouze zapotřebí potvrdit účast.

Přibližný počet webových stránek, které se ve svém obsahu zmiňují o Nette frameworku je přes 293 000. To rozhodně není malé číslo a vývojář tak má velkou šanci, že najde odpověď na svoji otázku.

Verdikt

Místo, kde se schází nejvíce uživatelů Nette frameworku, je na oficiálním diskuzním fóru. Tam se nachází mnoho témat, která se za ty roky stačila probrat. Začátečník, který se z nějakého důvodu při vývoji své aplikace zastavil, tam s velkou pravděpodobností najde odpověď, jak pokračovat ve vývoji své aplikace. Vytvořit druhé fórum, které bude pouze v angličtině je dobrý nápad. Cizinci se na anglickém fóru budou lépe orientovat a zároveň to frameworku zvýší šanci dostat se do světa. Pravidelné akce jsou dobré jak pro začátečníky, tak i pro pokročilé. Začátečník se naučí něco nového, zatímco pokročilý si může pohovořit s ostatními a jen si potvrdit to, co dělá, a že to dělá dobře.

4.1.4 Dokumentace

Píšeme první aplikaci je zvýrazněný odkaz, kterého si každý vývojář všimne, jakmile vstoupí na stránku dokumentace Nette frameworku. QuickStart je název sedmi na sebe navazujících kapitol, které vývojářovi pomohou začít s tvorbou první aplikací ve frameworku. Na první stránce jsou kapitoly přehledně vypsány, takže má vývojář přehled, jaké kroky bude absolvovat. Návod je napsaný pro nejnovější verzi 2.4. S výslednou aplikací z návodu vznikne funkční blok, kde administrátor píše články a uživatelé je mohou komentovat.

V návodu se vývojář dozví, jaké je základní struktura frameworku a k čemu slouží jednotlivé adresáře. Představí se mu Tracy (Laděnka), která je velmi užitečným nástrojem pro vývoj webové aplikace. Tracy je diagnostický nástroj pro jazyk PHP, který dokáže rychle odhalit a opravit chyby, logovat chyby, měřit čas a vypisovat proměnné. Pomocí Admineru, nástroje pro správu databáze, vytvoří vývojář databázi, kterou naplní daty, a následně vypíše řádky na webovou stránku. Návod vývojáře seznámí se šablonovacím systémem Latte. Latte má zabezpečit výstupy, ušetřit a zpříjemnit práci. Pátá kapitola obsahuje návod na to, jak vytvořit dobře zabezpečený formulář, který vývojář může využít například pro vkládání komentářů do diskuze. Nechybí ani kapitola o autentizaci uživatele, čímž se ve webové aplikaci zabrání tomu, aby nemohl psát komentáře neregistrovaný návštěvník webu.

V samotném závěru vývojář vytvoří modelovou třídu a pracuje s databází - seznamuje se s důležitostí modelu. (Nette Foundation, 2008-2017a)

Příručka obsahuje podrobný popis více než dvaceti komponent frameworku. Všechny komponenty jsou aktuální, tudíž napsány pro nejnovější verzi frameworku.

Na hlavní stránce nechybí odkaz na API reference, kde je k dispozici podrobný popis všech tříd, metod a parametrů. Stránky jsou přehledné a orientace v nich je bezproblémová.

Jedinou jazykovou mutací je angličtina. Při výběru jazykové mutace, byla angličtina dobrou volbou, jelikož to jazyk velice rozšířený. Zarážející je ovšem to, že při změně jazyka několik kapitol zmizí. Autor textu zřejmě na překladu některých kapitol stále pracuje.

Verdikt

QuickStart je velmi dobře strukturovaný a sepsaný. Informační hodnota pro začínajícího vývojáře je značná, nicméně chybí například vysvětlení routování URL, což je důležitou vlastností frameworku. Vývojář se k routování při vyvíjení aplikace dostane velmi brzy. Některé komponenty v příručce jsou zpracované opravdu detailně, některým naopak chybí důležité informace. Bohužel, jak jsem již zmínil, při změně na anglický jazyk některé kapitoly komponentu zmizí. Odkaz na API je lehce dohledatelný a dobře zpracovaný.

4.1.5 Vývoj

Vývojáři z Nette frameworku slibují, že každá zveřejněná verze bude aktivně udržována minimálně jeden rok a o její bezpečnostní a kritické chyby bude postaráno dva roky. “Během prvního období mohou být opravovány všechny nedostatky a konec období je označen jako *end of active support*. V dalším období jsou opravovány pouze kritické chyby a bezpečností problémy, konec je označen jako *end of life*.” (Nette Foundation, 2008-2017b) Na webových stránkách Nette se nachází přehledná tabulka všech vydaných verzí s odkazem na každou z nich. Pod odkazem je vývojáři k dispozici velice podrobný popis změn, které s verzí přicházejí.

4.2 Zend Framework

4.2.1 Bezpečnost

V Zend frameworku musí vývojář použít k obraně proti Cross-Site Scripting (XSS) komponentu jménem Zend\Escaper. Tato komponenta má třídu Zend\Escaper\Escaper, která nabízí pět metod pro escapování výstupů. Metody escapování výstupů HTML Body, HTML Attribute, JavaScriptu, CSS a URL. Záleží jen na vývojáři, jakou použije metodu a jestli jí použije na správném místě. (Zend, 2006-2017b)

Součástí komponenty Form v Zend frameworku je funkce Element\Csfr, díky které se vývojář dokáže bránit proti Cross-Site Request Forgery (CSRF) útoku. Pokud se při vytváření formuláře bude volat právě tato funkce, tak se automaticky s formulářem vytvoří skryté pole se zašifrovaným, náhodně vygenerovaným prvkem, který se při odeslání formuláře bude kontrolovat. Lze nastavovat čas platnosti prvku a pokud stránka obsahuje více, než jeden formulář je také možné pole pojmenovat, aby nedocházelo k přepsání prvků. (Zend, 2006-2017c)

Jelikož ve frameworku standardně není žádná funkce jako *init_set*, která by se starala o změnu konfigurace samotného PHP, podařilo se odcizit session identifikátor. Na localhostu byl konfigurační soubor PHP nastaven do základního nastavení. Jelikož si framework nenastavil direktivy PHP sám, útok proběhl úspěšně. Vývojář může podle dokumentace vytvořit konfigurační soubor, který se bude volat při spuštění webové aplikace, a který dle konfigurace může změnit direktivy PHP a tím zabránit ukradení session.

Verdikt

Pokud si vývojář začátečník pročte celý tutoriál od prvního odstavce až po poslední, nenajde tam jedinou zmínku o escapování výstupů. Vývojář, který chce zabezpečit svoji webovou aplikaci najde Zend\Escaper v dokumentaci. Tato komponenta by se ovšem měla dostat do podvědomí hned na začátku seznamování se s frameworkem. Framework dává na výběr vývojáři z pěti metod použití escapování, ale dle mého názoru by bylo lepší automatické rozpoznávání kontextu. Ochrana proti CSRF je zajištěná a její implementace do webové aplikace je velice jednoduchá. Výhodou je, že funkce umožňuje pojmenovat každé pole originálním názvem, webová stránka tak může obsahovat několik formulářů najednou.

Krádež session klíče proběhla úspěšně, ale jen z toho důvodu, že nebyl vytvořen konfigurační soubor, který by tomuto útoku zabránil. Zend dává možnost se útoku bránit, ale měl by tuto možnost nabídnout hned v základním nastavení frameworku. Zabezpečení je dobré, ale mělo by být v Zend frameworku více upřednostňováno a více zautomatizováno.

4.2.2 Rychlost

Obrázek 4: Výsledné hodnoty rychlosti pro Zend [Zdroj: autor]

ZEND FRAMEWORK

Pokus	INSERT - Trvání [ms]	UPDATE - Trvání [ms]	DELETE - Trvání [ms]
1	62,3	62,8	82,3
2	60,1	59,3	85,1
3	65,2	64,4	83,6
4	59,9	64,1	98,4
5	69,3	53,7	102,2
6	54,7	66,9	80,8
7	57,3	78,5	67,2
8	51,9	53,6	63,8
9	54,6	68,9	72,6
10	87,2	74,3	99,7
11	67,3	71,0	70,4
12	50,0	62,2	73,3
13	51,7	69,5	90,2
14	51,3	68,1	80,9
15	67,3	60,7	85,6
16	52,9	61,0	90,4
17	64,6	68,7	92,3
18	61,7	64,4	88,1
19	52,0	62,6	89,9
20	59,9	59,5	93,3
Průměr	60,1	64,7	84,5
Min	50,0	53,6	63,8
Max	87,2	78,5	102,2

V Zend frameworku modelová aplikace vyžaduje přítomnost Zend\Db\Sql\Sql komponenty, jelikož se stará o propojení aplikace s databází. Komponenta je abtrakní databázová vrstva, která vytváří specifické SQL dotazy podle platformy. Pomocí API dokáže pracovat s objektově orientovanými daty. Aby se dosáhlo přístupu k Zend\Db\Sql\Sql je zapotřebý vytvořit objekt třídy Zend\Db\Adapter\Adapter. Ve vytvořeném kontroleru je zapotřebý vložit *use Zend\Db\Sql\Sql;* řádek, který umožní se odvolávat na třídu Sql. Dále pak soubor pokračuje takto:

```
$driverArray = array(
        'driver' => 'Pdo_Mysql',
        'database' => 'Bakalarka',
        'host' => 'localhost',
        'username' => 'user,
        'password' => 'pass',
    );
    $adapter = new \Zend\Db\Adapter\Adapter($driverArray);
    $sql = new Sql($adapter);
    for ($i = 1; $i <= 100; $i++) {
       $insert->into("posts")->values([
          'title' => 'Titulek',
          'content' => 'Obsah',
       ]);
       $statement = $sql->prepareStatementForSqlObject($insert);
       $results = $statement->execute();
    }
```

Na začátku se vytvoří konfigurační pole, ve kterém se nastaví připojení k databázi. Konfigurační pole se předá adaptéru, který se propojí se třídou Sql a umožní vytvářet dotazy na databázy. Metoda *prepareStatementForSqlObject* slouží ke kontrole vykonávaných dotazů. *Execute()* provede příkaz.

Verdikt

Zend framework jsem nainstaloval do svého počítače pomocí Composeru, jelikož jsem chtěl mít pod kontrolou všechny nainstalované komponenty. Z toho důvodu jsem mohl, na rozdíl od Nette hned při instalaci zrušit stahování debuger a caching nástrojů. Ty jsou určitě dobrým pomocníkem při vytváření webových aplikací, při měření rychlosti by ovšem mohly favorizovat daný framework. Naměřené hodnoty jsou ve srovnání s ostatními frameworky velice dobré a ve většině dotazů na databázi byl nejrychlejší. Mazání z databáze je kategorie, ve které byl o přibližně osm milisekund rychlejší Symfony framework.

4.2.3 Komunita

Zend framework na svých webových stránkách píše o tom, že nebyl vytvořen pouze framework, ale že byla vytvořena hlavně dobrá komunita. Jelikož je to open source projekt, tak za svůj vývoj může děkovat tisícům přispěvatelů, kteří se starali a stále starají o framework. Na oficiálním diskuzním fóru je přes 16 000 témat, ve kterých je více než 51 000 tisíc příspěvků. (Zend, 2017a) Někoho může zarazit, že takto velký framework má tak málo témat. Jakmile se ale vývojář dostane na Zend GitHub[13], tak vše pochopí. Zend žádá vývojáře, kteří mají jakýkoliv problém s frameworkem nebo do něj chtějí přispět svým kódem, aby se přesměrovali na stránky GitHubu. Na GitHubu vývojář najde repozitář se všemi komponenty Zend frameworku. Po kliknutí na požadovanou komponentu může napsat *issue*, pokud má nějaký problém, nebo *pull-request*, pokud by chtěl nějakým způsobem přispět do zdrojového kódu frameworku. Těchto *issue* a *pull-requestů* jsou tisíce u každého komponentu. K tomu, aby mohl programátor přispět do zdrojového kódu, musí splnit určité požadavky, jinak se tomu vývojáři od frameworku nebudou zaobírat. (Zend, 2017b)

Webových stránek zmiňujících se ve svém obsahu jakkoliv o Zend frameworku je přes 420 tisíc, což je úctyhodný počet. Mezi výsledky se nachází i několik českých stránek, které jsou zaměřeny na framework a pravidelně o něm píšou články.

[13] Nejznámější server pro uložení open-source projektů, napojený na verzovací systém Git. (Vávra, 2012) Dostupné z: https://www.zdrojak.cz/?p=3677

Verdikt

Zend s heslem, že spolu s frameworkem založili i dobrou komunitu, mají pravdu. Počet zdrojů, které se zabývají jakýmkoliv způsobem Zend frameworkem je opravdu mnoho. Vývojář začátečník má mnoho možností, ze kterých zdrojů bude čerpat a nemusí se držet jen oficiálních stránek. Pravděpodobně nalezne i stránky ve svém rodném jazyce, které mu umožní se lépe seznámit s frameworkem. Veškerá práce se zdrojovým kódem je zajišťována přes GitHub, což může být ze začátku práce lehce nejasná, ale v konečném výsledku jsou veškeré komponenty se zdrojovým kódem pěkně strukturované a přehledné. Práce v GitHubu je efektivní a vložit celý projekt na tuto sít byl dle mého názoru pro Zend tím správným krokem. Zend na svých webových stránkách neuveřejňuje žádné oficiální akce, na kterých by se vývojáři mohli setkat. Myslím, že je to velká škoda, stačilo by jen pár zajímavých přednášek. Důležité je osobní setkání a také výměna zkušeností.

4.2.4 Dokumentace

Hlavní stránka nabízí možnost volby dokumentace všech verzí frameworku. Pokud se tedy vývojář dostane k jiné verzi než k té nejnovější, může bez problému hned začít. Zend framework nabízí soubor obsahující několik návodů, které mají pomoci vývojáři začít realizovat svojí první aplikaci.

První kapitola je návod pro úplně první práci s frameworkem. Radí, jak nainstalovat framework a nastavit Apache tak, aby vývojář měl úvodní webovou stránku a mohl s frameworkem začít pracovat. Postupně se vývojář dozvídá, jak funguje Model a jeho práce s databází. Následuje routování, předávání parametrů v URL a práce s kontrolorem. Vytváření formulářů, které budou validní, zabezpečené a dokážou pracovat s databází, se nachází v poslední kapitole návodu. Práce s pohledy doprovází vývojáře celým návodem, jinak by si nemohl zobrazit výstup aplikace.

Všechno popsané v odstavci výše je pouze jedna kapitola z mnoha. Pro základní pochopení fungování frameworku by první kapitola měla stačit, ale dále se v dokumentaci nachází - přidání navigace a stránkování, testování, pokročilejší konfigurace, návod na internacionalizaci a tzv. In-Depth. In-Depth kapitola obsahuje detailní články o vybraných komponent frameworku. Tato kapitola slouží pro vývojáře, kteří si chtějí své znalosti o daných komponent prohloubit.

Výpis všech komponent se nachází hned na hlavní stránce, najít je tedy není problém. S každým komponenty jsou spojeny dva odkazy, první odkazuje na GitHub a druhý na dokumentaci. Z GitHubu je možnost prohlížet zdrojový kód komponenty s možností stažení. V některých případech se v dolní části stránky na GitHubu objeví krátký návod na použití daného komponentu. Odkazem na dokumentaci se vývojáři zobrazí pouze příkazu pro Composer[14].

Odkazy na API reference Zend frameworku 1 a 2 se nachází hned na hlavní stránce a nelze je tedy přehlídnout. Bohužel odkaz na verzi 3 chybí. Vývojáři na ní nejspíše teprve pracují. Přehlednost v API referencích není příliš přehledná. (Zend, 2006-2017d)

Verdikt

Vývojáři viditelně odkazují na starší verze frameworku a proto není problém se kouknout na návody a do příručky nejstarší verze frameworku. Návody, které pomáhají vývojářům s prvními kroky ve frameworkem jsou velice rozsáhlé a obsahují několik podkapitol. Jsou napsané pouze v anglickém jazyce a to je určitě plus pro rozvoj frameworku. Jestliže vývojář ovládá alespoň trochu anglický jazyk, tak se nemusí bát. Návody jsou sepsány jednoduchou angličtinou, přesto výstižně. V první kapitole chybí možná pouze autentizace, jinak bylo vše podrobně popsáno. Příručka je na rozdíl od tutoriálů trochu zklamáním, jelikož by se dalo očekávat něco víc, než příkaz pro Composer a na GitHubu jen pár řádků ohledně komponentu.

4.2.5 Vývoj

Pravidelně vývojáři z Zend frameworku oznamují tzv. Long Term Support (LTS) verze, které vývojáři zaručí to, že tři roky bude jeho verzi podporována. Během třech let se taková verze bude udržovat a opravovat se budou bezpečnostní a kritické chyby. Pro vývojáře je dobré vědět, že verze z roku 2015 do roku 2018 bude aktuální a zabezpečená. Podle (Böhmer, 2010, s. 24) jsou požadavky na testování komponent v Zend frameworku vysoké. To může mít za následek to, že na některé verze se může čekat delší dobu. To pak ale

[14] Composer je pro správu balíku v prostředí PHP. Pomocí něho stahujete balíčky například webového frameworku a on vám zaručí, že automaticky vyřeší všechny závislosti ke stahovanému balíku. (IUNAS, 2013-2014) Dostupné z: http://via.iunas.cz/php/composer/instalace.html

vývojáři ve výsledku může ušetřit čas, jelikož se neopravují chyby již při běžící aplikaci. Veškeré změny, které se ve verzích provedli se nacházejí na stránkách Zend frameworku, kde je krátký popis toho, co všechno se změnilo. Pokud ale vývojář chce porovnat jednotlivě změněny řádků s předchozí verzí, tak může, skrz webové stránky GitHubu. (Zend, 2006-2017e)

4.3 Symfony Framework

4.3.1 Bezpečnost

Součástí Symfony frameworku je šablonovací nástroj Twig, který má za úkol vývojářům ulehčit a zpřehlednit prezentační vrstvu aplikace. Výhodou Twig nástroje je, že pokud mu vývojář neřekne jinak, tak jsou všechny výstupní data automaticky zabezpečená proti Cross-Site Scripting (XSS) útoku. Stačí tedy vložit proměnou dle pravidel nástroje do složených závorek a webová aplikace je okamžitě zabezpečená. Pro zrušení escapování je potřeba k proměnné přidat filtr *raw*. Chce-li vývojář escapovat v aplikační vrstvě v PHP, využije funkci *escape()*. Standardně funkce escapuje do HTML, ale druhý nepovinný parametr umožňuje vývojáři změnit escapování například do Javascriptu. (Symfony, 2016)

Proti Cross-Site Request Forgery (CSRF) útoku nemusí vývojář psát jediný řádek kódu, jelikož Symfony framework se o ochranu postará za něj. Zabezpečení se týká úplně každého formuláře, který vývojář vytvoří, ke každému z nich automaticky vytvoří skryté pole s časovým prvkem. Ten se při každém potvrzení formuláře kontroluje a ověřuje se totožnost uživatele. Funkce ochrany má několik možností, jak ji může vývojář přizpůsobit svým potřebám. Změnit lze doba platnosti prvku nebo mu vytvořit unikátní název, aby nedocházelo k přepsání. Vývojář má možnost ochranu i zcela vypnout. (Symfony, 2017b)

Symfony framework se ve svém zdrojovém kódu zabývá několika konfiguracemi a to například session, routováním, validací nebo překladem. Pro test krádeže session je důležité, aby framework v konfiguraci nezapomněl pracovat s direktivami PHP a proto dokáže proti krádeži session bojovat. Standardně je ochrana webové aplikace zapnutá, takže vývojář nemusí nic hledat ani nastavovat. V mém případě se session identifikátor odcizit nepodařilo a webová aplikace byla díky Symfony frameworku dobře zabezpečená. (Symfony, 2017b)

Verdikt

Jelikož Symfony framework používá šablonovací nástroj Twig a ten automaticky escapuje výstup, vývojář se tedy nemusí útokem XSS vůbec zabývat. Zabývat se nemusí ani CSRF útokem, jelikož ochrana proti němu je ve frameworku také zcela automatická. V obou dvou případech vývojář nemusí ve své webové aplikaci napsat ani jeden řádek navíc. S pokusem odcizit session identifikátor neměl framework jediný problém a útoku odolal. A to vše opět bez jediného zásahu vývojáře do zdrojového kódu své webové aplikace. Symfony je tedy výtečným pomocníkem při ochraně proti bezpečnostním útokům, jelikož je proti útokům velice odolný.

4.3.2 Rychlost

Obrázek 5: Výsledné hodnoty rychlosti pro Symfony [Zdroj: autor]

SYMFONY FRAMEWORK

Pokus	INSERT - Trvání [ms]	UPDATE - Trvání [ms]	DELETE - Trvání [ms]
1	69,7	86,6	57,6
2	70,1	93,7	54,9
3	72,8	85,7	65,9
4	54,8	72,1	76,5
5	59,9	87,9	73,1
6	87,1	78,5	53,8
7	79,3	77,5	67,2
8	69,0	58,6	64,2
9	61,9	57,4	51,7
10	53,8	63,9	64,1
11	52,5	59,5	77,3
12	61,8	61,9	49,1
13	62,2	69,0	69,1
14	60,6	66,0	68,3
15	81,6	75,1	70,5
16	69,7	56,2	77,2
17	75,5	66,1	78,5
18	82,8	66,1	68,7
19	80,6	65,0	70,9
20	78,3	70,1	65,9
Průměr	69,2	70,8	66,2
Min	52,5	56,2	49,1
Max	87,1	93,7	78,5

Nastavení připojení modelové aplikace k databázi se u Symfony frameworku provádí v konfiguračním souboru jménem *config.yml*. Výsledná podoba souboru vypadá takto:

```
doctrine:
    dbal:
        driver:   pdo_mysql
        host:     localhost
        dbname:   Bakalarka
        user:     user
        password: pass
        charset:  UTF8
```

Nastavení konfiguračního souboru pouze připravuje modelovou aplikaci k práci s Doctrine frameworkem[15] respektive s jeho DataBase Abstraction Layer (DBAL) vrstvou. Vrstva nabízí intiutivní a flexibilní API, které komunikuje s většinou relačních databází. Umožňuje tak pracovat s daty jako s objekty. (Symfony, 2017b).

```
public function updateDatabase() {
        $conn = $this->get('database_connection');
        for ($i = 1; $i <= 100; $i++) {
            $random = rand(1,2020);
            $conn->query("UPDATE posts SET title='TitulekNovy', content='ObsahNovy' WHERE
id=$random");
        }
}
```

Zdrojový kód ukazuje, jak vypadá metoda pro aktualizaci obsahu v tabulce *posts*. Nejprve metoda pomocí služby *database_connection* vytvoří spojení s databází a následně v cyklu aktualizuje obsah tabulky. Cyklus obsahuje funkci *rand*, která náhodně vygeneruje číslo, které se dosadí na dalším řádku do *id* hodnoty v dotazu na databázi. Parametry funkce *rand*

[15] "Doctrine 2 je nový ORM framework pro jazyk PHP. V porovnání s již existujícími systémy pro mapování objektů na relační databázi přináší zajímavý posun a má velkou šanci stát se v budoucnu převládajícím ORM pro aplikace psané v jazyce PHP." (Tichý, 2010) Dostupné z: https://www.zdrojak.cz/clanky/doctrine-2-uvod-do-systemu/

určují minimální a maximální hodnotu náhodně vygenerovaného čísla. Hodnoty parametrů jsou určeny podle počtu řádků z předešlého testování INSERT dotazu.

Verdikt

Symfony framework při nainstalování obsahuje debugovací nástroj, který pomáhá programátorům při vývoji, musel jsem ho tedy stejně jako v předchozích testovaných webových aplikacích vypnout. Cache komponenta byla představena v Symfony 3.1 (Symfony, 2017b) a není standartě zapnuta. V testu rychlosti dotazů na databázi si framework vedl celkem obstojně. V dotazu na vložení do databáze byl sice rychlejší než Nette, ale Zend ho o pár milisekund předběhl a to i v editaci. Naopak v dotazu na mazání byl rychlejší než Zend. Rychlost zpracování dotazů na databázi je dobrá.

4.3.3 Komunita

Symfony má jako jediný z frameworků na svých webových stránkách i část pouze o komunitě. Zde se vývojář dozví veškeré informace o tom, kolik je přispěvatelů (více než 300 000), kde a jak může přispět, nebo kde se bude konat nadcházející událost. Zaninotto (2005) píše, že oficiální diskuzní fórum vzniklo roku 2005 na podnět vývojářů. Aktuálně je ovšem pouze v režimu čtení. V hlavičce stránky Symfony framework informuje, že pokud má vývojář jakékoliv dotazy, tak má pokračovat na stránky Stack Overflow[16]. SensioLabs je rozsáhlá webová stránka, která je určena pro vývojáře Symfony. Pokud bude vývojář chtít aktivně komunikovat s komunitou, musí se právě na této webové stránce zaregistrovat. Po vytvoření profilu dostává uživatel odznáčky za aktivitu a tím se posouvá v žebříčku od začátečníků k pokročilým vývojářům.

Jelikož je Symfony velice rozšířený framework po celém světě, nepořádá události jen pro několik desítek lidí. Každý rok pořádá akci jménem Web Summer Camp, který navštíví tisíc lidí. Kemp je pořádán v Chorvatsku a trvá čtyři dny. Na události se sejde většinou více než patnáct přednášejících, kteří nemluví čistě jen o programování, ale také třeba o grafice webových stránek. Cena takového kempu je přibližně 15 000 korun. Tato cena je pouze za

[16] Stack Overflow je největší online komunita, kde se programátoři mohou učit novým věcem, sdílet své znalosti a rozvíjet svou kariéru (mají příležitost si najít práci). (Stack Exhange Inc, 2017) Dostupné z: http://stackoverflow.com/company/about

lístek a nejsou v ní započítané náklady na cestu. Velkou výhodu mají ti vývojáři, kteří bydlí v místě konání, jelikož další náklady mohou být pro některé návštěvníky i více než dvojnásobné. (Symfony, 2017c)

Výsledný počet webových stránek, které ve svém obsahu zmiňují Symfony framework je něco přes 262 000. Vzhledem k tomu, že pod frameworkem Symfony běží i populární projekty jako je Drupal či phpBB je to překvapivě nízké číslo.

Verdikt

Vývojáři si stěžovali na to, že webové stránky Symfony nejsou dostatečně interaktivní a proto bylo vytvořeno oficiální fórum, které je aktuálně nastaveno pouze pro čtení. V hlavičce je pouze odkaz na další stránky, na kterých je nutné se přihlásit a pouze tak je vývojáři umožněno upozornit nebo se dotazovat na nějaký problém. Pokud se vývojář rozhodne jakkoliv přispět do frameworku, tak se musí zaregistrovat na webovou stránku SensioLabs. Dále existuje Symfony Slack, který má sloužit jako komunikační kanál mezi uživateli frameworku. Komunitní systém se zdá celkově dost nejednotný a všechny otázky, odpovědi, příspěvky a zdrojové kódy jsou roztroušené na několika webových stránkách. Pro vývojáře by bylo daleko pohodlnější, kdyby mohl kontrolovat jen jednu jedinou stránku, na které by bylo pohromadě všechno potřebné. Naopak akce v Chorvatsku, kam každoročně přijede tisíce lidí, jsou s nápadem a dobře organizované.

4.3.4 Dokumentace

První věc, čeho si vývojář nejspíše všimne je, že chybí návod na vzorovou aplikaci. Místo toho dokumentace obsahuje sekci *začínáme*, kde jsou návody na framework, ale samotné návody na sebe nijak nenavazují a jejich pročtením nevznikne vývojářova první aplikace. V sekci *začínáme* je několik kapitol, které vývojáře seznámí s instalací frameworku, vytvořením první stránky, routingem a kontrolerem. Návod obsahuje popis tzv. Web Debug Toolbar nástroje, který se využívá na ladění a odstraňování chyb. V praxi se nachází v aplikaci vpravo dole, kde na něj stačí kliknout a zobrazí se přehled o použitých parametrech, přihlášení nebo routování. Dalším nástrojem frameworku Symfony je Twig. Ten se využívá u šablon, kde vývojáři dovolí psát stručný a přehledný kód tak, aby bylo jeho programování co nejefektivnější. Aby se vývojáři vůbec dozvěděli o tom, jakým způsobem fungují formuláře frameworku, musí do sekce *průvodce*. V této sekci je několik běžně

používaných funkcí, se kterými se vývojáři setkávají při tvorbě webových aplikací. Najít zde lze například - práce s formuláři a jejich validace, rozesílání emailů nebo práci s session.

Většina komponent je v příručce rozebírána velice podrobně, od instalace, až po popis pokročilejších funkcí. K některým komponentám je ovšem sepsán pouze návod na instalaci a jen několik řádků o tom, k čemu má komponent sloužit. Při procházení dokumentace se mi podařilo najít bohužel také jednu komponentu s nefunkčním odkazem.

Na stránkách dokumentace Symfony lze použít vyhledávací pole pro zobrazení API referencí. Do vyhledávacího pole je možné zapsat jakoukoliv verzi. Vždy se zobrazí webová stránka s API referencemi vyhledávané verze. (Symfony, 2017b)

Verdikt

Návod v dokumentaci, ve které se krok po kroku představuje Symfony framework a vývojář za jeho pomoci vytváří svoji první aplikaci, chybí. Takový návod by měl být nezbytnou součástí dokumentace. Nástroje Twig a Web Debug Toolbar jsou v dokumentaci představeny hned na začátku, ale nikde není zmínka o modelu. Symfony je vnímán jako framework, který funguje na návrhovém vzoru MVC s možností nevyužít kontroler. Důvod, proč tomu tak je, by mohl být v dokumentaci alespoň zmíněn. Některé funkce ze sekce *průvodce* by mohly být přítomny v sekci *začínáme*, jelikož se nimi vývojář setká hned při začátcích vytváření své webové aplikace. Příručka je velmi obsáhlá, bohužel je zde možné najít také nefunkční odkazy. Celá dokumentace je sepsaná v angličtině. Díky vyhledávacímu poli pro API reference je vyhledávání velice efektivní a jednoduché.

4.3.5 Vývoj

Long Term Support (LTS) jsou dlouhodobě podporované verze. Vývojáři se Symfony tyto verze udržují zabezpečené vždy čtyři roky od jejich zveřejnění, o kritické chyby se starají tři roky. LTS verze vychází každé dva roky a je určena pro vývojáře, kteří chtějí hlavně stabilní systém. Mezi těmito dlouho podporovanými verzemi vycházejí ještě Standart verze, ve kterých je o kritické chyby postaráno osm měsíců a o bezpečnostní čtrnáct měsíců. Standart verze jsou publikovány dvakrát ročně a jsou vhodné pro vývojáře, kteří chtějí pracovat s nejaktuálnější verzí a využívat tak výhod nejnovějších funkcí nebo se podílet na testování. Na stránkách Symfony frameworku lze najít plán všech připravovaných verzí do roku 2019,

který informuje všechny vývojáře, kdy by měly přijít nové funkce a v jakém období končí jejich vývoj. Symfony framework bere vývoj velice vážně a snaží se plnit plány, které si předem určil. Vývojář se proto může spolehnout na pravidelnost ve vydávání verzí, ať už major nebo minor. (Symfony, 2017a)

4.4 Závěrečné shrnutí

Neuspělo ani jedno ze tří bezpečnostních rizik, které podstupoval Nette framework při testování bezpečnosti. Nette na svých webových stránkách prohlašuje, že má dokonalé zabezpečení a "používá revoluční technologii, která eliminuje výskyt bezpečnostních děr". (Nette Foundation, 2008-2017a) Po otestování bezpečnosti frameworku je jasné, že vývojáři z Nette vědí, proč takové hesla na své stránky píší. Nemohu tvrdit, že Nette framework je nejbezpečnější webový framework, jelikož před všemi útoky webovou aplikaci neuchrání. Nabízí ale širokou škálu funkcí, které webovou aplikaci zabezpečí a ochrání před útoky. V rychlostním testu Nette tolik framework nevyniká, ale rozdíl v rychlostech se od ostatních liší jen v několika milisekundách. Nette je proto i tak velice rychlým frameworkem, který vývojářům pomáhá tvořit dostatečně rychlé webové aplikace. Komunita je tvořena především českými vývojáři a v České republice nemá webový framework konkurenci. (Zeman, 2016) To však má velký dopad na popularitu frameworku v zahraničí, kde se Nette jen velmi těžce prosazuje. Pokud se začínající vývojář před vytvářením své první webové aplikace podívá do dokumentace, bude to pro něj určitě dobrým startem, až na pár vlastností se seznámí s celým frameworkem. Kvalita anglického překladu dokumentace je na tom o trochu hůře. I přes to, že je malý počet vývojářů, kteří se podílejí na vývoji Nette, framework udržuje pravidelnost vydávání verzí (jak major, tak minor) a udržuje krok s technologiemi.

Výsledky testu bezpečnosti u Zend frameworku ukazují, že vývojář má možnost se bránit všem testovaným bezpečnostním rizikům Je ovšem zapotřebí, aby se o rizicích nejdříve vůbec dozvěděl. Ve frameworku je k dispozici několik komponent, které pomáhají zabezpečit webovou aplikaci, ale v dokumentaci nejsou zmíněny a vývojář si je musí najít sám. Při testu rychlosti má Zend jedny z nejlepších výsledků. Vývojáři se proto mohou spolehnout, že rychlost jejich webových aplikací bude na velice dobré úrovni. Základ komunity Zend frameworku se nachází na webové stránce GitHub. Tam vývojáři můžou

pohlížet na všechny verze frameworku, přispívat do zdrojového kódu, hlásit problémy nebo se zeptat na rady. Na internetu je k dispozici mnoho stránek o Zend frameworku, ale ty nejaktuálnější informace najdou vývojáři právě na GitHubu. Návody pro vývojáře jsou velice rozsáhlé a obsahují několik dalších kapitol a podkapitol, které do detailu obeznámí vývojáře o funkcionalitách frameworku. Bohužel chybí informace k jednotlivým komponentám a vývojář musí hledat informace jinde, než v příručce. Každá verze, kterou Zend framework vydá, projde testy. Na testy se kladou vysoké požadavky, proto je s příchodem nových verzí menší pravděpodobnost, že přijdou i chyby. Vývojáři se tak mohou spolehnout na jejich stabilitu a funkčnost.

O zajištění bezpečnosti webové aplikace se Symfony framework stará automaticky, a proto přes něj neprošel ani jeden z prováděných bezpečnostních útoků. V dokumentaci je o některých z testovaných útoků zmínka, ale vývojář nemusí napsat ani řádek kódu navíc a jeho webová aplikace bude i přesto zabezpečená. Framework má v testu zpracování některých dotazů rychlejší, než ostatní frameworky, v jiných je zase o trochu pomalejší. I přes to, že framework není v testu ten nejrychlejší, může vývojář vytvořit velice rychlou webovou aplikaci, protože rozdíl výsledných hodnot, jak jsem zmiňoval u Nette frameworku, je velice malý. Komunitní systém Symfony frameworku je dost nejednotný a vývojáři mají několik možností, kde se zaregistrovat a sledovat aktivitu frameworku. Na jednu stranu vývojáři mají možnost se rozhodnout, které webové stránky budou využívat, na druhou stranu jsou informace o frameworku dost roztroušené. Každoroční sraz v Chorvatsku je dobrá příležitost pro vývojáře Zend frameworku naučit se něco nového. Části dokumentace jako je například příručka, jsou velice dobře sepsané a velice obsáhlé. Některé části, jako je například návod pro vytvoření první aplikace ve frameworku, bohužel chybí. To pro vývojáře znamená, že musí procházet jednotlivé komponenty a postupně zjišťovat, jak která funguje. Symfony framework má na svých webových stránkách podrobný plán, který ukazuje přesné vydání jednotlivých verzí, a to až do roku 2022. Symfony má několik tisíc vývojářů (Symfony, 2017a), kteří se starají o vývoj frameworku a přesně plní plán, který si navrhli. Vývojáři proto přesně vědí, kdy přijde například minor verze nebo co mají očekávat v budoucnu za změny.

5 Závěr

Porovnáním Nette s ostatními testovanými frameworky pro vývojáře znamená, že se jim dostane dobře zabezpečeného a rychlého frameworku s rozsáhlou komunitou a to především v České Republice. Pokud je tedy webový vývojář z České Republiky, tak je to pro něj velkou výhodou a dostává se mu více možností rozvoje. Dokumentace je zpracována kvalitně, ale to bohužel neplatí pro anglický překlad. Nette framework se snaží vývojářům ulehčit mnoho práce a chová se velice *chytře* (dělá některé věci automaticky). To se hodí začínajícím vývojářům, kteří v začátcích potřebují trochu pomoci.

Zend framework se na rozdíl od Nette nechová tolik *chytře* a všechno závisí na vývojáři. To dává na jednu stranu vývojáři velkou moc, na druhou stranu se může stát, že na něco zapomene. I co se bezpečnosti se týče, vývojář musí všechno sám jasně definovat, jinak jeho webová aplikace bude nezabezpečená. Výhoda frameworku je, že vývojáři stačí pouze dvě webové stránky (oficiální stránky a GitHub), aby dokázal najít všechny aktuální informace. K výhodám patří i velice dobrá rychlost, protože vydané verze jsou velice ostře testovány a tak vykazují daleko menší chybovost.

Symfony posouvá hranici *chytrosti*, jelikož v případě bezpečnosti vývojář nemusí napsat jediný řádek kódu navíc a jeho webová aplikace bude i tak zabezpečená. Zabezpečení frameworku je na velice vysoké úrovni, ale i zpracování požadavků na databázi má velice rychlé. O tom, že se vývojář nemusí bát budoucího vývoje frameworku svědčí podrobně vypracovaný plán následujících verzí Symfony. V plánu je také zařazena akce v Chorvatsku, což pro vývojáře může být dobrá příležitost naučit se něco nového a setkat se s ostatními vývojáři Symfony.

Cílem práce bylo u každého webového frameworku ověřit jeho deklarované vlastnost, vyhodnotit jejich zásadní vlastnosti a ty poté porovnat s ostatními frameworky. Ze závěrečného shrnutí vyplývá, že každý z testovaných frameworků má svoje výhody, ale i nevýhody. Souhrn by měl pomoci vývojářům vybrat ten správný webový framework podle jejich představ. Osobně pevně doufám, že se v budoucnu najde vývojář, který si tuto bakalářskou práci přečte a rozhodne se díky ní snáze, jaký z dostupných webových frameworků začne používat.

Je docela možné, že kdyby modelové aplikace vytvořili vývojáři, kteří se v daném frameworku pohybují delší dobu, mohly by se výsledné hodnoty lišit. Do bakalářské práce se promítají moje vlastní zkušenosti s frameworky a znalosti, které jsem při práci s nimi získal. Rozdíl ve výsledných hodnotách by byl značnější, pokud by se vybraná metrika aplikovala na jednotlivé frameworky a testovala se pomocí rozsáhlejších a složitějších modelových aplikací.

6 Seznam použité literatury

BÖHMER, Marian. *Zend Framework: Programujeme webové aplikace v PHP.* 2010. Brno: Computer Press, 2010. ISBN 978-80-251-2965-4.

HISKEY, Daven. *The first website ever made.* Today I Found Out [online]. 2010 [cit. 2017-04-12]. Dostupné z: http://www.todayifoundout.com/?p=1458

ŠKRÁŠEK, Jan. *PHP Frameworky.* Programujte [online]. 2008 [cit. 2017-04-12]. Dostupné z: http://programujte.com/clanek/2008022000-php-frameworky/

ČÁPEK, David. *MVC Architektura.* IT Network [online]. 2017 [cit. 2017-04-19]. Dostupné z: http://www.itnetwork.cz/navrhove-vzory/mvc-architektura-navrhovy-vzor

TICHÝ, Jan. *Model není pouze databáze.* PHP Guru [online]. 2007 [cit. 2017-04-19]. Dostupné z: http://www.phpguru.cz/?p=19

NETTE FOUNDATION. *Dokumentace.* Nette Framework [online]. 2008-2017a [cit. 2017-04-19]. Dostupné z: https://doc.nette.org/cs/2.4/

NETTE FOUNDATION. *Download.* Nette Framework [online]. 2008-2017b [cit. 2017-04-19]. Dostupné z: https://nette.org/cs/download

NETTE FOUNDATION. *Fórum.* Nette Framework [online]. 2008-2017d [cit. 2017-04-20]. Dostupné z: https://forum.nette.org/cs/

FOWLER, Martin. *Supervising Controller.* Martin Fowler [online]. 2006 [cit. 2017-04-19]. Dostupné z: https://martinfowler.com/eaaDev/SupervisingPresenter.html

BERNARD, Borek. *Prezentační vzory z rodiny MVC.* Zdroják [online]. 2009 [cit. 2017-04-19]. Dostupné z: https://www.zdrojak.cz/clanky/prezentacni-vzory-zrodiny-mvc/

HASSMAN, Martin. *David Grudl: Nette Framework čeká zlomový rok.* Zdroják [online]. 2014 [cit. 2017-04-19]. Dostupné z: https://www.zdrojak.cz/clanky/david-grudl-nette-ceka-zlomovy-rok/

SKVORC, Bruno. *Best PHP framework 2015.* SitePoint [online]. 2015 [cit. 2017-04-19]. Dostupné z: https://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/

ZEND. *About.* Zend Framework [online]. 2006-2017a [cit. 2017-04-19]. Dostupné z: https://framework.zend.com/about

ZEND. *Zend Escaper Introduction.* Zend Framework [online]. 2006-2017b [cit. 2017-04-20]. Dostupné z: https://framework.zend.com/manual/2.1/en/modules/zend.escaper.introduction.html

ZEND. *Zend Form Element - CSRF.* Zend Framework [online]. 2006-2017c [cit. 2017-04-20]. Dostupné z: https://framework.zend.com/manual/2.4/en/modules/zend.form.element.csrf.html

ZEND. *Tutorials.* Zend Framework [online]. 2006-2017d [cit. 2017-04-20]. Dostupné z: https://docs.zendframework.com/tutorials/

ZEND. *Long Term Support.* Zend Framework [online]. 2006-2017e [cit. 2017-04-20]. Dostupné z: https://framework.zend.com/long-term-support

ZEND. *Forums.* Zend Forums [online]. 2017a [cit. 2017-04-20]. Dostupné z: http://forums.zend.com

ZEND. *Zend Framework GitHub Repository.* GitHub [online]. 2017b [cit. 2017-04-20]. Dostupné z: https://github.com/zendframework

ZANINOTTO, Francios. *The forum is now open.* Symfony Framework [online]. 2005 [cit. 2017-04-20]. Dostupné z: http://symfony.com/blog/the-forum-is-now-open

BLOCK, Jonathan. Contributor License Agreement (CLA). GitHub [online]. 2014 [cit. 2017-04-19]. Dostupné z: https://github.com/zendframework/zf1/wiki/Contributor-License-Agreement-(CLA)

SYMFONY. *Escaping.* Symfony Framework [online]. 2016 [cit. 2017-04-20]. Dostupné z: https://symfony.com/doc/master/templating/escaping.html

SYMFONY. *What is Symfony.* Symfony Framework [online]. 2017a [cit. 2017-04-19]. Dostupné z: http://symfony.com/what-is-symfony

SYMFONY. *Documentation.* Symfony Framework [online]. 2017b [cit. 2017-04-20]. Dostupné z: http://symfony.com/doc/current/index.html

SYMFONY. *Community.* Symfony Framework [online]. 2017c [cit. 2017-04-20]. Dostupné z: http://symfony.com/community

CENZIC. *Vulnerability Report.* Cenzic [online]. 2014 [cit. 2017-04-19]. Dostupné z: https://www.infopoint-security.de/medien/cenzic-vulnerability-report-2014.pdf

FERSCHMANN, Petr. *Přehled útoků na webové aplikace.* Zdroják [online]. 2008 [cit. 2017-04-20]. Dostupné z: https://www.zdrojak.cz/clanky/prehled-utoku-na-webove-aplikace/

PRESTON-WERNER, Tom. *Sematic Versioning.* Semver [online]. 2016 [cit. 2017-04-20]. Dostupné z: http://semver.org

NETTE FOUNDATION. *Poslední sobota.* Posobota [online]. 2012 [cit. 2017-04-20]. Dostupné z: https://www.posobota.cz

ZEMAN, Martin. *Výsledky: Technologie na českém webu.* Zdroják [online]. 2016 [cit. 2017-04-20]. Dostupné z: https://www.zdrojak.cz/clanky/vysledky-technologie-na-ceskem-webu/