Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Nette framework

Bakalářská práce

Autor: Lucie Gavlasová
Studijní obor: Aplikovaná informatika

Vedoucí práce: Ing. Jiří Štěpánek

Hradec Králové srpen 2017

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracovala samostatně a s použitím uvedené literatury.

V Hradci Králové dne 17.8.2017 Lucie Gavlasová

Poděkování:

Ráda bych poděkovala vedoucímu mé bakalářské práce Ing. Jiřímu Štěpánkovi za vedení práce a odborné rady. Zároveň bych ráda poděkovala své rodině za jejich dlouhodobou podporu.

Anotace

Tato bakalářská práce se zaměřuje na český PHP framework Nette. Jedna část je zaměřena na detailní charakteristiku frameworků a jejich architektura. Další část se vymezuje základními vlastnostmi a porovnáním nejvíce používaných frameworků. Zahrnuta budou frameworky pro jazyk PHP, ale i C# a JAVA. Ve třetí části je prakticky ukázáno využití Nette frameworku v reálné internetové aplikaci. Je představena aplikace pro firmu vyrábějící zdravé svačiny, ve které je možná objednávka jídel do škol nebo firem.

Annotation

Title: Nette framework

This Bachelor Thesis focuses on the Czech PHP framework Nette. The one part is focused on the detailed characteristics of the frameworks and their architecture. The next part is defined by the basic features and the comparison of the most used frameworks. Frameworks will be included for language PHP, but also C # and JAVA. In the third part, the use of the Nette framework in the real internet application is shown. An application for a healthy snack company is presented, which enables ordering meals for schools or businesses.

Obsah

1 Úvod

Webové aplikace jsou dnes již nezbytnou součástí internetu. Pro uživatele je mnohem výhodnější, aby například pro zpracovávání jakýchkoliv dat byly webové stránky dynamické. Nabízí se zde tedy otázka pro vývojáře, jak snadno takovou aplikaci vytvořit, aby byla kvalitní. Cílem všech tvůrců, ať už statických či dynamických webů je, aby jejich práce byla provedena co nejjednodušeji, ale zároveň přehledná a uživatelsky přívětivá. V případě, když vývojáři vše píší od prvního kódu sami, zjistí, že některé operace se stále opakují. Z tohoto důvodu jsou vyvinuty různé frameworky, pro ulehčení vývoje. Člověk se tak nemusí zabývat typickými problémy, ale může se zaměřit na tvorbu samotného projektu. Vytvoří to zároveň i předlohu, dle které bude kód přehledný.

Frameworků je vytvořena celá řada. Vždyť si musíme uvědomit kolik jenom existuje programovacích jazyků. A určitě je na některé jazyky dostupné hned několik frameworků. Asi nejvíce jsou používané jazyky C#, PHP a JAVA. Proto si je v práci představíme a navzájem porovnáme.

2 Cíl práce

Cílem práce je popsat Nette Framework a porovnat ho s podobnými možnostmi. Nette je sice PHP Framework, ale zde se bude srovnávat i s dalšími jazyky, aby bylo ucelené porovnání. Bude se sledovat jakou mají jednotlivá řešení architekturu a jejich základní vlastnosti. Díky tomu se zjistí, jak tento framework obstojí oproti konkurenci. V praktické části se ukáže, jak s Nette Frameworkem pracovat. Předvede se aplikace pro firmu Brunch, vyrábějící svačiny do škol a firem. Projekt bude zajišťovat objednávky svačin. Budou tam demonstrovány některé základní komponenty. Například se ukáže, jak se pracuje s uživatelskými rolemi a omezení jejich přístupu.

3 Obecné vymezení řešené problematiky

Webová aplikace je aplikace, která je dostupná pro uživatele prostřednictvím elektronické online sítě, jako je internet nebo intranet. Termín webová aplikace může také představovat počítačovou softwarovou aplikaci, která je na straně klienta kódována v jazyce srozumitelnému pro webový prohlížeč, konkrétně se například jedná JavaScript v kombinaci se značkovacím jazykem jako je HTML v jeho nejnovější verzi, přičemž je samotná aplikace závislá na obyčejném webovém prohlížeči, který vlastně zajišťuje její použitelnost.

Webová aplikace standardně používá kombinaci skriptů běžících na straně serveru – server-side script jako jsou ASP. NET, PHP a některé další, které budou vymezeny v dalších kapitolách textu. Stejně jako skriptů běžících na straně klienta – client-side script jako jsou HTML, JavaScript a další pro vytvoření webové aplikace. Skripty běžící na straně klienta se starají o prezentaci informace, zatímco skripty běžící na straně serveru mají na starosti výběr, zpracování a ukládání informací. (1) Využívání webových aplikací má řadu specifických pozitiv, ale také i konkrétní negativa, nejpodstatnější z nich budeme formulovat níže. (2)

Pozitiva:

- Webová aplikace zbavuje vývojáře odpovědnosti za vytváření různých rozhraní klienta pro specifické typy počítačů nebo operačních systémů. Jelikož klient běží ve webovém prohlížeči, uživatel může používat IBM, MAC, Windows nebo Linux, této souvislosti je to jedno.
- Webové aplikace jsou populární hlavně kvůli všudypřítomným webovým prohlížečům a pohodlí využívat webový prohlížeč jako klienta, někdy nazývaný i jako tzv. tenký klient, protože sám o sobě logiku aplikace nemá. Schopnost aktualizovat a udržovat webové aplikace bez instalování softwaru na tisíce klientských počítačů je hlavním důvodem jejich popularity a široké využitelnosti.

Negativa:

- Prvním z negativ je, pokud je internetová konektivita pomalejší, tak bude odezva aplikace časově delší, což znamená, že každý úkon bude trvat déle.
- Dalším, a podstatným důvodem, je bezpečnost, je tu šance neoprávněného vstupu a tím i znehodnocení nebo odcizení informací, případně i dalších dat, proto je zabezpečení velmi důležité.
- Webová aplikace musí být dostupná 24/7, proto je velmi důležité mít podporu a servis pro konkrétní server, na kterých běží aplikace stejně 24/7. Tato skutečnost je pro výběr vhodného poskytovatele také rozhodující.

4 Frameworky – programové rámce webových aplikací

Programový rámec aplikace představuje komplexní soubor návodů a specifikací, který poskytuje platformám, nástrojům a programovému prostředí, řešení designu, integraci, nastavení, bezpečnost a spolehlivost distribuce. (1) Programový rámec aplikace může obsahovat prezentační logiku, zpracování dat na straně serveru, management sezení, obchodní logiku, datovou logiku, ukládání dat do mezi paměti, služby pro přihlášení a mnoho dalších prvků, které je možné customizovat. (1)

V minulosti programátoři trávili hodně času při programování málo významných funkcí pro nové aplikace. Programový rámec webových aplikací zajišťuje standardní funkcionalitu aplikace jako přiřazování požadavků, volání metod, výběr a sestavení zobrazení. S jednoduchými funkcemi a předpřipravených architekturou stačí, když programátor jen malými změnami přizpůsobí interface a může se plně věnovat práci s daty. Ve zkratce, programové rámce webových aplikací zjednodušují používání technologií webových vrstev a tím pomáhají vývojářům aplikací soustředění se na problematiku řešení projektu. (2)

Je doporučeno vybrat si existující a ověřený programový rámec webové aplikace, jak navrhnout a vytvořit vlastní framework. Programový rámec webové aplikace poskytuje následující pozitiva: (3)

- **Odděluje vývojářské role.** Programové rámce aplikací již v základu poskytují různá rozhraní pro různé vývojáře. Toto oddělení dovoluje jednotlivým vývojářům pracovat nezávisle na sobě a usnadňuje další přístup.
- **Poskytuje centralizovaný bod kontroly.** Většina frameworků poskytuje bohatý výběr nastavitelných funkcí, jako například šablony, regulace přístupu nebo nastavení přihlášení.
- **Může být pořízený namísto jeho vytváření.** Čas, který by vývojáři strávili při vytváření podobné funkcionality, mohou věnovat vývoji obchodní logiky.
- **Poskytuje bohatý výběr funkcí.** Vybraný programový rámec může obsahovat mnoho užitečných funkcí, na které byste neměli znalosti pro jejich sestavení nebo čas na jejich vývoj.

- **Zabezpečuje stabilitu a bezpečnost pro aplikaci.** Frameworky jsou obvykle vytvářeny velkými organizacemi nebo skupinami, jsou používány a testovány ve velké síti. Proto frameworkový kód má větší pravděpodobnost být stabilní jako běžný kód.
- **Má komunitní podporu.** Populární frameworky přitahují komunity zanícených uživatele, kteří posílají hlášení o chybách, poskytují konzultace a tréninky, publikují návody k obsluze a vytvářejí užitečné add-ony.
- **Může podporovat ověřování vstupů.** Mnoho frameworků má důslednou specifikaci na validaci vstupů. Validace je běžně dostupná na straně uživatele, na straně serveru nebo jak na straně uživatele, tak na straně serveru zároveň.
- **Může být kompatibilní s různými nástroji.** Vhodné nástroje mohou zvýšit produktivitu a spolehlivost. Některé frameworky už mají integrované celé sady vývojářských nástrojů.

Všechny z těchto důvodů mohou výrazně pomoci designérům a vývojářům ve splnění požadavků aplikace. Využití frameworků má také významná negativa: (3)

- **Cena a cenové nastavení.** Některé frameworky je třeba zakoupit a nejsou tedy k dispozici bezplatně.
- **Nižší kontrola.** Vývojáři aplikací nemají takový přehled a kontrolu nad aplikací, pokud používají design někoho jiného ve vlastním softwaru.
- **Nejsou vždy bezchybné.** Pokud se používá cizí kód, může se stát, že obsahuje chyby, z nichž některé mohou být pro celkovou funkčnost frameworku zásadní.

Můžeme konstatovat, že mnoho vývojářů si myslí, že používání frameworků zvyšuje kvalitu implementace a designu.

4.1 Hlavní problémy při výběru vhodného frameworku

Velká potřeba programových rámců webových aplikací vyústila vydáním velkého množství frameworků, zejména v posledních letech. Každý prezentuje výhody toho svého řešení, ale mezi komunitami se vedou rozsáhlé diskuse, který framework je lepším, vhodnějším a univerzálnějším řešením. Toto značně znemožňuje poskytnutí

podstatných informací o pozitivech a negativech konkrétního produktu – frameworku. Proto neexistují žádné "univerzální" návody na to, jak vybrat ten správný framework, který by byl nejvýhodnější pro vývoj konkrétní aplikace.

Z tohoto hlediska je nejdůležitější zvážit všechna pozitiva používání frameworku. Protože nesprávně zvolený framework může významně znepříjemnit vývojový proces, čímž by se mohl prodražit nebo časově výrazně natáhnout. Další problém spočívá v tom, uvědomit si, že nejdůležitější otázkou v procesu výběru správného řešení požadované aplikace je: *"Do jaké míry je vybraný framework vhodný pro vývoj požadované aplikace?"* V praxi to znamená, že neexistuje takový framework, který by byl vhodný pro řešení všech typů projektů.

Můžeme hovořit o tom, že klíčovou částí při výběru aplikačního frameworku bude to, zda je zvolený framework kompatibilní se zvolenou IS architekturou, a zda je nastavení poskytovaných služeb na odpovídající úrovni abstrakce. Pro autory frameworku je nastavení správné úrovně abstrakce velmi důležité rozhodnutí. Na jedné straně je to touha zjednodušení a zredukování vývojářského úsilí pro programátora konečné aplikace, ale na druhé straně zjednodušení je příliš omezující a tím limituje možnosti programátora konečné aplikace. Je to nejzákladnější a v praxi poměrně často přehlížena úloha, která může ušetřit spoustu času a finančních prostředků. I z tohoto důvodu je nutné také této části věnovat patřičné časové i odborné úsilí. (3)

Komunita vývojářů aplikací má stále nedostatky z hlediska volby správné metodologie při výběru správného a vhodného frameworku pro konkrétní aplikaci. Stále není jasné, jaké druhy otázek by měly být zodpovězeny, a která kritéria by měla být splněna při výběru správného frameworku. Na druhé straně nastává otázka: *"Měla by být tato procedura provedena vždy, když vývojář potřebuje framework?"* Tato procedura by mohla být velmi zjednodušená, pokud by lidé, kteří pracují s různými technologiemi, sdíleli by své zkušenosti a mohli je tak vhodně komparovat.

Aktuální vývoj je směřován na vytvoření metodologie pro porovnání programových rámců webových aplikací a ohodnocení momentálně dostupných frameworků.

Můžeme konstatovat, že jsou poměrně populární OpenSource technologie webových aplikací, které se rychle rozrůstají a přitahují stále více pozornosti. Toto vyvolává následující otázky nebo konkrétní problémy: (2)

- Jaké jsou výhody webových technologií při vývoji frameworků?
- Které frameworky jsou dostupné pro vývoj webových aplikací?
- Které jsou důležitá kritéria pro komparaci jednotlivých frameworků?
- Jak zjistit který framework patří ke konkrétní aplikaci?
- Jak seřadit a kategorizovat takové rozsáhlé množství frameworků, aby bylo možné pozorovat trendy v technologii?

5 Charakteristika vybraných frameworků

Na počátku vývoje webových technologií, webové stránky obsahovaly pouze statický obsah a několik CGI skriptů. V současnosti se i vlivem nových technologií a technologického rozvoje vyžaduje od webových aplikací mnohem více interaktivity a funkcionality. Množství technologií webových aplikací stále narůstá a směrování jejich vývoje je těžko předvídatelné. Problém vzniká při výběru platformy pro vývoj webové aplikace, která bude muset splňovat všechny moderní požadavky. Ke splnění tohoto požadavku je důležité být dobře seznámen s aktuálním vývojem v této oblasti. Zaměříme se tak na evoluci vývoje webových aplikací v podobě přehledu aktuálně dostupných frameworků. V této souvislosti charakterizujeme výhody a nevýhody používání OpenSource technologií. (1)

5.1 OpenSource

Většina zmíněných frameworků je dostupná jako OpenSource, což je důvodem k diskusi tohoto fenoménu, jeho pozitiv a negativ pro obchodně-podnikatelské činnosti a vývojáře. Produkt je nazýván jako OpenSource právě když je dodáván s otevřeným zdrojovým kódem, který může být dále upravován. Některé licence požadují, aby vývojáři tohoto produktu měli přístup ke všem změnám a případným aktualizacím. Toto je opak vlastního softwaru, kde není poskytován zdrojový kód. Momentálně už množství společností používá OpenSource software, a ty ostatní začínají uvažovat o této možnosti. Aplikace OpenSource je dána zejména finančními požadavky a konkrétní finanční úsporou. Je důležité detailně zvážit správné použití, a ne pouze kopírovat řešení od konkurence, protože některé projekty mohou přinést mnoho pozitiv, ale jiné mohou zcela ztroskotat. Proto při rozhodování je třeba vybrat kompromis mezi hlavními pozitivy a riziky. (1)

5.1.1 Specifická pozitiva OpenSource

- **Otevřenost –** když má vývojář k dispozici zdrojový kód je pro něj jednoduché řešit problémy. To znamená, že není závislý na jednom dodavateli.

- **Přizpůsobivost** – OpenSource je jednoduše přizpůsobitelný software, protože ho můžeme upravovat, aby mohl spolupracovat s jiným OpenSource softwarem, a dokonce i s uzavřenými protokoly nebo jiným vlastním softwarem. Toto řeší situace uzavřeného kódu, který mnohdy omezuje to, když je pouze dodavatel schopen řešit konkrétní situace.
- **Cena** – OpenSource nutně nemusí znamenat, že je k dispozici zdarma. Většina OpenSource softwaru je volně dostupná a nevyžaduje žádné licence ani jiné poplatky. To umožňuje redukci ceny, tím trávení více času při vytváření bezpečnějších a personalizovanějších řešení pro dané potřeby.

5.1.2 Negativa OpenSource

- **Náročná použitelnost** – OpenSource balíky jsou v mnoha případech vytvářeny profesionály a jsou určeny také pro profesionály. Mnoho funkcí vyžaduje vytvoření konfiguračních souborů, skriptů a předkompilovanými zdrojového kódu, což může značně znemožnit manipulaci.
- **Vysoká cena operativy** – produkty OpenSource vyžadují vysoký stupeň technické odbornosti pro operativu a udržování produktu.
- **Vysoká cena technické podpory** – podpora je dražší, protože OpenSource nemá standardně žádnou podporu.
- **Bez záruky** – nikdo neposkytuje záruku na OpenSource, protože je to OpenSource.

5.2 Členění frameworků

Počet frameworků, které byly vydány za posledních pět let je z obecného hlediska velký. Všechny jsou založeny na různých principech, ale obecně je můžeme rozdělit do následujících skupin: (2)

- Architektura s MVC přístupem,
- Architektura publikačního XML,
- Architektura s přístupem na skupiny šablon.

5.2.1 Charakteristika architektury MVC

Zkratka MVC je definována jako Model, View a Controller. Každé z těchto slov označuje určitou úroveň tohoto architektonického schématu. Pojem MVC byl poprvé použit v roce 1979 při práci na programovacím jazyce SmallTalk. SmallTalk je objektově orientovaný programovací jazyk, který měl vliv na mnoho moderních programovacích jazyků. MVC definuje tři základní vrstvy: (4)

- Model, datový model aplikace, reprezentuje informace, na kterých aplikace pracuje, její obchodní logiku,
- View, prezentační vrstva, transformuje model do internetové stránky, vhodné pro komunikaci s uživatelem.
- Controller, kontrolní vrstva, reaguje na uživatelovu činnost na stránce podle potřeby a vyvolává změny v datovém modelu, nebo v prezentační vrstvě.

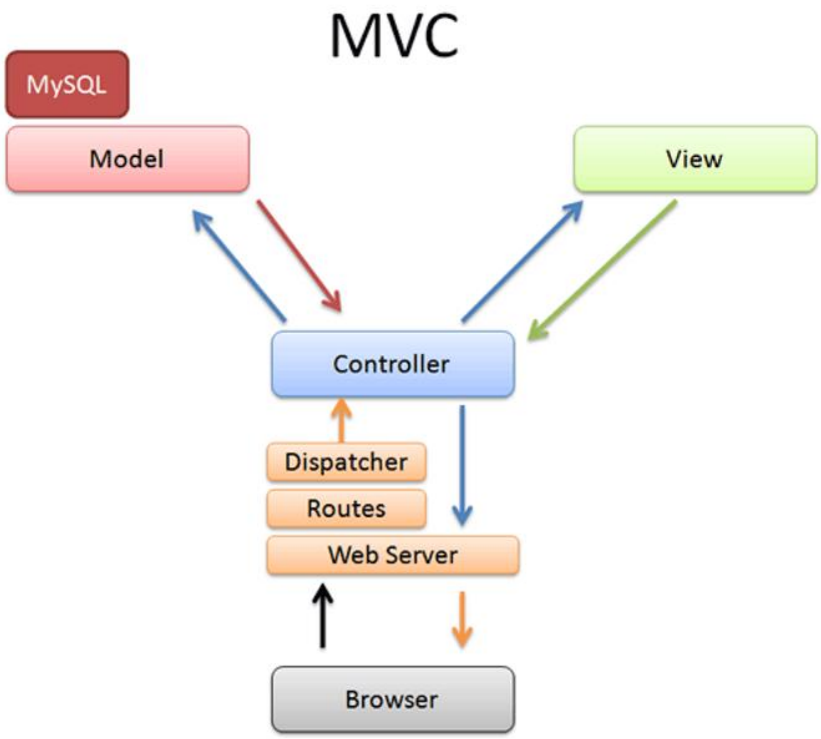
MVC v kontextu webových aplikací je velmi logickou strukturou. Prezentační vrstva je samotná HTML nebo XHTML stránka spolu s CSS styly. Kontrolní vrstva je kód, který dynamicky vybírá data a generuje obsah v rámci HTML šablony, konkrétně například PHP, ASP, JAVA. Poslední vrstvou, datovým modelem, je samotný obsah, nebo data, která jsou uložena v databázi, XML souboru nebo jiném zdroji. Tato vrstva zároveň obsahuje obchodní logiku čili pravidla, podle kterých upravuje obsah na základě činnosti uživatele. (4)

Příklad spolupráce vrstev v MVC architektuře pro akci přidání do košíku v elektronickém obchodě zahrnuje následující činnosti, obecně platný postup pro uplatnění v různých typech elektronických obchodů.

1. Uživatel interaguje s rozhraním, prezentační vrstva, tak, že vybere produkt a klikne na tlačítko "Přidat do košíku", čímž odešle požadavek – request na server.
2. Controller dostane upozornění o uživatelově požadavku z prezentační vrstvy a posune ji dál, správná akce – přidání do košíku.
3. Akce na přidání do košíku zpřístupní datový model, aby aktualizovala objekt "košík".

4. Pokud tato úprava proběhne v pořádku, akce připraví obsah, který se přiloží do odpovědi – response na zadaný požadavek.
5. Prezentační vrstva sestaví HTML kód stránky z odpovědi akce a šablony stránky.
6. Rozhraní čeká na další činnost uživatele, která cyklus znovu nastartuje.

MVC struktura je uvedena na schématu níže.



Obrázek 1 - MVC struktura (5)

5.2.2 Hlavní výhody používání MVC architektury

Z výše uvedených informací vyplývá mnoho pozitiv pro vývoj webových aplikací ve frameworku s MVC přístupem, které již byly zmíněny. Nejviditelnějším přínosem je přehlednost kódu. Když je rozdělen na logické vrstvy, vývojář při jakémkoliv problému hned ví, zda má hledat chybu v šabloně dokumentu, akci zpracovávající data, zda v datovém modelu. Zároveň takto napsaný kód je čitelnější a srozumitelnější i pro člověka, který ho nepsal. (1)

5.2.3 Hlavní nevýhody používání MVC architektury

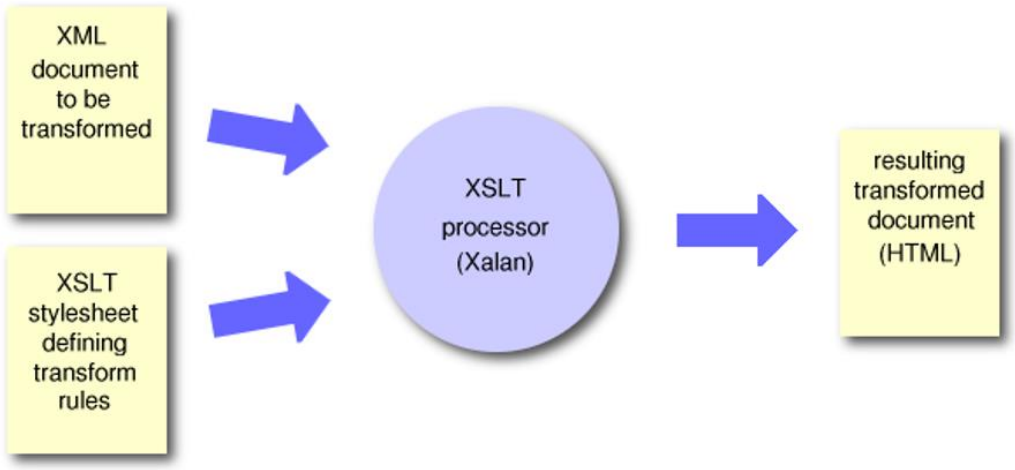
Přidáváním funkcí a různých vylepšení se framework stává rozsáhlým a robustním. Použít takový systém na vytvoření jednoduché internetové stránky je nevhodné a hlavně zbytečné. Několik testů výkonu a rychlosti ukázalo, že jednoduchá webová stránka vytvořena čistě v PHP je mnohem rychlejší, než kdyby byla vytvořena ve frameworku. (1)

5.2.4 Charakteristika architektury XML

Základem pro tyto typy frameworků je model 2X a jeho základní komponenty jsou: (2)

- **XML Dokument** – toto je dokument s hierarchickou strukturou, který obsahuje data pro konkrétní objekt nebo skupinu objektů.
- **XSL Dokument** – toto je XML style sheet, obsahuje instrukce pro formátování příslušného XML dokumentu.
- **XSL Transformátor** – toto je software, který je zodpovědný za provedení XML dokumentu a aplikování pravidel vyspecifikován v XSL dokumentu. Výsledný dokument je výstup transformace – může to být v podstatě cokoli od webové stránky po PDF dokument.

Komponenty publikačního XML frameworku jsou uvedeny na schématu níže.

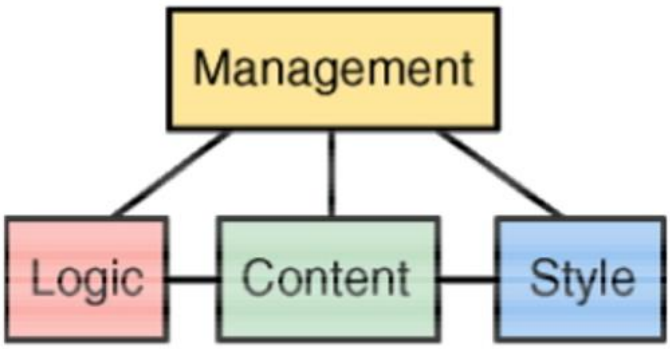


Obrázek 2 - Komponenty publikačního XML frameworku (2)

Používá se například ve frameworku Cocoon. Cocoon má velmi široké komerční využití, dovoluje dynamické publikování XML obsahu s použitím XSLT transformací. Cocoon se skládá ze tří částí které by měly by oddělené:

- **Obsah** – informace, které se zobrazují na webové stránce, skladované v XML dokumentech.
- **Styly** – výsledný vzhled stránky, nejčastěji implementován použitím XSLT transformací.
- **Logická struktura** – která se používá pro generování dynamického obsahu pro uživatele.

Všechny tyto části mohou pracovat společně za předpokladu, že vztahy mezi jednotlivými částmi se nezmění. Toto dovoluje vývojářům pracovat produktivněji při vytváření webové aplikace, přičemž minimalizují spotřebované zdroje. (2) Vztahy ve frameworku Cocoon jsou vyjádřeny ve schématu níže.



Obrázek 3 - Vztahy ve frameworku Cocoon

Cocoon je založen na hlavním principu rozvodné architektury. Rozvody sestávají z konkrétních vstupních údajů, následovaných množstvím procesních kroků. Každý procesní krok má jako vstup, tak výstup předchozího kroku. Na konci rozvodů je vyprodukovaný finální výstup. Cocoon má několik rozvodových komponentů, které mohou být seskupeny v závislosti na roli v rozvodů.

- Vstupy rozvodů jsou: generátory a čtečky.
- Procesní kroky jsou: transformátory a akce.
- Výstupy rozvodů jsou: serializátory.
- Podmíněné procesy jsou: výběrčí a spojovací.

Rozvody Cocoon se v zásadě většinou skládají z generátorů a serializátorů, ale mohou také obsahovat jakékoliv množství procesních kroků. Data procházejí rozvody jako SAX akce. Generátory odpovídají za čtení údajů z datového zdroje, tato data přecházejí do rozvodů jako série SAX akcií. Na jeden XML dokument, který se dostane do rozvodů, může být použito i více různých transformátorů. Transformátory jsou hlavní procesní kroky v Cocoon rozvodech. Přijímají SAX akce jako vstupy, provedou potřebné procesy, a posunou výsledek do dalších rozvodů, zase jako SAX akce. Poslední částí rozvodů je serializér. Serializér skládá proud SAX akcií přímo odevzdaných generátory nebo předcházejících procesních kroků do formátu, který je potřebný pro odpověď. Serializér může produkovat XML dokumenty, HTML, čistý text, PDF a mnoho jiných formátů.

Tyto tři komponenty jsou jádrem jakýchkoliv rozvodů. Ostatní komponenty, které rozvody mohou obsahovat, jsou popsány níže.

- **Čtečky** jsou speciální případ generátorů vstupu, nejsou komponenty XML. Jejich úkolem je přistoupit na externí zdroj, a zkopírovat ho přímo do odpovědi. Normálně jsou používány pro práci se statickými daty, jako jsou obrázky nebo kaskádové styly.
- **Akce** jsou prostředky pro připojení doplňkových dynamických funkcí do rozvodů a často jsou specifické pro konkrétní aplikace.
- **Spojovače** jsou ekvivalentem pro "if" podmínku. Pokud je nějaké tvrzení pravdivé pak proces pokračuje v patřičné části rozvodů.

5.2.5 Hlavní výhody tohoto frameworku

- Oddělení obchodní logiky od prezentační logiky. Část XML reprezentuje obchodní logiku, a data, s nimiž webový vývojáři pracují. Zatímco XSL obsahuje prezentační logiku, se kterou pracují webový designéři. (2)
- Architektura rozvodů poskytuje velmi přehledný design samotné aplikace. (2)

5.3 Nette framework

Nette Framework je framework, který podporuje tvorbu aplikací založených na architektuře MVP. Nette je framework s otevřeným zdrojovým kódem napsaný v jazyce PHP 5 s plným využitím objektů. Nette je vyvíjen v českém programátorském prostředí. Při tvorbě aplikace v prostředí frameworku musí programátor přizpůsobit určitým pravidlům, které mu mají usnadnit tvorbu aplikace. Frameworky obvykle za programátora dělají věci, které se mnohokrát opakují a stávají se nudnou rutinou.

Například tvorba formulářů, ošetření proti SQL injection a XSS útokům a mnohé jiné. Na straně druhé si využití frameworku vybírá specifické nároky v podobě zmíněných pravidel. Dále, framework může mít určité limity při vytváření některých komponent aplikace. Například pokud programátor potřebuje vytvořit netradiční formulář, framework nemusí nabízet zabudované funkce na takové případy a tehdy je třeba kombinovat zabudované funkce s vlastním kódem. Případně napsat celé řešení samostatně. (1)

5.3.1 Životní cyklus aplikace

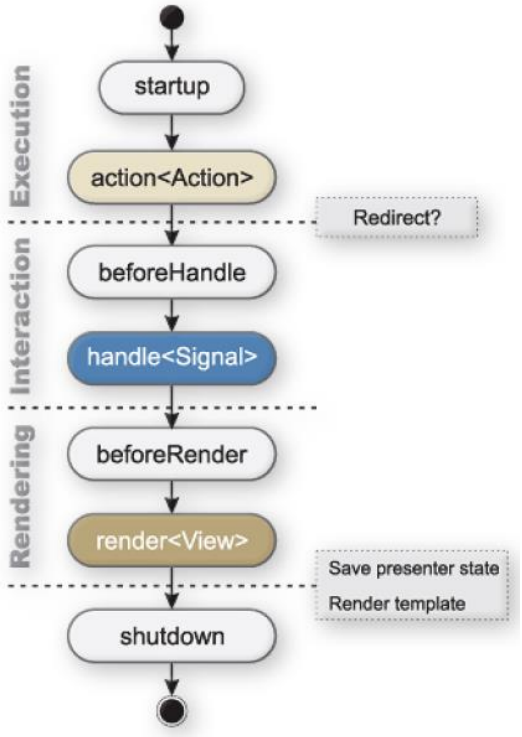
Životní cyklus aplikace lze rozdělit do následujících bodů:

1. Router z URL vytvoří objekt PresenterRequest, objekt obsahuje název presenter.
2. PresenterLoader z názvu Presenter odvodí třídu a případně název souboru.
3. Presenter vybírá metody podle aktuálního action a view, případně i subrequest.
4. Presenter načte šablony
5. Renderování: v tomto a předchozím bodě se obvykle vytvářejí odkazy na jiné Presenter a jejich akce a pohledy – action a view, do toho se zapojuje opět PresenterLoader a Router.

Takovým pomyslný jádrem aplikace je presenter. Presenter reaguje na události pocházející od uživatele a zajišťuje potřebné změny v modelu. Také presenter sám o sobě má určitý životní cyklus, který se dělí na čtyři fáze:

1. výkonná fáze – execution,
2. změna vnitřních stavů – interaction,
3. vykreslování – rendering,
4. ukončení činnosti – shutdown.

Životní cyklus Presenter je uveden na schématu níže.



Obrázek 4 - Životní cyklus Presenteru

Na schématu výše jsou bílou barvou označené metody pro všechny akce a pohledy. Modrou barvou je označena metoda zpracovávající konkrétní signál a hnědou metoda pro konkrétní pohled. Výkonná fáze obsahuje dvě metody. Metoda startup je vyvolána při každém začátku životního cyklu Presenter. Metoda nemusí být explicitně napsána, v případě, pokud chybí, framework se o její načtení postará. Metoda action {Action} by měla obsahovat provedení operací, po kterých by mělo nastat přesměrování.

Fáze měnící vnitřní stavy obsahuje metodu handle {Signal}, která zpracovává signály – subrequesty. Fáze vykreslování obsahuje metody beforeRender a render {View}. Metoda beforeRender může obsahovat nastavení filtrů pro vykreslení.

Metoda render {View} má na starosti vykreslení pohledu a tvorbu odkazů v šablonách a přiřazení proměnných do šablon. Pak přichází fáze, kdy se uloží všechny vnitřní stavy aplikace a perzistentní proměnné a šablona se vykreslí na výstup. Následuje fáze ukončení činnosti vyvolána při ukončení životního cyklu Presenter. (1)

6 Komparace frameworků a výběr vhodného řešení

6.1 ASP. NET

ASP. NET je produkt společnosti Microsoft, která když vydávala jeho první verzi, nemohla tušit, jak velký úspěch bude ASP mít. ASP. NET se velmi rychle rozšířil mezi vývojáři a začal se využívat při vytváření a vývoji webových aplikací. ASP. NET prošel postupem času vývojem, jeho první verze přišla na trh již v lednu 2002. Poslední verze je 4.5, a jako taková je podporována operačními systémy Windows Vista a jeho vyššími verzemi až do Windows 10. Jedním z nových prvků této verze je .NET for Metro Style apps. Metro je designový jazyk, který také vyvinula společnost Microsoft. Název tohoto jazyka byl inspirován reklamními tabulemi, které jsou umístěny v metru, protože aplikace vytvořené v tomto designérském jazyce jsou zaměřeny na typografii a pohyb. Tvůrci frameworku ASP. NET se věnovali zejména vývoji a zdokonalování serverových částí a nevěnovali až takovou pozornost vizuální stránce, neboť při webových aplikacích se o ni starají kaskádové styly. (6)

6.1.1 Návrhové vzory ASP. NET

Návrhové vzory se používají zejména při vytváření webových aplikací. Za jejich vznikem stojí architekt Christopher Alexander, který jako první přišel s myšlenkou používání návrhových vzorů v architektuře. V architektuře se objevují opakovaně tytéž problémy, které lze řešit pomocí návrhových vzorů. Jejich podstatou je konkrétní řešení problému, který se opakuje a tím je možné takový problém pokaždé vyřešit pomocí konkrétního návrhového vzoru. (7)

Stejně jako se návrhové vzory aplikují v architektuře, jsou použitelné i v softwarové oblasti. Softwarové návrhové vzory mají původ u vývojáře, kteří dlouhá léta vyvíjeli objektově orientované aplikace. Jejich zkušenosti jsou sepsány a byli i vydané v knižní podobě, a to souhrnně jako *"Bible návrhových vzorů".* Tuto publikaci

vytvořili čtyři vývojáři, kteří jsou také označováni jako GoF.[1] Podle nich jsou návrhové vzory rozděleny do tří kategorií:

1. **Tvorba** – zabývá se tvorbou objektů.
2. **Struktura** – zaměřuje se na vztahy mezi objekty a na jejich dědičnost.
3. **Chování** – zabezpečuje komunikaci mezi objekty.

Výhody používání návrhových vzorů můžeme rozdělit do dvou rovin, konkrétně: (7)

1. **Zvýšení produktivity** – při využití návrhových vzorů, programátor nemusí vymýšlet řešení, které byly již vymyšlené a tím může ušetřit dost velké množství času, který může strávit při řešení podstatnějších problémů a optimalizaci aplikace.
2. **Vylepšení kódu** – návrhové vzory jsou zpracovány profesionálními programátory, proto si můžeme být jisti, že pomocí nich víme vytvořit velmi kvalitní objektově orientovanou aplikaci.

Nejdůležitějším faktorem je, že návrhové vzory jsou testovaná a odzkoušená řešení, proto je jejich efektivita vysoce důvěryhodná. Jejich podstata spočívá v řešení neustále se vyskytujících problémů. Můžeme konstatovat, že ne všechny podobné problémy jsou totožné, aby bylo možné na jejich řešení použít návrhové vzory. Problém můžeme rozložit na srovnatelné části a pomocí návrhových vzorů ho vyřešit částečně. Označování návrhových vzorů je velmi důležité, protože názvy odrážejí jejich účel a chování, čímž můžeme poměrně rychle zhodnotit, který bude vhodný pro řešení konkrétního problému.

Samozřejmě neexistuje návrhový vzor na vyřešení každého problému. Vývojáři, kteří již jednou využili pro řešení problémů návrhové vzory, mají tendenci je využívat na řešení zcela všech problémů, což může v některých situacích vést

[1] V anglickém originále Gang of Four.

k přesně opačnému efektu, a to, že z jednoduchého problému vytvoří mnohem složitější. Proto je důležité před použitím návrhového vzoru pečlivě zvážit, zda je řešením konkrétního problému nejlépe použít návrhovým vzor, protože ne vždy je třeba sahat po tomto řešení. (7)

6.2 Návrhové principy

Návrhové principy tvoří základ návrhových vzorů, a proto jsou jejich důležitou součástí. Níže si uvedeme nejznámější a nejpoužívanější principy:

- **KISS** (Keep It Simple Stupid) - toto je princip, který se snaží o odstranění nepotřebných složitých částí kódu při vývoji aplikací, a tím naučit programátora vytvářet jednoduchý kód.
- **DRY** (Dont Repeat Yourself) - princip, který zabraňuje opakování se částí kódu v logice systému, což znamená, že každá část kódu, která představuje určitou logiku, může být definovaná pouze jednou.
- **Tell, Dont Ask** - zásada, která se týká zapouzdření, jako základní vlastnost OOP. Prvotně je třeba objektem určit aktivity, které mají provést ještě před tím než jejich požádáme o informace o jejich stavu. Na základě tohoto principu odlišujeme funkce jednotlivých objektů což napomáhá zabránit příliš těsnému propojení mezi objekty, které by mohlo vzniknout.
- **YANGI** (You Ain´t Gonna Need It) - podstatou YANGI principu je použití pouze takové funkcionality, jaká je nezbytně nutná. YANGI je argument, který standardně podávají členové vývojářského týmu, kteří si mysli že architekt řeší některé věci příliš komplikovaně nebo příliš nákladně. Toto napomáhá vytvářet mnohem optimalizovanější aplikace.
- **TDD** (Test-Driven Development) - je metoda vývoje softwaru při které je zdrojový kód opakovaně testován. Tento koncept vytváří z něčeho fungujícího něco dokonalé. Po každém testu je zdrojový kód prokopán a optimalizován, následně je spuštěn stejný nebo podobný test, přičemž tyto testy opakují znovu a znovu, pokud je to nezbytné pro dosažení optimálního řešení. TDD je schopen produkovat aplikace nejvyšší kvality v mnohem kratším čase než jiné metody. (8)

- **SOC** (Separation of Concerns) - je nejpodstatnější princip při vývoji softwaru. Podstatou tohoto principu je rozložit aplikaci (software) na co nejmenší části, které zapouzdřují data a chování, přístupné dalším třídám. Což zajišťuje vysokou dostupnost při udržování a testování kódu.

6.3 Programové rámce v PHP

Zkratka PHP kdysi zastupovala "Personal Home Page", ale dnes tato zkratka zastupuje "Hypertext Preprocessor", což je skriptovací jazyk. Tento programovací jazyk je především určen pro tvorbu dynamických webových stránek. Nejčastěji se používá tak že kód jazyka PHP je přímo začleněn do HTML nebo XHTML.

6.3.1 Popis a použití PHP

PHP je vedle ASP. NET a JAVA jedním ze tří nejrozšířenějších skriptovacích jazyků pro webové aplikace. Oblíbeným se stal především díky jednoduchosti použití, bohaté zásobě funkcí a tomu, že kombinuje vlastnosti více programovacích jazyků a nechává tak vývojáři částečnou volnost při tvorbě syntaxe. V kombinaci s operačním systémem Linux, databázovým systémem MySQL nebo PostgreSQL a webovým serverem Apache je velmi často využíván při tvorbě webových aplikací. Pro tuto kombinaci se běžným zkratka LAMP, což značí spojení Linux, Apache, MySQL a PHP. (3)

Pokud se PHP používá pro tvorbu dynamických stránek, jedná se o skripty spouštěné na straně serveru, kde jsou tyto skripty i prováděny a následně produkují výstup, který je předložen konečnému uživateli. Syntaxe jazyka samotného je kombinací více programovacích jazyků jako například C, Perl, JAVA. Tyto jazyky nejsou přímo kopírovány, ale PHP je jimi inspirované. Značné pozitivum pro jazyk PHP je nezávislost na použitém operačním systému. PHP se v dnešní době používá k psaní malých, ale i velkých systémů a je možné jej použít i pro vytváření desktopových aplikací. (3)

6.4 PHP frameworky

Podstatu frameworků je možné definovat jako zbytečné vytváření nové pracovní činnosti, pokud už existuje funkční aplikace frameworku. Framework je v podstatě soubor tříd a aplikací, SDK a API, který pomáhá jednotlivým komponentům při spolupráci. Frameworky šetří programátorům množství času předdefinovaných běžných funkčních částí, a v případě PHP ošetřují některé chyby jazyka. Hlavním záměrem PHP frameworků je splnění určitých kritérií, kterými jsou: **modularita, lehkost, bezpečnost, kompatibilita, efektivita, konfigurovatelnost a customitovatelnost.**

6.4.1 Zend Framework

Zend je nejrozšířenějším frameworkem v PHP. Má velmi širokou základnu uživatelů, a proto je k němu mnoho návodů a různých rad, které se dají velmi rychle a snadno najít. Vyvinula ho společnost Zend Technologies, která se podílela i na vývoji samotného jazyka PHP. Zend je založen na jednoduchosti objektově-orientovaných osvědčených postupech. Jeho hlavním zaměřením je vývoj spolehlivých a bezpečných web 2.0 aplikací a webových služeb. (9)

Celý framework je navržen tak, že programátor si může vybrat pouze jednotlivé části, které potřebuje, což vede ke zrychlení webových aplikací, urychlení vývoje a snížení nákladů na údržbu. Zend využívá moderní technologie, jako jsou AJAX, vyhledávač Lucene a Web Services. Tento framework je licencovaný pod New BSD licencí. (9)

6.4.2 Nette Framework

Již uvedený Nette Framework je výkonný framework, určený pro rychlé a pohodlné vytváření webových aplikací. Je vyvíjen komunitou českých, a také slovenských programátorů. Snahou tohoto frameworku je eliminovat rizika a umožnit opětovnou použitelnost zdrojového kódu. Komunita vývojářů začlenila do tohoto frameworku moderní technologie jako jsou AJAX, SEO, DRY, KISS, MVC, cool URL. Velké množství práce tato odborná komunita rovněž věnovala bezpečnosti a eliminaci bezpečnostních děr. (10)

Velmi přínosnou částí pro programátora je webová ladička, která v případě chyby vytváří výpisy o konkrétní chybě přímo na obrazovku a programátor nemusí hledat tyto chyby v logu. Obsahuje rovněž vlastní šablonovací systém Latte. Dokumentace není až tak rozsáhlá, ale obsahuje popis všech podstatných částí a je dostupná také v českém jazyce, což je nesporá komunikační a obsahová výhoda. Nette Framework je vhodný pro začátečníky, ale i na vývoj větších aplikací. Tento framework je taktéž licencovaný pod New BSD licencí. (8)

6.4.3 Symfony Framework

Hlavním cílem tohoto frameworku je urychlit vytváření a údržbu webových aplikací a nahradit opakující se psaní zdrojového kódu a tím usnadnit programování. Symfony může být použit na jakékoliv konfiguraci serveru, úplně mu postačí Unix nebo Windows s webovým serverem a nainstalovaným PHP. Symfony je kompatibilní téměř s každým databázovým systémem. (11)

Je zaměřen hlavně na vývoj robustních aplikací v podnikovém kontextu. Což znamená, že vývojář mě plnou kontrolu nad konfigurací: od adresářové struktury až po knihovny a pluginy. Téměř vše se dá přesně přizpůsobit koncovým požadavkům. Je také propojen s dalšími nástroji, které pomáhají při testování a ladění. Celý framework je zcela zdarma s MIT licencí.

6.4.4 Cake PHP Framework

Tento framework je spíše vhodný pro menší aplikace, je podstatně jednodušší než Zend nebo Symfony. Je sice jednoduchý, ale i tak je celkově dobře vybaven. Je možné ho rozšiřovat o různé pluginy vytvořené komunitou, ale integrace komponent třetích stran je už obtížnější. Obsahuje jednoduchý ORM nástroj. Chybí mu však šablonovací systém, ale je možné použít jakýkoliv jiný dostupný. Cake PHP je spíše vhodný pro začátečníky. Stejně jako Symfony je distribuován s MIT licencí. (12)

6.5 Komparace PHP Framework a ASP. NET

V prostředí internetu je mnoho odborných článků i diskusí odborné a laické veřejnosti, která je lepší platforma PHP nebo ASP. NET Bohužel většina z těchto

názorů není objektivní, jejich preferování je standardně založeno na propagaci jednoho programovacího jazyka a kritiky toho druhého programovacího jazyka Pokud se zaměříme na zhodnocení dat těchto příspěvků, můžeme vysledovat, že většina z těchto informací je neaktuální. Je to dost nešťastné, protože většina z těchto zastaralých informací se zobrazuje na prvních příčkách ve vyhledávačích.

Pokud jde o výkon, ukážeme si, které faktory ho ovlivňují, a výsledku prokážeme, že upřednostnění jednoho programovacího jazyka před druhým je ve většině případů bezpředmětné. Také si ukážeme, které faktory ovlivňují škálovatelnost, přičemž oba jazyky jsou velmi dobře škálovatelné. Pokud se dostaneme k ceně a podpoře tak při tomto kritériu už bude PHP Framework ve výhodě, protože je to OpenSource a standardně používá platformu LAMP, kterou jsme již zmínili, přičemž ASP. NET, je nákladově náročnější. Z hlediska času potřebného na vývoj, při použití ASP. NET toto zabere přibližně dvakrát více času než při PHP Framewok. (13)

6.5.1 Hodnocení škálovatelnosti a snadnosti údržby

V obecné rovině škálovatelnost a jednoduchost údržby nemá nic společného s tím, která z těchto dvou platforem je lepší, protože primárně jsou odvislé od:

- programátorových zkušeností,
- aplikace "best practices",
- aplikace stabilního a solidního frameworku,
- dodržování směrnic a standardů.

6.5.2 Hodnocení výkonu a rychlosti

O výkonu a rychlosti se vede mnoho diskusí, ale většina z nich je neobjektivní a zaměřena na podporu jednoho z jazyků, namísto toho, aby byli uživatelé relevantně informovali. Existuje mnoho faktorů, které je třeba zohlednit při měření rychlosti webových aplikací, takže rychlost jakéhokoliv programovacího jazyka nemá žádný zřetelný efekt na rychlosti většiny dnešních webových stránek. Pokud programovací jazyk potřebuje provést velké množství úkolů, jako je to u Googlu nebo Yahoo, tak v takovém případě je třeba věnovat nejvíce pozornosti při výběru toho nejrychlejšího programovacího jazyka, aby zvládal provést co největší

množství úkolů v co nejkratším čase – i to je důvod proč tyto stránky používají v kódu svých aplikací více programovacích jazyků, každý z nich zpracovává jen ty procesy, které dokáže zpracovávat nejrychleji. (13)

Většina Linuxových serverů běží bez nějakých extra a nepotřebných balíků nebo GUI rozhraní, proto operační systém využívá jen velmi malou část CPU a RAM, což umožňuje databázi a webovému serveru využívat mnohem více výkonu. Pokud jde o servery Windowsu, tak tyto běží naopak s mnoha spuštěnými balíky a GUI rozhraními, přičemž využívají mnohem více výkonu jen na samotný běžný provoz. Další výhodou LAMP oproti platformě ASP. NET je to, že PHP je nezávislé a je schopné běžet na jakékoliv serverové platformě, jako je Linux, Unix, Mac OS X, Windows, zatímco ASP. NET je výhradně pro Windows platformu. (13)

6.5.3 Hodnocení ceny

PHP Framework, MySQL, Apache server a Linux OS jsou zdarma, a také všechny modifikace jsou také zdarma. Platforma LAMP je velmi populární u hostingových společností, což snižuje cenu hostingu ve srovnání s Windows hostingem. ASP. NET je zdarma, pokud si zakoupíte operační systém Windows, ale jsou tam další náklady spojené s licencemi pro Microsoft Windows Server, Microsoft SQL a pozdější vylepšení. Pro uvedení příkladu: Microsoft Server 2008 R2 Standard – 64bitový stojí přibližně $ 1029 a Microsoft SQL server 2008 standardní edice pro malé společnosti stojí také přibližně $ 1038. Přičemž tato cena se může ještě výrazně zvýšit, pokud se aplikace běžící na této platformě stane populární a bude ji třeba kvůli výkonu rozložit na více serverů.

6.6 Komparace PHP Framework a JAVA

JAVA přináší mnoho výhod jako programovací jazyk. Toto jsou její klíčové výhody:

- Jednoduchost naučit se tento jazyk.
- Objektově orientované programování.
- Nezávislost na platformě.

Tyto výhody představují mnoho pozitivního z této aplikace. JAVU je jednodušší se naučit, psát, kompilovat a debugovat oproti ostatním programovacím jazykům. V JAVĚ je velmi jednoduché vytvářet kód, který lze opakovaně použít, a tak vytvářet i stavebnicové programy. Nejpodstatnější výhodou JAVY je to, že je nezávislá na platformě, na které běží. To znamená, že můžete stejný program použít na různých operačních systémech. (14)

6.6.1 Výhody PHP Framework

Jazyk PHP Framework přináší mnoho výhod, nejpodstatnějšími jsou:

- Jednoduchost naučit se tento jazyk.
- Je ve formě OpenSource.
- Efektivnost při zpracování kódu.
- Rozsáhlá podpora.

Stejně jako JAVA je snadné se naučit programovat v tomto jazyce. PHP Framework je OpenSource což značí, že má širokou komunitní podporu, a pokud je kód napsaný správně, tak se PHP Framework považuje za nejefektivnější programovací jazyk webových aplikací.

6.6.2 Komparace odlišností mezi PHP Framework a JAVA

PHP Framework a JAVA jsou při porovnávání jejich výhod velmi podobné, ale ve skutečnosti jsou velmi rozdílné. PHP je skriptovací jazyk pracující na straně serveru, zatímco JAVA částečně pracuje na straně serveru a částečně na straně klienta. Pokud použijeme jazyk PHP, tak kód se bude provádět na serveru, přičemž při JAVA se většina kódu zpracovává na počítači uživatele. Z čehož vyplývá, že pokud uživatel nemá na počítači správný program jako například JAVA Runtime Enviroment, stránka se jednoduše nezobrazí, což se u PHP stát nemůže. (14)

Většinou záleží na preferencích samotného programátora, který z těchto jazyků si vybere. Hlavním důvodem, proč si programátoři častěji vyberou PHP, je rychlost. Přičemž v tomto pojetí myslím rychlost vývoje samotné aplikace, PHP má také tendenci mít méně problémů při využívání sdíleného hostingu, který je v dnešní

době velmi populární pro střední a menší projekty. Pro JAVU je výhodnější používat dedikovaný server pro hosting. Proto je JAVA vhodnější hlavně pro větší webové aplikace. Oba jazyky jsou široce používané pro vývoj webových aplikací. Někteří vývojáři používají dokonce oba jazyky současně, při vývoji svých aplikací. PHP je jednodušší na přestavbu a přizpůsobení aplikací než JAVA. Také je zdarma, zatímco JAVA je zpoplatněna. Výhodou v případě JAVY je, že studenti tohoto jazyka mohou získat certifikát kvalifikace, přičemž pro PHP Framework neexistuje žádný stupeň kvalifikace.

6.7 Výběr konkrétního řešení

Na webu je dostupné množství různých frameworků, a vývojáři je poměrně často využívají k vytváření svých webových aplikací. Tyto frameworky nabízejí řadu funkcí a tím redukují čas strávený při vytváření webových aplikací a utváří samotnou programátorskou práci jednodušší. Po předchozích srovnáních jednotlivých jazyků mezi sebou, pro naše potřeby vývoje webové aplikace malého až středního rozsahu, jednoznačně vychází nejlépe jazyk PHP Framework s platformou LAMP, protože je zdarma, což výrazně ušetří vstupní náklady. Pokud se podíváme na popis PHP frameworků, jednoznačně vychází nejlepší a nejpoužívanější PHP Framework se širokou uživatelskou základnou, Zend Framework.

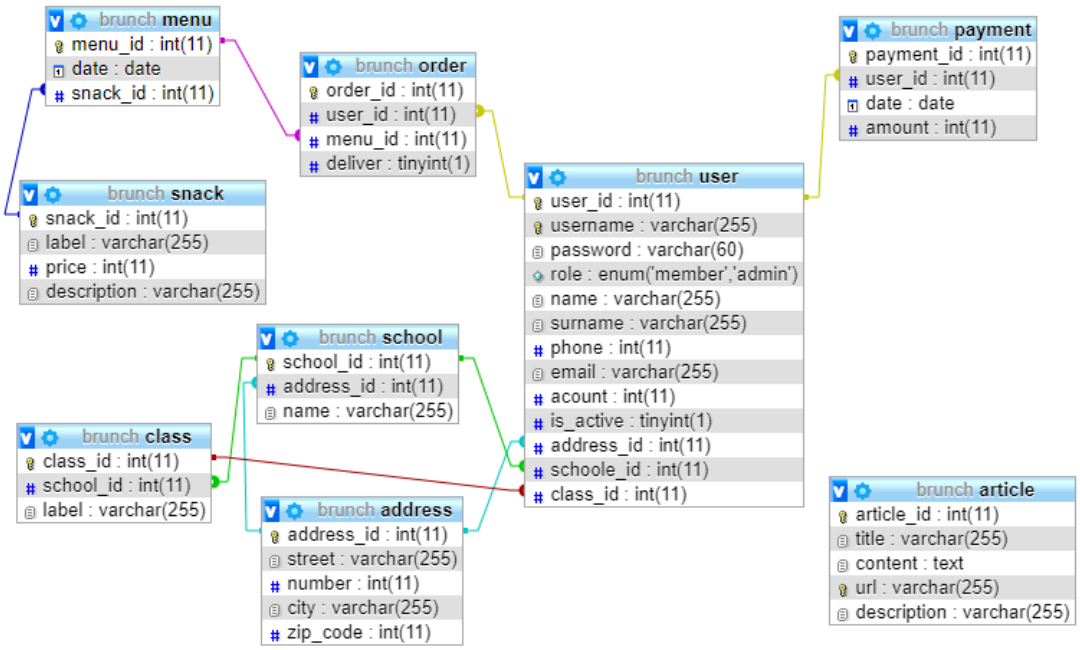
7 Návrh a implementace aplikace

V praktické části si ukážeme implementaci skutečné aplikace za použití Nette frameworku. Jedná se o aplikaci pro firmu Brunch, která vyrábí zdravé svačiny do škol nebo firem. Jejím cílem je, aby si mohli strávníci objednávat svačiny, které jim následně firma rozváží. Do systému je možné se zaregistrovat. Uživatelé mají přiřazeno jedno ze dvou uživatelských úrovní – member a admin. Member má práva pouze na objednávání svých svačin a může kontrolovat stav svého konta. Uživatel admin může vykonávat veškeré administrátorské úkony. Patří tam editace stránek, správa uživatelů či vytváření jídelníčku.

Webová aplikace je vyvíjena v prostředí NetBeans IDE 8.2 a na zprovoznění webového serveru Apache + PHP + MySQL je využit nástroj XAMPP. Technologie, které jsou v projektu použity jsou následující:

- MySQL 5.0.12
- PHP 7.1.7
- Nette 2.4

7.1 Datová struktura



Obrázek 5 - Datový model

Datová struktura obsahuje celkem devět entit: entity pro příspěvky, uživatele, platby, svačiny, jídelníčky, objednávky, adresy, školy a třídy.

Entita pro příspěvky („article“) obsahuje články, které jsou na webu statické. Atribut „url“ obsahuje text na URL adresu přes kterou se na článek odkazuje. Jelikož dva články nesmí mít duplicitní adresu, je tomuto atributu nastaven unikátní klíč. Dále zde jsou atributy „title“ a „description“, které se vnoří do hlavičky stránky. V „content“ je uložen obsah stránky v html formátu.

Uživatel („user“) má účel autorizace a autentizace. Z tohoto důvodu jsou zde atributy „username“, který musí být unikátní, „password“ a „role“. Role jsou v aplikaci tři – admin, member a guest. Do databáze se ale zaznamenávají jen první dva. Guest nemá skoro žádná oprávnění, může pouze zobrazovat články a pro to není potřeba registrace. Další atributy jsou na identifikaci zákazníka – „name“, „surname“, „phone“, „email“. „Address_id“ či „school_id“ by mělo být vždy vyplněno, aby prodejce věděl, kam má svačinu doručit. Poslední atribut je „acount“, kde je evidováno kolik má zákazník peněz na účtu, ze kterých může platit objednávky.

Platba („payment“) slouží k uložení záznamu kdy („date“) a kolik („amount“) si zákazník nahrál peněz na účet.

Svačina („snack“) je určena pro zaznamenání typu svačin, které firma nabízí. Do pole „label“ se zaznamenává identifikační popis svačiny. Každé jídlo má jinou hmotnost a obsahuje různé suroviny, proto má každý jinou cenu - „price“. V „description“ je popsáno co svačina obsahuje za ingredience.

Jídelníček („menu“) je každý den jiný proto na to musí být zvláštní tabulka. Obsahuje ale pouze datum („date“) na kdy menu je a svačinu („snack_id“) kterou obsahuje.

Objednávka („order“) obsahuje informace o zaslaných objednávkách. V atributu „deliver“ se zaznamenává, zda byla již svačina doručena.

7.2 Layout aplikace

Není možné zde zaznamenat všechny obrazovky aplikace, budou zde ale představeny některé nejdůležitější.

Obrázek 6 - Úvodní stránka

Úvodní stránka obsahuje obecné informace. Jak je možné vidět na každé stránce je zobrazeno horní menu, ze kterého lze odkazovat na jednotlivé stránky „o nás“, „svačiny“ a „kontakt“. Také se zde můžete dostat na přihlašovací stránku. V případě,

že je uživatel přihlášen se menu ještě rozšíří o odkaz do administrátorské stránky v případě role admin a vstup do uživatelského profilu v případě role member.

Obrázek 7 - Registrační formulář

Na obrázku 7 můžeme vidět běžný registrační formulář. Je zde použita i kontrola proti spamu.

Obrázek 8 - Jídelníček

V jídelníčku jsou vždy zobrazeny aktuální nabídky jídel. Ty jsou vždy členěny dle jednotlivých dnů. Jak je možno i vidět v menu, tuto stánku mohou zobrazit jen registrovaní uživatelé. Také je tu vyznačeno, jaká jídla má uživatel aktuálně objednaná. V případě, že by uživatel chtěl jídlo odhlásit, stačí pouze odškrtnout zaškrtávací políčko a následně potvrdit formulář.

Obrázek 9 - Výpis článků

Na obrázku 9 je vidět výpis článků, které jsou v aplikaci k dispozici. Do tohoto výpisu může vstoupit pouze uživatel s rolí admin. Zde je možno články spravovat ať už upravovat či mazat.

7.3 Vlastní realizace

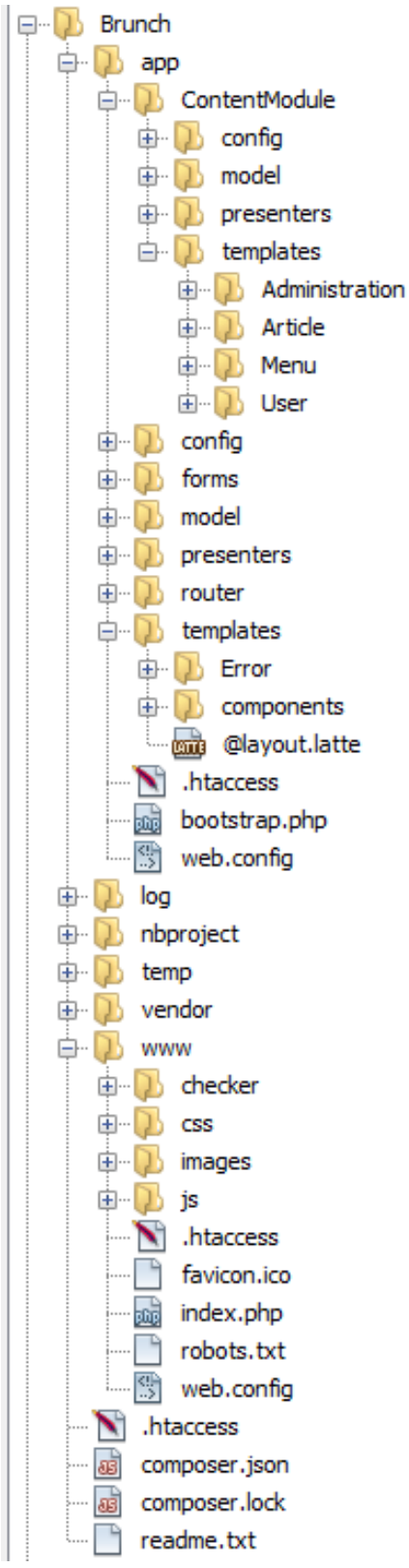
7.3.1 První nastavení

Na začátku bylo potřeba stáhnout Nette Sandbox z oficiálních stránek Nette Frameworku. (15) Po založení nového projektu bylo nutné vymazat některé komponenty. Ponechány byly pouze třídy a formuláře pro autorizaci a autentifikaci. Dále zůstaly soubory na routování URL a konfiguraci. Poté se v rootu projektu vytvořil soubor .htaccess. díky kterému bude aplikace brát složku /www jako root projektu, a tudíž se do URL adresy už nemusí vyplňovat na začátek www. Dalším krokem bylo založení databáze. Údaje o ní se pak zapsaly do konfiguračního souboru.

7.3.2 Adresářová struktura

Do struktury byla přidána část „ContentModule“ obsahující vlastní modely, presentery a šablony. Zde je veškerý obsah používající registrovaní uživatelé. Tento díl má i vlastní konfigurační soubor.

Mimo tuto část jsou složky pro obsah obecného rázu. Patří tam složky pro model, presentery a šablony ke kódu, který bude využit ve vícero komponent. Složka config obsahuje dva soubory „config.neon“ a „config.local.neon“, kde jsou údaje k databázi. Adresář router má soubor „RouterFactory“ ve kterém je nastavené routování URL.



Obrázek 10 - Adresářová struktura

7.3.3 Modely

Obvykle se pro každou entitu v databázi používá nový model. V této aplikaci se využívají modely pro články, uživatele, jídelníček a objednávky. Pár z nich si zde představíme. Všechny modely v aplikaci dědí vlastnosti od „BaseManageru“. Tato třída dědí z Nette\Object a obsahuje napojení na databázi.

```php
use App\Model\BaseManager;
use Nette\Database\Table\IRow;
use Nette\Database\Table\Selection;
use Nette\Utils\ArrayHash;

/** Třída poskytuje metody pro správu článků v redakčním systému ...4 lines */
class ArticleManager extends BaseManager {

    /** Konstanty pro manipulaci s modelem*/
    const
            TABLE_NAME = 'article',
            COLUMN_ID = 'article_id',
            COLUMN_URL = 'url';

    /** Vrátí seznam článků v databázi ...4 lines */
    public function getArticles() {
        return $this->database->table(self::TABLE_NAME)->
                order(self::COLUMN_ID . ' DESC');
    }

    /** Vrátí článek z databáze podle jeho URL ...5 lines */
    public function getArticle($url) {
        return $this->database->table(self::TABLE_NAME)->
                where(self::COLUMN_URL, $url)->fetch();
    }

    /** Uloží článek do systému ...4 lines */
    public function saveArticle($article) {
        if (!$article[self::COLUMN_ID])
            $this->database->table(self::TABLE_NAME)->insert($article);
        else
            $this->database->table(self::TABLE_NAME)->
                where(self::COLUMN_ID, $article[self::COLUMN_ID])->
                update($article);
    }

    /** Odstraní článek ...4 lines */
    public function removeArticle($url) {
        $this->database->table(self::TABLE_NAME)->
                where(self::COLUMN_URL, $url)->delete();
    }
```

Obrázek 11 - ArticleManager.php

Nejprve se vytvořily konstanty pro entitu článku, mezi něž patří název tabulky, id článku a URL článku. Tyto konstanty se později budou používat pro selekci dat z databáze. Je výhodné je používat z důvodu případných změn v budoucnu. Když bychom měnili název atributu, tak v aplikaci to můžeme opravit jen na jednom místě.

První metoda v modelu je „getArticles()“. Za pomocí selectoru nám vrátí seznam všech článků. Tyto položky budou seřazeny sestupně podle id.

Druhá metoda je „getArticle($url)“, která přijímá vstupní paramatr url. Díky tomuto parametru se vybere jeden článek. Vzhledem k tomu, že url je atribut, který nemůže mít duplikát víme, že se vybere jen jeden článek.

Třetí metoda „saveArticle($article)“ ukládá celý článek. Nejprve je provedena kontrola, zda není již článek vytvořen. V případě že není žádný nalezen, se dle vstupního parametru vloží nový. Jestliže je nějaký nalezen, tak se pouze uloží aktualizace.

Poslední metoda „removeArticle($url)“ článek maže. Opět je vstupní parametr url, dle kterého se najte položka, která se má odstranit.

```php
<?php

namespace App\CoreModule\Model;

use App\Model\BaseManager;
use Nette\Database\Table\IRow;
use Nette\Database\Table\Selection;
use Nette\Utils\ArrayHash;

/** Třída poskytuje metody pro správu článků v redakčním systému ...4 lines
class MenuManager extends BaseManager
{
        /** Konstanty pro manipulaci s modelem. */
        const
                TABLE_NAME = 'menu',
                COLUMN_ID = 'menu_id',
                COLUMN_DATE = 'date',
                COLUMN_SNACK = 'snack_id';

        /** Vrátí seznam článků v databázi ...4 lines */
        public function getMenus()
        {
                return $this->database->table(self::TABLE_NAME)->
                        order(self::COLUMN_ID . ' DESC');
        }

        //načte aktuální jídelníček
        public function getActualMenus()
        {
            return $this->database->table(self::TABLE_NAME)->
                    where(self::COLUMN_DATE . '>= ?', 'CURDATE()')->
                    order(self::COLUMN_DATE.','. self::COLUMN_SNACK);
        }
```

Obrázek 12 - MenuManager.php

V modelu pro jídelníček si opět nejprve nadefinujeme konstanty. Potřebujeme zde název tabulky, id jídelníčku, datum a id svačiny.

První použitá metoda je zde „getMenus()“. Vrací nám všechny jídelníčky srovnané sestupně dle jejich id.

Druhá metoda „getActualMenus()“ také vrací výpis jídelníčku, ale jen pro budoucí data. Tj. ty co už proběhly mít v seznamu nebudeme. Celý seznam bude srovnaný podle data a následně podle id svačiny.

7.3.4 Presentery

Jako základ pro všechny presentery je vy tvořena třída „BasePresenter“. Zde se kontroluje zda má uživatel pro určitou akci oprávnění.

```php
/** Vykreslí seznam článků do šablony. */
public function renderList()
{
	$this->template->articles = $this->articleManager->getArticles();
}

/** Odstraní článek ...4 lines */
public function actionRemove($url)
{
	$this->articleManager->removeArticle($url);
	$this->flashMessage('Článek byl úspěšně odstraněn.');
	$this->redirect(':Core:Article:list');
}

/** Vykresluje editaci článku podle jeho URL ...4 lines */
public function actionEditor($url)
{
	// Pokud byla zadána URL, pokusí se článek načíst a předat jeho hodnot
	if ($url) ($article = $this->articleManager->getArticle($url))
		? $this['editorForm']->setDefaults($article)
		: $this->flashMessage('Článek nebyl nalezen.');
}
```

Obrázek 13 - ArticlePresenter.php

V presenteru pro články je první metoda „renderList()“. Jedná se o podpůrnou metodu pro vykreslení šablony seznamu článků. Zde se naplňují všechny atributy, které šablona potřebuje pro vykreslení. Je zde využita metoda z modelu getArticles().

Druhá metoda je „actionRemove($url)“. Zde se za pomocí předaného parametru url odstraní článek, následně se uloží zpráva pro vypsání, jaká akce byla provedena, a nakonec se stránka přesměruje na list článků.

Další metoda „actionEditor($url)“ zkoumá, jestli existuje článek dle url. Jestliže článek je, předá jeho detaily editačnímu formuláři. V případě, že neexistuje uloží se zpráva, že článek nebyl nalezen.

```php
/** Vrátí formulář pro editor článků ...4 lines */
protected function createComponentEditorForm()
{
        $form = new Form;
        $form->addHidden('article_id');
        $form->addText('title', 'Titulek')->setRequired();
        $form->addText('url', 'URL')->setRequired();
        $form->addText('description', 'Popisek')->setRequired();
        $form->addTextArea('content', 'Obsah');
        $form->addSubmit('submit', 'Uložit článek');
        $form->onSuccess[] = [$this, 'editorFormSucceeded'];
        return $form;
}
```

Obrázek 14 - Definice formuláře pro úpravu článku

Metoda „createComponentEditorForm()“ vyrobí instanci třídy Form. Tento formulář naplní formulářovými prvky s validacemi. Tyto validace se ukáží jak u klienta, tak i na serveru.

```php
/** Funkce se vykonaná při úspěsném odeslání formuláře; zpracuje hodnoty formuláře
public function editorFormSucceeded($form, $values)
{
        try {
                $this->articleManager->saveArticle($values);
                $this->flashMessage('Článek byl úspěšně uložen.');
                $this->redirect(':Core:Article:', $values->url);
        } catch (UniqueConstraintViolationException $ex) {
                $this->flashMessage('Článek s touto URL adresou již existuje.');
        }
}
```

Obrázek 15 - Metoda pro zpracování odeslaného formuláře

Metoda „editorFormSucceeded($form, $values)“ má za úkol zpracovat hodnoty poslané z formuláře. Nejprve se pokusí článek uložit. Když vše projde v pořádku, načte se hláška o úspěchu a stránka se přesměruje na detail článku. V případě chyby se vyhodí výjimka a načte se hláška o neúspěchu.

```php
public function renderDefault($url)
{
        $this->template->menus = $this->menuManager->getActualMenus();
        $date = DateTime::from(date('Y-m-d'));
        $date->modify('-1 day');
        $this->template->date = $date->format('Y-m-d');
        $identity = $this->getUser()->getIdentity();
        if ($identity) $this->template->orders = $this->orderManager->
                getOrdersByUser($identity->getData()['user_id']);
        $this->template->checked = '';

}
```

Obrázek 16 - MenuPresenter.php

Dále je zde presenter pro jídelníček. Ten obsahuje metodu „renderDeault($url)“. Zde se načítají různé proměnné, aby šablona správně fungovala. Nejprve se načte seznam aktuálních jídelníčků a poté se načte seznam všech objednávek přihlášeného uživatele.

7.3.5 Formuláře

```php
private function login($form, $instructions, $register = false) {
    $presenter = $form->getPresenter(); // Získej presenter ve kterém je formulář umístěn.
    try {
        // Extrakce hodnot z formuláře.
        $username = $form->getValues()->username;
        $password = $form->getValues()->password;

        // Zkusíme zaregistrovat nového uživatele.
        if ($register)
            $this->user->getAuthenticator()->register($username, $password);
        $this->user->login($username, $password); // Zkusíme se přihlásit.
        // Pokud jsou zadány uživatelské instrukce a formulář je umístěn v presenteru.
        if ($instructions && $presenter) {
            // Pokud instrukce obsahují zprávu, tak ji pošli do příslušného presenteru.
            if (isset($instructions->message))
                $presenter->flashMessage($instructions->message);

            // Pokud instrukce obsahují přesměrování, tak ho proveď na příslušném presenteru.
            if (isset($instructions->redirection))
                $presenter->redirect($instructions->redirection);
            elseif ($this->user->isInRole('admin') && isset($instructions->redirectionAdmin))
                $presenter->redirect($instructions->redirectionAdmin);
            elseif ($this->user->isInRole('member') && isset($instructions->redirectionMember))
                $presenter->redirect($instructions->redirectionMember);
        }
    } catch (AuthenticationException $ex) { // Registrace nebo přihlášení selhalí.
        if ($presenter) { // Pokud je formulář v presenteru.
            $presenter->flashMessage($ex->getMessage()); // Pošli chybovou zprávu tam.
            $presenter->redirect('this'); // Proveď přesměrování.
        } else { // Jinak přidej chybovou zprávu alespoň do samotného formuláře.
            $form->addError($ex->getMessage());
        }
    }
}
```

Obrázek 17 - Přihlašovací funkce

Zde je vytvořena třída obsahující tzv. továrničku na formuláře uživatele. Vytváří se z důvodu znuvupoužitelnosti na jiném místě.

První metoda je „login($form, $instructions, $register = false)“, která po potvrzení formuláře přihlašuje uživatele. Nejprve se musí načíst presenter, ve kterém je formulář umístěn. Dále se do proměnných načítají hodnoty z formuláře. Když je registrace nastavena na true pokusí se uživatele zaregistrovat a pak ho přihlásit. Jestli byly předány nějaké instrukce, tak se je poté předá presenteru. Vše je ošetřeno tak, když nějaká akce neprojde, vyhodí se výjimka a vypíše se příhodná hláška.

```php
/** Vrací formulář se společným základem ...5 lines */
private function createBasicForm(Form $form = null) {
    $form = $form ? $form : new Form;
    $form->addText('username', 'Jméno')->setRequired();
    $form->addPassword('password', 'Heslo')->setRequired();
    return $form;
}

/** Vrací komponentu formuláře s přihlašovacími prvky a zpracování přihlašování poc
public function createLoginForm($instructions = null, Form $form = null) {
    $form = $this->createBasicForm($form);
    $form->addSubmit('submit', 'Přihlásit');
    $form->onSuccess[] = function (Form $form) use ($instructions) {
        $this->login($form, $instructions);
    };
    return $form;
}

/** Vrací komponentu formuláře s registračními prvky a zpracování registrace podle
public function createRegisterForm($instructions = null, Form $form = null) {
    $form = $this->createBasicForm($form);
    $form->addPassword('password_repeat', 'Heslo znovu')->setRequired()
            ->addRule(Form::EQUAL, 'Hesla nesouhlasí.', $form['password']);
    $form->addText('y', 'Zadejte aktuální rok')->setType('number')->setRequired()
            ->addRule(Form::EQUAL, 'Špatně vyplněný antispam.', date("Y"));
    $form->addSubmit('register', 'Registrovat');
    $form->onSuccess[] = function (Form $form) use ($instructions) {
        $this->login($form, $instructions, true);
    };
    return $form;
}
```

Obrázek 18 - Továrna na přihlašovací formulář

Další metoda je „createBasicForm(Form $form = null)“. Zde se převezme instance třídy Form z parametru nebo se vytvoří nová. Ta se pak naplní formulářovými prvky s validacemi, použitelné pro všechny formuláře. Tj. uživatelské jméno a heslo.

Třetí metoda „createLoginForm($instructions = null, Form $form = null)“ je pro vytvoření formuláře k přihlášení uživatele. Nejprve se zavolá předchozí metoda a pak se načtou ostatní prvky specifické pro tento formulář.

Poslední metoda „createRegisterForm($instructions = null, Form $form = null)“ je podobná jako předchozí, jen se načtou prvky potřebné pro registraci.

7.3.6 Šablony

```
{define title}Výpis článků{/define}
{define description}Výpis všech článků.{/define}
{block content}
<table>
        <tr n:foreach="$articles as $article">
                <td>
                        <h2><a n:href=":Core:Article: $article->url">{$article->title}</a></h2>
                        {$article->description}
                        <br />
                        {if $admin}
                                <a n:href=":Core:Article:editor $article->url">Editovat</a>
                                <a n:href=":Core:Article:remove $article->url">Odstranit</a>
                        {/if}
                </td>
        </tr>
</table>
```

Obrázek 19 - Šablona pro výpis článků

Zde vidíme použití šablony pro výpis článků. Každá položka ze seznamu $articles vypíše svůj titulek s možností na něj odkázat. Také bude moci admin články editovat a mazat.

```
{define title}Jídelníček{/define}
{define description}Jídelníček.{/define}
{block content}

<form n:name=menuForm class=form>
    {for $i = 1; $i <= $menus->count(); $i++}
        {if ($menus[$i]->date > $date)}
            <fieldset>
                <div hidden>{$date = $menus[$i]->date}</div>

                <legend>{$menus[$i]->date->format('Y-m-d')}</legend>
                <table>
        {/if}
                <tr>
                    {foreach $orders as $order}
                        <div hidden>{$checked = 'checked'}</div>
                        {breakIf $order->menu_id == $menus[$i]->menu_id}
                        <div hidden>{$checked = ''}</div>
                    {/foreach}

                <td class="cb">
                    <input type="checkbox" name="{$menus[$i]->date->format('Y-m-d')}"
                           value="{$menus[$i]->menu_id}" {$checked} />
                </td>
                <td class="cb">{$menus[$i]->snack->label}</td>
                <td class="name">{$menus[$i]->snack->description}</td>
                <td class="price">{$menus[$i]->snack->price} Kč</td>
                </tr>
            {if (($i == $menus->count()) || ($menus[$i + 1]->date > $date))}
                </table>
            </fieldset>
            {/if}
    {/for}

    <input type="submit" name="send" value="Odeslat">
</form>
```

Obrázek 20 - Šablona pro jídelníček

V této šabloně se položky ze seznamu $menus člení dle data. Také je zde kontrola, když má uživatel svačinu objednanou, tak bude označená.

8 Závěry a doporučení

Cílem práce bylo popsat Nette Framework a porovnat ho s podobnými řešeními. V první části bylo nastíněno, co to jsou frameworky a s jakými problémy se u nich setkáváme. Ve druhé části byly porovnány různé architektury frameworků. Byly popsány výhody i nevýhody OpenSource. Zjistilo se, jak funguje architektura MVC a XML. Také byl více přiblížen námi požadovaný Nette Framework. Ve třetí části se důkladně porovnaly různé typy frameworků. Nejprve se popsal ASP. NET Framework a jeho návrhové vzory. Dále se srovnaly PHP frameworky. Byly vybrány ty nejznámější jako Zend Framework, Nette Framework, Symfony Framework a Cake PHP Framework. Poté bylo, ale ještě potřeba porovnat frameworky napříč mezi jednotlivými jazyky. Srovnávaly se PHP frameworky spolu s ASP. NET a PHP framework a JAVA. Zajímalo se o cenu, výkon, rychlost a škálovatelnost. Nakonec z tohoto porovnání vyšlo nejlépe použití PHP frameworku. Je zdarma a na menší a střední aplikace dostatečně stačí. Z těchto zase nejlépe vychází Zend Framework, ale ani náš cílený Nette Framework za ním nijak moc nepokulhává.

Nette Framework byl použit v praktické části práce. Byla vytvořena aplikace pro objednávání svačin. Několikrát byly použity formuláře, které byly díky Nette velmi snadno použitelné a stejně tak validace k nim. Mile mě překvapil snadný přístup k databázi. Vývoj byl s tímto frameworkem velice intuitivní. Častokrát mi byly nápomocny nástroje pro ladění kódu. Bylo znát, že je vyvinut tak, aby i v případě vlastní chyby, byla snadno sjednána náprava. Rozhodně s ním byla práce mnohem rychlejší.

Cílem bylo vytvořit znovupoužitelnou, bezpečnou a snadnou aplikaci, a to se dle mého názoru povedlo. Myslím, že se vývojáři nemají vůbec za co stydět. Určitě se Nette s konkurenčními světovými frameworky může oprávněně srovnávat.

Seznam použité literatury

1. **GROVE, Ralph F.** *Web Based Application Development.* : Jones & Bartlett Learning, 2009. stránky 34-39. ISBN 9780763759407.

2. **HOLLOWAY, J.** Web application Framework - An Overview. University of Wales Aberstwyth, 2003. stránky 24-27.

3. **BRAMPTON, M.** *PHP 5 CMS Framework Development - 2nd Edition.* Birmingham, U.K : Packt Publishing Ltd, 2010. stránky 16-20. ISBN 9781849511353.

4. **MUNRO, J.** *20 Recipes for Programming MVC 3 Faster, Smarter Web Development.* Sebastopol : O'Reilly Media, Inc., 2011. stránky 41-45. ISBN 9781449317652.

5. **Stackoverflow. [Online] [Citace: 14. 08 2017.] https://stackoverflow.com/.**

6. Microsoft. ASP.NET. [Online] Microsoft, 2017. [Citace: 14. 08 2017.] http://www.asp.net/.

7. MILLETT, S. ***Professional ASP.NET Design Patterns.*** **Indianapolis, IN : Wiley, 2010. stránky 18-22. ISBN 978-0-470-29278-5.**

8. BECK, K. ***Test-driven development.*** **Boston : Addison-Wesley, c2003. stránky 32-34. ISBN 978-03211-4653-3.**

9. ***Zend Framework.*** **[Online] Zend Technologies Ltd., 2017. [Citace: 13. 08 2017.] http://framework.zend.com/.**

10. ***Nette Framework.*** **[Online] Nette Foundation, 2017. [Citace: 14. 08 2017.] http://nette.org/.**

11. ***Symfony.*** **[Online] Sensio Labs, 2017. [Citace: 13. 08 2017.] http://symfony.com/ .**

12. ***CakePHP - Build fast, grow solid.*** **[Online] Cake Software Foundation, Inc, 2017. [Citace: 14. 08 2017.] https://cakephp.org/.**

13. KOHAN, B. *Comentum.* [Online] Comentum Corp., 2017. [Citace: 12. 08 2017.] http://www.comentum.com/.

14. LANGE, F. *Codecentric.* [Online] Codecentric AG, 2008. [Citace: 14. 08 2017.] https://blog.codecentric.de/en/2008/07/comparison-of-java-and-php-for-web-applications/.

15. *Nette Framework.* [Online] Nette Foundation, 2017. [Citace: 14. 08 2017.] https://nette.org/cs/download.

Seznam obrázků

Zadání práce

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2015/2016

Studijní program: Aplikovaná informatika
Forma: Prezenční
Obor/komb.: Aplikovaná informatika (ai3-p)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Gavlasová Lucie	E. Beneše 1433/1A, Hradec Králové - Moravské Předměstí	I1301118

TÉMA ČESKY:

Nette framework

TÉMA ANGLICKY:

Nette framework

VEDOUCÍ PRÁCE:

Ing Jiří Štěpánek - KIT

ZÁSADY PRO VYPRACOVÁNÍ:

Cílem práce je zjistit funkčnost PHP frameworku Nette a porovnat výhody a nevýhody s dalšími frameworky. Prakticky si ho budeme demonstrovat při návrhu reálné webové aplikace.

Osnova práce:
I. Úvod
II. PHP frameworky
III. Nete framework
IV. Návrh a implementace aplikace
V. Závěr

SEZNAM DOPORUČENÉ LITERATURY:

Zdroják [online]. 2009 [cit. 2015-10-14]. Dostupné z: https://www.zdrojak.cz/serialy/zaciname-s-nette-framework/

Nette [online]. 2011 [cit. 2015-10-14]. Dostupné z: https://doc.nette.org/cs/2.2/

ITnetwork [online]. 2015 [cit. 2015-10-14]. Dostupné z: http://www.itnetwork.cz/php/nette

Podpis studenta: ..... **Datum:** 19.10.2015

Podpis vedoucího práce: ..... **Datum:** 19.10.2015