**Česká zemědělská univerzita v Praze**

**Provozně ekonomická fakulta**

**Katedra informačního inženýrství**


Bakalářská práce

# Frameworky pro tvorbu webových aplikací

Martin Štěpař

Česká zemědělská univerzita v Praze | Provozně ekonomická fakulta

Katedra informačního inženýrství | Akademický rok 2010/2011

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

**Martin Štěpař**

obor Informatika

Vedoucí katedry Vám ve smyslu Studijního a zkušebního řádu ČZU v Praze čl. 16 určuje tuto bakalářskou práci.

Název práce: **Frameworky pro tvorbu webových aplikací**

## Osnova bakalářské práce:

0. Úvod
0. Cíl práce a metodika
1. Literární rešerše - úvod do problematiky, seznámení s vývojem pro web a frameworkovými technologiemi pro web
2. Seznámení s vybraným frameworkem
3. Analýza a implementace ukázkové aplikace
4. Závěr
5. Seznam použitých zdrojů
6. Přílohy

Rozsah hlavní textové části: 30 - 40 stran

Doporučené zdroje:

McArthur K.: Pro PHP: Patterns, Frameworks, Testing and More. Apress 2008. ISBN 1590598199, 9781590598191

Beck K.: Refaktoring: zlepšení existujícího kódu. Grada Publishing a.s. 2003. ISBN 8024702991, 9788024702995

Kosek, J.: PHP - tvorba interaktivních internetových aplikací.Vydání první. Praha: Grada Publishing 1999. ISBN 80-7169-373-1

Lavin P.: PHP objektově orientované, GRADA, 2009, 978-80-247-2137-8

Nette Foundation, Nette Framework - dokumentace, URL: http://nettephp.com/cs/dokumentace

jQuery Project, jQuery Documentation, URL: http://docs.jquery.com/

Vedoucí bakalářské práce: **Ing. Jiří Brožek**

Termín odevzdání bakalářské práce: březen 2011

..... Vedoucí katedry

..... Děkan

V Praze dne: 28. 3. 2011

# Čestné prohlášení

Prohlašuji, že svou bakalářskou práci „Frameworky pro tvorbu webových aplikací“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou v práci citovány a jejichž výčet je uveden v seznamu použitých zdrojů na konci práce.

Jako autor práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 27. března 2011 …….

# Poděkování

Na tomto místě bych rád poděkoval vedoucímu mé bakalářské práce Ing. Jiřímu Brožkovi za jeho ochotu, cenné připomínky a čas, který mi věnoval.

Dále pak chci poděkovat Davidu Grudlovi za jeho přínos české PHP komunitě a za mnoho užitečných rad a zkušeností z oblasti vývoje, které mi poskytl.

Poděkování zasluhuje i můj kolega Jiří Dolejš za přínosnou kritiku mé bakalářské práce z pohledu profesionálního odborníka.

# Frameworky pro tvorbu webových aplikací

## Souhrn

Bakalářská práce se zabývá tvorbou webových aplikací za pomoci specializovaných frameworků. V rešerši popisuje principy fungování internetu, možnosti jeho využití pro aplikace a nástroje, které lze k tomuto použít. Také zmiňuje specifika vývoje.

V praktické části byla navržena a realizována ukázková aplikace s využitím frameworků Nette Framework a jQuery.

## Klíčová slova

Webové frameworky, MVC, MVP, PHP, Nette Framework, jQuery

## Summary

This bachelor thesis deals with creating web applications using specialized frameworks. In background research describes the principles of the Internet, the possibilities of its use for applications and tools that can be used to this. It also mentions the specificity of development.

In the practical part was designed and implemented a demo application using frameworks Nette Framework and jQuery.

## Keywords

Web frameworks, MVC, MVP, PHP, Nette Framework, jQuery

# Obsah

1 Úvod ..... 10

2 Cíl práce a metodika ..... 11

3 Literární rešerše - úvod do problematiky, seznámení s vývojem pro web a frameworkovými technologiemi pro web ..... 12

3.1 Internet ..... 12

3.1.1 Webová stránka ..... 12

3.1.2 Webová aplikace ..... 12

3.1.3 Komunikace ..... 13

3.2 Platformy, jazyky a frameworky ..... 14

3.2.1 Primárně na straně serveru ..... 14

3.2.2 Primárně na straně klienta ..... 15

3.3 Specifika vývoje pro web ..... 16

3.3.1 Bezstavový protokol ..... 16

3.3.2 Koncové zařízení ..... 17

3.3.3 Výkon uživatelova počítače ..... 18

3.3.4 Závislost na internetu ..... 18

4 Seznámení s vybraným frameworkem ..... 20

4.1 Historie ..... 20

4.2 Úvod do Nette Framework ..... 20

4.3 Součásti frameworku ..... 20

4.3.1 Aplikace ..... 20

4.3.2 Formuláře ..... 23

4.3.3 Šablony ..... 24

4.3.4 Laděnka ..... 25

4.3.5 Ostatní ..... 29

4.4 Komunita a vývoj ..... 29
5 Analýza a implementace ukázkové aplikace ..... 31
5.1 Zadání ..... 31
5.1.1 Technické detaily ..... 31
5.2 Analýza domény a tvorba modelu ..... 32
5.2.1 Modelová vrstva ..... 32
5.3 Tvorba uživatelského rozhraní ..... 33
5.3.1 Wireframy ..... 33
5.3.2 Bez JavaScriptu ..... 34
5.3.3 S JavaScriptem ..... 34
5.4 Tvorba webové aplikace ..... 35
5.4.1 Struktura aplikace ..... 35
5.4.2 Presentery ..... 36
5.4.3 Šablony ..... 37
5.4.4 Model ..... 38
6 Závěr ..... 39
7 Seznam použitých zdrojů ..... 40
7.1 Tištěné zdroje ..... 40
7.2 Internetové zdroje ..... 40
8 Přílohy ..... 41
8.1 Obsah přiloženého CD s aplikací ..... 41

# Seznam obrázků

3.1 Komunikace klient-server pomocí protokolu HTTP ..... 13
4.1 Životní cyklus presenteru ..... 22
4.2 Ukázka části jednoduchého formuláře v Nette ..... 23

4.3 HTML kód bez použití šablony ..... 25
4.4 HTML kód s použitím šablonovacího jazyka ..... 25
4.5 Standardní výpis výjimky v PHP ..... 26
4.6 Výpis výjimky Laděnkou ..... 27
4.7 Výjimka zobrazená při AJAXovém požadavku. ..... 28
4.8 DebugBar ..... 29
5.1 Diagram entit modelové vrstvy ..... 32
5.2 Wireframe úvodní stránky aplikace s vybranou první záložkou ..... 33
5.3 První nakódovaný design aplikace ..... 34

# 1 Úvod

V dnešní době je internet nesdílnou součástí našeho života. Nespočet webových stránek či aplikací nám nabízí služby všeho druhu, jaké si jen umíme představit. Na internetu můžeme nakupovat v tzv. eshopech, plánovat si dovolenou pomocí interaktivních map, získávat informace ze zpravodajských serverů, být ve spojení se svými přáteli skrze sociální sítě, sdílet fotky a komentáře a spousty dalšího.

Vytváření takovýchto webů si žádá odpovídající nástroje. Předmětem této bakalářské práce bude základní seznámení se s frameworky – tedy nástroji, které se pro tvorbu těchto webů používají, usnadňují ji a bez níž by bylo vytváření takových služeb mnohem složitější – a následná praktická ukázka implementace takové služby.

# 2 Cíl práce a metodika

Cílem této bakalářské práce je shrnutí problematiky v oblasti tvorby webových aplikací za pomoci frameworků. Práce shrnuje princip webových stránek a internetu, nabízí stručný přehled existujících nástrojů pro vývoj a následně popisem vybraného frameworku představuje možnosti, které mají současní tvůrci internetových aplikací.

Cílem praktické části bude vytvořit jednoduchou webovou aplikaci, na které si předvedeme výše popisované postupy a principy. Zaměříme se na tzv. „best practice“, tj. doporučovaný osvědčený způsob, jakým lze daný problém řešit. Pro tvorbu aplikace využijeme zdarma dostupné frameworky: Nette Framework pro serverovou část řešení a jQuery pro řešení na straně klienta.

# 3 Literární rešerše - úvod do problematiky, seznámení s vývojem pro web a frameworkovými technologiemi pro web

## 3.1 Internet

Internet je celosvětový systém navzájem propojených počítačových sítí založený na protokolu TCP/IP. Uživatelé připojení v této globální síti mohou využívat různých služeb, které nabízí, jako je sdílení souborů, webové stránky, email.

### 3.1.1 Webová stránka

Webová stránka je textový soubor. Obsahuje formátovací značky jazyka HTML (či XHTML). K zobrazení stránky slouží webový prohlížeč, kterému se pomocí výše zmíněných značek říká, jak má stránku vykreslit. Nebo, lépe řečeno – jaký význam má daný prvek ve stránce. Podle toho jej prohlížeč vykreslí. Značek existuje velké množství:

- nadpisy,
- odkazy – slouží k propojení s dalšími stránkami,
- obrázky,
- formuláře – slouží pro získání vstupu od uživatele,
- …

Grafickou podobu webové stránky lze ovlivňovat kaskádovými styly (CSS) a přidávat jim interakci (nejčastěji JavaScriptem).

Tento koncept zde existuje od začátku a jeho podstata zůstala zachována. Jak to souvisí s webovou aplikací?

### 3.1.2 Webová aplikace

Podstatou webové aplikace jsou webové stránky. Z nich se skládá, jimi je prezentována uživateli. Z tohoto pravidla existují výjimky, o kterých se zmíním níže.

Za webovou aplikaci můžeme považovat každou webovou prezentaci, která na základě interakce s uživatelem zpracovává nějaká data. V tomto smyslu můžeme za

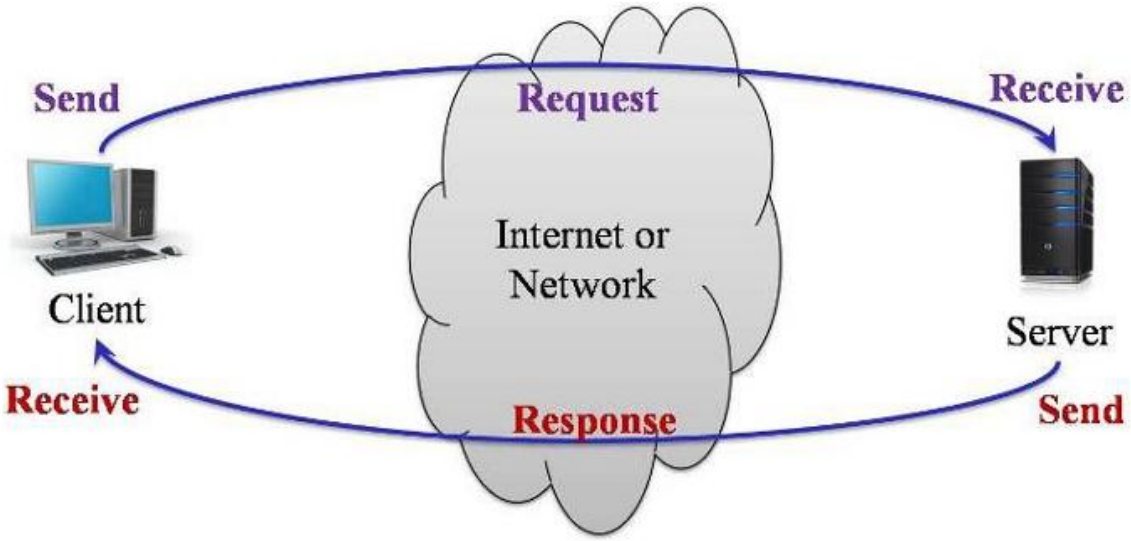
webovou aplikaci považovat osobní stránku s kontaktním formulářem, stejně tak, jako online kancelářský balík. V obou případech je třeba výkonné moci, která zajistí zpracování dat.

Než se pustíme do podrobného popisu, jaké možnosti pro oživení statických webových stránek máme, je třeba se seznámit s principem, jakým lze s webovými stránkami komunikovat.

### 3.1.3 Komunikace

Pro přístup k webovým stránkám se používá protokol HTTP[1]. Tento protokol funguje na bázi klient-prohlížeč. Původně sloužil pouze k přenosu textových dokumentů, v dnešní době umí přenášet také binární soubory.

Komunikace probíhá tak, že klient vyšle na server požadavek a ten vrátí určitou odezvu. Pokud klient uzná za vhodné, může poslat další požadavek – a server opět pošle odpověď. Na jednotlivé požadavky je nahlíženo jako na zcela nezávislé požadavky bez jakékoliv souvislosti – server není schopen určit žádnou vazbu mezi požadavky, byť by přišly ze stejného klienta. Tomuto chování se říká „bezstavovost“ protokolu HTTP (více v kapitole 3.3.1).




3.1 Komunikace klient-server pomocí protokolu HTTP[2]

[1] Hypertext Transfer Protocol – [2]
[2] http://lh5.ggpht.com/_nE94IN7BRj0/S1sQbGeVsGI/AAAAAAAAACw/_uPuAzJ3oaQ/Client-Server+Interaction+-+Sheet2.jpg

Existuje několik dotazovacích metod protokolu HTTP. Pro aplikace jsou důležité především metody:

- GET – slouží pro získávání souborů ze serveru,
- POST – slouží pro zaslání dat na server.

Data je možné přenášet také metodou GET, oproti metodě POST je zde však jeden zásadní rozdíl: v metodě GET jsou data součástí URL, v metodě POST jsou součástí požadavku.

Každý dokument v síti internet má jedinečnou adresu URL[3], podle které jej lze dohledat. Touto adresou se také odkazujeme na jednotlivé webové stránky

Jak tedy webové stránky oživit a vytvořit z nich aplikace?

## 3.2 Platformy, jazyky a frameworky

Základem aplikace je programovací, resp. skriptovací jazyk. Existuje zde mnoho platforem, jazyků a frameworků v nich napsaných. Některé jazyky jsou určeny pro konkrétní prostředí (server nebo prohlížeč), jiné lze nasadit v obou jmenovaných.

### 3.2.1 Primárně na straně serveru

Technologie na straně serveru se začaly používat jako první. Je to pochopitelné, neboť první verze prohlížečů kromě zobrazení HTML stránky nic jiného neuměly. Pojďme si představit nejznámější jazyky a jejich frameworky:

- PHP – jeho počátky se datují do roku 1994. Původně jednoduchý systém psaný v Perlu se postupem času změnil v komplexní skriptovací jazyk pro tvorbu webů. [1] Za jeho masovým rozšířením stojí výrazná jednoduchost v naučení, která bohužel začátečníky často svádí ke psaní tzv. špagetového kódu[4].
    - Nette Framework[5] – český MVP framework, více o něm bude v praktické části

[3] Uniform Resource Locator
[4] Pojem značící takový kód, který je nepřehledný, nestrukturovaný, zamotaný, snažící se řešit všechno najednou. Je velmi náročný na údržbu a výrazně zvyšuje možnost výskytu chyby.
[5] http://nette.org/cs/

    - Zend Framework[6] – velmi kvalitní zahraniční MVC Framework, obsahuje velké množství částí a pluginů pro služby třetích stran
    - Doctrine 2[7] – ORM framework vycházející z pojetí pěti vrstvého modelu [3]
- .NET – platforma od firmy Microsoft. Zahrnuje v sobě podporu více programovacích jazyků.
    - ASP.NET[8] – framework vycházející z ASP. Je kompilován a proto je rychlejší oproti klasickým interpretovaným jazykům. Dělí se na WebForms a MVC verzi.
- Ruby – jazyk přenositelný mezi OS. Jak říká jeho autor: „Ruby je jednoduchý na pohled, ale uvnitř je velmi komplexní – stejně jako lidské tělo.“ [4]
    - Ruby on Rails – oblíbený MVC framework, využívá jej mnoho známých aplikací: Github, Twitter
- Python – hybridní jazyk (umožňuje psát objektově, procedurálně i funkcionálně)
    - Django[9] – model je zde základem aplikace. Z definovaných modelových tříd se generují databázové tabulky a následně i administrační rozhraní.

### 3.2.2 Primárně na straně klienta

Zde máme dvě rozdílné technologie:

- skriptovací
- určující vzhled

Nejprve se podívejme na vzhled. Ten se u webových stránek definuje kaskádovými styly (CSS). Jde o obyčejné textové soubory. Protože je CSS docela upovídané a není vždy úplně jednoduché dosáhnout požadovaného vzhledu napříč prohlížeči, vznikly některé frameworky, které umožňují jednodušší zápis či přednastavené layouty.

[6] http://framework.zend.com/
[7] http://www.doctrine-project.org/projects/orm
[8] http://www.asp.net/
[9] http://www.djangoproject.cz/

- LESS[10] – poskytuje zjednodušený zápis CSS, který následně kompiluje do skutečného CSS. Lze využít taky verzi, která kód převádí v reálném čase pomocí JavaScriptu.
- Blueprint[11] – usnadňuje práci s typografií, umožňuje zarovnávat prvky do mřížky,…

Do první jmenované kategorie zahrneme nástroje, které umožňují naprogramovat chování aplikace:

- JavaScript – dnes standardní součástí každého prohlížeče
  - jQuery[12]
  - Prototype[13]

  Oba zmíněné frameworky usnadňují práci s JavaScriptem, vylepšují objektovou podporu. Oba se také pyšní velkou sbírkou praktických doplňků.
- Flash[14] – technologie firmy Adobe, vyžaduje plugin, lze pomocí něj vyvíjet i aplikace nezávislé na webu
- Silverlight[15] – obdoba Flashe od firmy Microsoft

## 3.3 Specifika vývoje pro web

Vývoj webových aplikací se od vývoje aplikací desktopových do jisté míry liší. Je to dáno různými specifiky daných platforem. Zásadní pro vývoj jsou:

### 3.3.1 Bezstavový protokol

Jak bylo popsáno výše, všechny součásti webové stránky se přenáší protokolem HTTP. Tento protokol je bezstavový. To znamená, že si ani jedna ze stran komunikace (prohlížeč a server) nepamatuje předchozí požadavky.

[10] http://lesscss.org/
[11] http://www.blueprintcss.org/
[12] http://jquery.com/
[13] http://www.prototypejs.org/
[14] http://www.adobe.com/
[15] http://silverlight.net/

V praxi to vypadá tak, že u webové aplikace se server vždy zeptá: „tebe neznám, jsi přihlášený“?

Oproti tomu, desktopová aplikace běží v paměti uživatelova počítače a od svého spuštění v ní prakticky „žije“ až do svého ukončení. Ví tak přesně, že se uživatel přihlásil, že je přihlášený - a tuto informaci ví neustále, dokud se uživatel neodhlásí či aplikaci nevypne.

Samozřejmě, tento problém má různá řešení. Lze použít parametry v URL adresách, lze pojmout celou stránku jako formulář, nebo využít tzv. „cookies“ - malé soubory informací, které si může aplikace s klientem vyměňovat a v jisté míře si tak označit klientovu identitu, která se bude přenášet napříč všemi požadavky. Každý z těchto způsobů má své výhody a nevýhody, záleží také, jaký způsob je preferovaný na použité platformě.

### 3.3.2 Koncové zařízení

Vyvíjíme-li aplikaci pro desktop, s velkou pravděpodobností budeme dobře obeznámeni s možnostmi platformy, na které poběží. Ve většině případů to bude konkrétní operační systém, výjimečně půjde o přenositelnou aplikaci – v takovém případě je počet možných OS limitován (prakticky jde o systémy Windows firmy Microsoft, systémy založené na UNIXu nebo systémy firmy Apple). Pravidlem ovšem bývá, že cílové prostředí je dopředu známo.

Oproti tomu u webové aplikace nemáme nikdy jistotu, v jakém zařízení a prohlížeči bude spuštěna, resp. zobrazena. Výjimkou jsou intranetové aplikace, ale těmi se nyní zabývat nebudeme.

Protože jsou webové aplikace dostupné přes internet, lze očekávat návštěvníky s

- libovolným prohlížečem (liší se podporou standardů, funkcí, kompatibilitou),
- různým rozlišením (stolní PC vs. mobilní zařízení),
- odlišným nastavením (vypnuté cookies, blokované stahování souborů,…),
- …

Všechny tyto detaily je při návrhu aplikace třeba zohlednit, neboť často může mít nefunkčnost webové aplikace i jiné následky, než „jenom“ nespokojeného uživatele. Vyskytnou-li se například takové potíže velké skupině uživatelů, může to mít negativní vliv na tržby webového obchodu a přechod jeho zákazníků ke konkurenci.

S koncovým zařízením souvisí i následující bod:

### 3.3.3 Výkon uživatelova počítače

V dřevních dobách internetu, kdy možnosti webové stránky byly omezeny na velikost písma a barvu pozadí, nehrál výkon uživatelova počítače žádnou roli. S postupným vývojem webových technologií se však tento aspekt změnil. Velkou měrou k tomu přispěl JavaScript, který umožňoval vykonávat kód samotné aplikace přímo v prohlížeči, a dále pak požadavky na obsah – audio, video a interaktivní prvky stránky.

V dnešní době jsou dokonce webové aplikace, které se snaží přenést všechnu svou logiku do prohlížeče. Jako příklad lze uvést službu Google Docs – kancelářský balík fungující v prohlížeči.

S nástupem technologií HTML5 byly do rukou vývojářů dány další nástroje, kterými lze vytvořit v prohlížeči prakticky cokoliv: standardizované značky pro multimediální obsah, canvas pro kreslení (a to i ve 3D), různé hrátky se vzhledem (vlastní písma, stíny,…). O potřebě vyššího výpočetního výkonu svědčí vydání prohlížeče MS Internet Explorer 9, který jako první přichází s plnou hardwarovou akcelerací pro vykreslování.

V tomto bodě se vývoj webových aplikací přibližuje vývoji aplikací desktopových. Přesto je třeba míti na paměti, že chování a vnímání uživatelů v souvislosti s webovými aplikacemi je lehce odlišné. Lze předpokládat, že si nikdo nebude pořizovat nový počítač jen kvůli tomu, aby si nakoupil boty v internetovém obchodě, který se sice pyšní využitím nejnovějších technologií, ale zobrazí si jej pouhých 10 % všech potenciálních zákazníků.

### 3.3.4 Závislost na internetu

Celkem logickým požadavkem na fungování webové aplikace je připojení k internetu. Samotná dostupnost připojení je v dnešní době na velmi dobré úrovni. Existuje nepřeberné množství možností, jak se připojit:

- satelit,
- kabel,
- ADSL,
- bezdrátové připojení WiFi,
- mobilní připojení,
- …

Internet je dostupný prakticky všude, kde je mobilní signál. Lze se připojit z každého zařízení, které má internetový prohlížeč. Přesto zůstává několik nevýhod:

- výrazně proměnlivá kvalita připojení – především u mobilního připojení je rychlost a kvalita připojení oproti jiným způsobům pomalá,
- ukládání dat na vzdáleném serveru – jako nevýhoda z toho důvodu, že v případě výpadku internetu nemáme žádný způsob, jak se k datům dostat.

V souvislosti se závislostí webových aplikací na internetu přišlo HTML5 s novými možnostmi, jak dovolit aplikacím fungovat i bez něj. Jde o speciální aplikační cache či o datové úložiště na straně klienta. Lze si tak představit například již zmíněný kancelářský balík od Googlu, který v případě výpadku internetu uloží data do úložiště v prohlížeči a následně po připojení k internetu data synchronizuje se serverem.

# 4 Seznámení s vybraným frameworkem

## 4.1 Historie

Nette Framework je PHP[16] framework od známého českého vývojáře Davida Grudla. První oficiální zmínka o Nette se datuje do roku 2005 [5], kdy David začal na svém blogu publikovat první články o chystaném frameworku. V roce 2008 byl Framework poprvé představen veřejnosti. Díky šikovné marketingové strategii se u nás vzápětí stal velmi rozšířeným PHP frameworkem. V současné době existuje kolem Nette široká skupina nadšenců a schopných programátorů, kteří také přispívají k jeho zlepšování.

## 4.2 Úvod do Nette Framework

Nette Framework je framework pro tvorbu webových aplikací psaných v jazyce PHP. Je založen na architektuře MVP[17]. Filozofií Nette je efektivní pomoc programátorovi při řešení běžných úkolů, jeho vývoj vychází především z reálných situací, které programátor řeší, než z akademického návrhu architektury.

Framework je volně k dispozici pod New BSD licencí či GNU GPL licencích verze 2 a 3. Lze jej tedy zdarma použít i v komerční sféře.

V nabídce je verze frameworku pro PHP 5.2 a 5.3. V prvním případě je možné vybírat mezi dvěma verzemi – s prefixy a bez nich. Výhoda prefixů spočívá v tom, že zamezují kolizím názvů tříd s jinými knihovnami třetích stran. V případě PHP 5.3 je pouze jediná verze, která používá jmenné prostory. Ty kolizím zabraňují již ze samotné podstaty.

## 4.3 Součásti frameworku

### 4.3.1 Aplikace

Je dobré, pokud celou webovou aplikaci zastřešuje nějaká rozumná a přehledná reprezentace kódu. V dobách, kdy pro PHP neexistovaly frameworky, se všechna logika aplikace psala do úvodního „index.php“ souboru, v lepším případě se některé části kódu oddělily do samostatných funkcí a souborů. Celé webové aplikaci tak chyběl nějaký

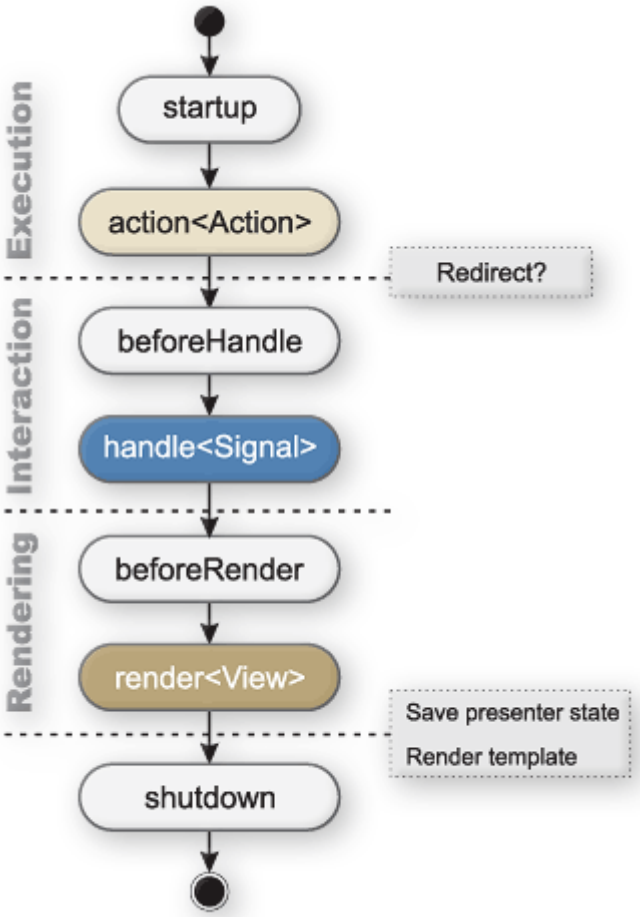
[16] PHP: Hypertext Preprocessor,– [1]
[17] Model-View-Presenter – [6]

ucelený systém, který by zajišťoval všechny činnosti spojené se samotným provozem aplikace.

V Nette existuje třída Application - ta tvoří srdce našeho webu. Zajišťuje přijmutí HTTP požadavku, jeho zpracování a následné odeslání odpovědi.

Přijmutí HTTP požadavku probíhá přes tzv. router. V Nette existuje mechanismus, který (zjednodušeně řečeno) mapuje třídy a jejich metody na URL – tomu se říká routování. V celé aplikace se při vytváření odkazů odvoláváme jen na presenter a jeho akci či signál (viz. níže), konečná podoba adresy potom záleží na nastavení rout. Při přijmutí požadavku udělá router přesný opak – najde k požadované adrese odpovídající presenter, akci a další parametry – a tyto údaje předá aplikaci k dalšímu zpracování.

Každý presenter má svůj životní cyklus. V jeho průběhu jsou volány různé funkce s různými významy. Pro zjednodušení lze říci, že na presenteru je volána nějaká akce, mohou být zpracovány signály, může dojít k vykreslení šablony a následnému odeslání odpovědi (s šablonou, přesměrováním, JSON kódem, souborem ke stažení…).




4.1 Životní cyklus presenteru[18]

Tuto odpověď převezme opět aplikace a odešle ji na výstup v podobě HTTP odpovědi. Tím běh aplikace končí.

Kromě tohoto má aplikace i další funkce. Umožňuje zobrazovat uživatelsky přívětivé chybové hlášky (s možností volby vlastních chybových stránek). Díky tomu, že zpracovaný příchozí požadavek převede do vhodného objektu, je možné jej uchovat například v SESSION a později jej opět vyvolat – toho se využívá například při automatickém odhlášení ze systému[19].

[18] http://files.nette.org/73/lifecycle2.gif
[19] Uživatel odešle formulář, ale protože kvůli dlouhodobé nečinnosti došlo k jeho odhlášení, uloží se požadavek do SESSION a uživatel je přesměrován na přihlašovací stránku. Po úspěšném přihlášení je původní požadavek znovu vyvolán a uživatel pracuje dále tam, kde přestal, a bez ztráty dat.

### 4.3.2 Formuláře

Důležitou součástí webové stránky jsou formuláře. Je to jeden z mála způsobů, jak získat vstupní údaje od uživatele. Dobrý formulář je proto základem celé aplikace.

Formulář se skládá ze dvou částí – jeho reprezentace na webové stránce, tj. kód (X)HTML, a obslužného kódu skriptu, který data přijatá z formuláře zpracuje. Takový skript musí umět spoustu věcí – správně ošetřit příchozí data, validovat je podle stanovených pravidel, předvyplnit políčka na stránce zadanými/uloženými hodnotami, upozornit uživatele na chyby. Na straně prohlížeče by mohl být pro větší uživatelovo pohodlí kód doplněn o JavaScriptovou validaci.

```
$form = new Form;
// …
$countries = array(
    '--- vyberte ---',
    'Europe' => array(
        'CZ' => 'Česká republika',
        'SK' => 'Slovensko',
        'GB' => 'Velká Británie',
    ),
    'CA' => 'Kanada',
    'US' => 'Spojené státy',
    '?'  => 'jiný',
);
// …
$form->addCheckbox('send', 'Odeslat poštou');

$form->addSelect('country', 'Stát:', $countries)
    ->skipFirst()
    ->addConditionOn($form['send'], NForm::EQUAL, TRUE)
        ->addRule(NForm::FILLED, 'Vyberte svůj stát');

echo $form; // vykreslí HTML kód formuláře
```

4.2 Ukázka části jednoduchého formuláře v Nette

Nette většinu těchto věcí umí řešit automaticky. Obsahuje několik předdefinovaných formulářových prvků a validačních pravidel (číslo, email, URL,…) a ochranu proti CSRF[20]. Také umí odebrat přebytečné mezery z jednořádkových textových políček, u selectboxů a jiných prvků s přednastavenými možnostmi výběru zaručuje, že bude zvolena vždy jen některá z povolených možností, a zajistí, že všechny textové vstupy budou validní UTF-8 řetězce.

[20] Cross-Site Request Forgery – [7]

Vykreslení formuláře je jednoduché – k dispozici je automatické vykreslení celého formuláře (či jeho částí) do HTML[21] a v případě složitého členění je možné automatiku nepoužít a o vykreslení jednotlivých části formuláře se postarat „ručně“. Samozřejmostí je již zmíněná validace na straně klienta pomocí JavaScriptu.

Zpracování dat se provádí „navěšením“ obslužných událostí na odesílací tlačítka. Nette tyto události volá dle toho, zda byla odeslaná data validní, není tedy třeba se starat o test na validitu či odeslání. Zde je velmi výhodné formuláře spojit s MVP aplikací, která jejich zpracování začlení do svého životního cyklu.

### 4.3.3 Šablony

Ať už píšeme jednoduchou osobní stránku či složitou webovou aplikaci, nejčastějším výstupem skriptů bude (X)HTML kód, určený pro webový prohlížeč. Vysvětlení tohoto kódu je nad rámec této práce, pro další potřeby postačí vědomost, že (X)HTML kód je textový soubor, obsahující kromě samotného textu také speciální značky, které prohlížeči říkají, jak stránku vykreslit (resp. význam). Protože tento kód chceme vytvořit dynamicky pomocí skriptu, bude třeba jej nějak propojit.

Představme si situaci, že chceme vykreslit seznam zboží v internetovém obchodě. Data získáme skriptem z databáze, následně je v cyklu projdeme a každý záznam vypíšeme jako řádek tabulky. Také chceme odlišit sudé a liché řádky. Bez šablon by takový postup mohl vypadat zhruba následovně:

```
<table border="3">
<thead>
    <tr>
        <th>Produkt</th>
        <th>Cena</th>
    </tr>
</thead>
<tbody>
<?php
$i = 1;
foreach($products as $product) { ?>
    <tr<?php if($i % 2 === 0) { ?> class="sudy"<?php } $i++; ?>>
        <td><?= htmlSpecialChars($product->name, ENT_QUOTES); ?></td>
        <td><?= htmlSpecialChars($product->price, ENT_QUOTES); ?> Kč</td>
    </tr>
<?php
```

[21] Ve starších verzích framework podporoval HTML i XHTML, kvůli omezenému vývoji XHTML se v novější verzi používá HTML 5.

```
} ?>
</tbody>
</table>
```

4.3 HTML kód bez použití šablony

Vidíme, že smíchání kódu stránky s kódem skriptu dělá oba kódy hůře čitelné a zápis je mnohem komplikovanější. Nette umožňuje stejný kus kódu zapsat mnohem jednodušeji:

```
<table border="3">
<thead>
    <tr>
        <th>Produkt</th>
        <th>Cena</th>
    </tr>
</thead>
<tbody n:inner-foreach="$products as $product">
    <tr n:class="$iterator->isEven() ? sudy">
        <td>{$product->name}</td>
        <td>{$product->price} Kč</td>
    </tr>
</tbody>
</table>
```

4.4 HTML kód s použitím šablonovacího jazyka

S použitím šablon se nemusíme starat o povinné escapování znaků, abychom se vyhnuli útokům XSS[22] – Nette proměnné escapuje samo a navíc kontextuálně. To znamená, že pozná, v jakém prostředí se proměnná vypisuje, a podle toho zvolí vhodnou formu escapování (JavaScript, (X)HTML,…).

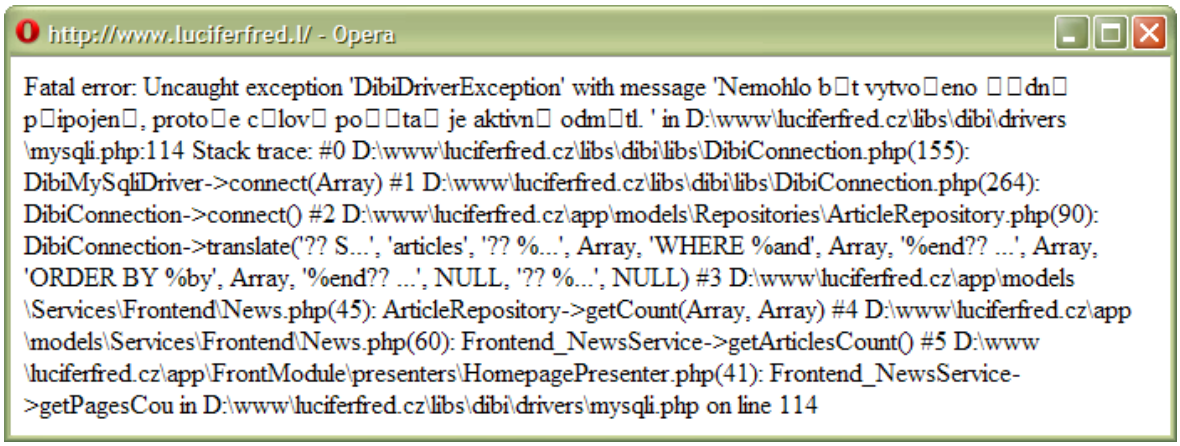
Kromě maker a funkcí uvedených v příkladu existuje celá řada dalších: jednoduché konstrukce jako jsou podmínky a cykly, systém helperů[23], bloky[24], kešování, vytváření odkazů (při využití MVP aplikace) a další.

### 4.3.4 Laděnka

I sebelepší programátor se někdy setká s chybou – od méně podstatného upozornění až po fatální chybu, která ukončí činnost aplikace. Jazyk PHP má sám o sobě pěkně vyvedená chybová hlášení, ale přeci jen pro účely ladění nejsou plně dostačující. Nejlépe je to poznat

[22] Cross-site scripting – [8]
[23] Libovolná formátovací funkce vybraného úseku kódu nebo proměnné – např. vykreslení data ve zvoleném formátu
[24] Umožňují určité části šablon vkládat na místa v jiných šablonách nezávisle na jejich zanoření.

při zobrazení výjimky, kdy se kamkoliv do výstupu vypíše nepřehledný výpis zásobníku, jak je patrné z následujícího obrázku.




**4.5 Standardní výpis výjimky v PHP**

S použitím Laděnky získáme mnohem přehlednější a užitečnější výpis informací.

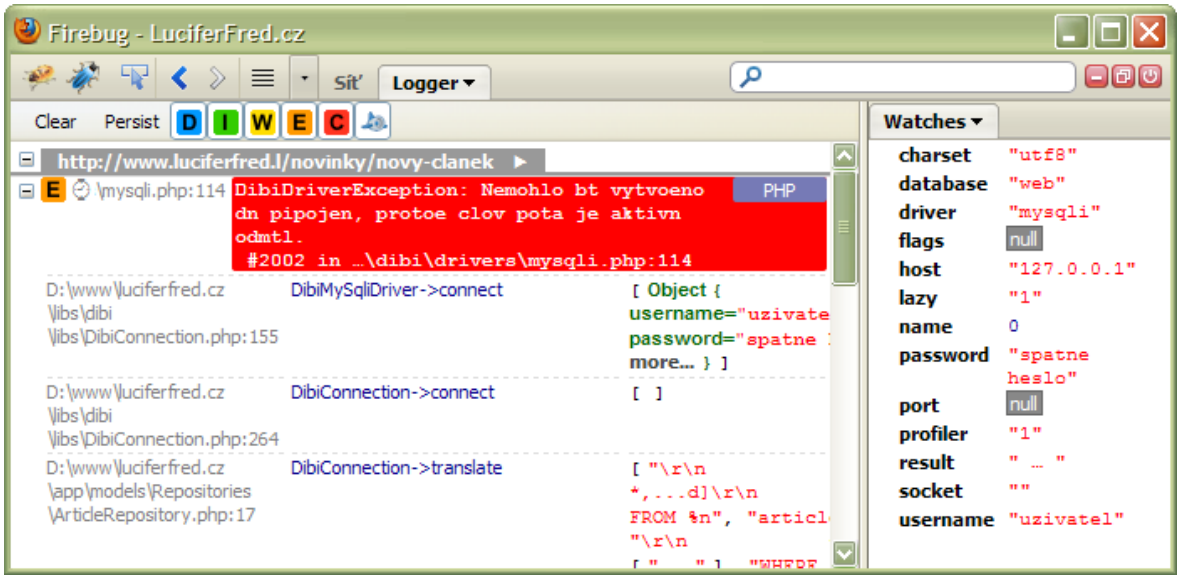

4.6 Výpis výjimky Laděnkou

Na první pohled vidíme jméno a kód výjimky, řádek, na kterém k chybě došlo včetně jeho okolí. Kliknutí na název souboru jej otevře v nastaveném editoru včetně skoku na příslušný řádek. Nabízí se možnost prohlédnout si celý zásobník včetně argumentů, které byly předávány (v našem případě si tak lze ověřit, zda se předaly správné údaje pro

přihlášení k DB). Níže je také uveden výpis informací o prostředí, o HTTP[25] požadavku i odpovědi a o konfiguraci serveru.

Tento výpis je užitečný nejen při vývoji aplikace, ale i při jejím samotném provozu na produkčním serveru. Tam se tato obrazovka neukáže běžnému návštěvníkovi, ale uloží se do adresáře s logy v podobě HTML souboru. Vývojáři pak stačí jedno kliknutí a má všechny podrobné informace o tom, jakým způsobem k chybě došlo a jak ji reprodukovat.

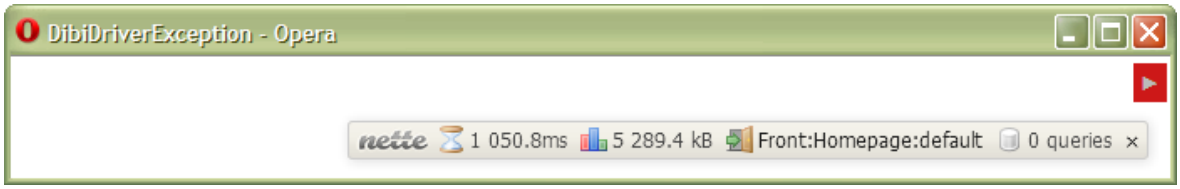
Laděnka také usnadňuje práci při vývoji AJAXových aplikací[26]. V kombinaci s vhodným prohlížečem umožňuje komunikovat s FireLoggerem[27] a zobrazit nám tak případnou chybu či obsah proměnných.




4.7 Výjimka zobrazená při AJAXovém požadavku.

Posledním pomocníkem, který nám Laděnka může nabídnout, je tzv. Debug Bar. Jde o jednoduchou sadu panelů zobrazenou na stránce, která obsahuje panely s různými informacemi: upozornění na méně závažné chyby, routování, dotazy do databáze a spousty dalších dle potřeby, neboť lze vytvářet i panely vlastní. Obsah jednotlivých panelů se zobrazuje najetím kurzoru myši.

[25] Hypertext Transfer Protocol
[26] Asynchronous JavaScript and XML
[27] FireLogger – http://firelogger.binaryage.com




4.8 DebugBar

Na obrázku je také patrná „schovaná“ Laděnka – šipečka v pravém horním rohu.

### 4.3.5 Ostatní

Kromě výše zmíněných součástí nabízí Framework celou řadu dalších nástrojů:

- NEON[28] – jazyk určený pro zápis konfigurace. Je podobný jazyku YAML[29], oproti němu je však lehce zjednodušen a optimalizován pro použití v Nette Frameworku.
- SafeStream – nástroj rozšiřující standardní funkce pro práci se soubory. Zajišťuje, že každá operace proběhne atomicky a izolovaně. [9]
- Image – knihovna pro práci s obrázky.
- Komponentový model
- Autentizaci[30] a autorizaci[31] uživatelů
- String – knihovna pro práci s řetězci. Kromě jiného obsahuje také upravené metody pro práci s regulérními výrazy, které správně ošetřují všechny možné chybové situace.

Kompletní popis všech částí frameworku by byl nad rámec této práce, pro zájemce mohu doporučit online dokumentaci [9].

## 4.4 Komunita a vývoj

S vývojem Nette pomáhá početná komunita vývojářů. Dobrá rozšiřitelnosti frameworku umožnila vzniknout mnoha doplňkům. Na webu[32] najdeme nové formulářové prvky, podporu online plateb, stahování souborů, překladače pro vícejazyčné weby, nástroj pro tvorbu PDF či podporu logování provozu aplikace.

[28] http://ne-on.org/
[29] http://www.yaml.org/
[30] Určení identity uživatele, který vstupuje do systému. [10]
[31] Ověření práv uživatele. [10]
[32] http://addons.nette.org/

Dobrým zvykem je také pravidelné neformální měsíční setkání členů komunity na akci s názvem Poslední sobota[33], kde probíhají krátké přednášky na vybraná témata a následná diskuze o dalším vývoji frameworku.

[33] http://nette.org/cs/komunita

# 5 Analýza a implementace ukázkové aplikace

Součástí distribuce Nette Frameworku je také základní aplikace nazývaná „sandbox“. Jde o předpřipravenou kostru aplikace, z které lze vycházet při tvorbě aplikace vlastní.

Obsahuje:

- ukázkový model pro přihlašování uživatelů,
- konfiguraci,
- základní adresářovou strukturu,
- úvodní presenter,
- přihlašovací presenter,
- šablony presenterů,
- a bootstrap soubor.

Nevýhodou této aplikace je, že nelze předem navolit její podobu. Bylo by tedy pěkné mít možnost nějak jednoduše podobu tohoto sandboxu určit předem a stáhnout si tak aplikaci upravenou sobě na míru.

## 5.1 Zadání

Vytvořit aplikaci, která umožní pomocí přehledného grafického rozhraní nastavit parametry výsledného sandboxu. Takto přizpůsobený sandbox lze následně stáhnout ve formátu ZIP archivu. Aplikace by mohla sloužit jako webová služba pro vývojáře a v budoucnu se stát součástí oficiálního webu Nette Frameworku.

### 5.1.1 Technické detaily

Aplikace bude splňovat následující požadavky:

- její funkčnost není závislá na zapnutém či vypnutém JS,
- v případě zapnutého JS ulehčuje práci využitím AJAXu,
- nastavení se uchovává pro každého uživatele zvlášť, nejlépe do SESSION,
- není vyžadována žádná zvláštní identifikace uživatele (např. registrace),
- uživatel má možnost definovat presentery a jejich rodiče,

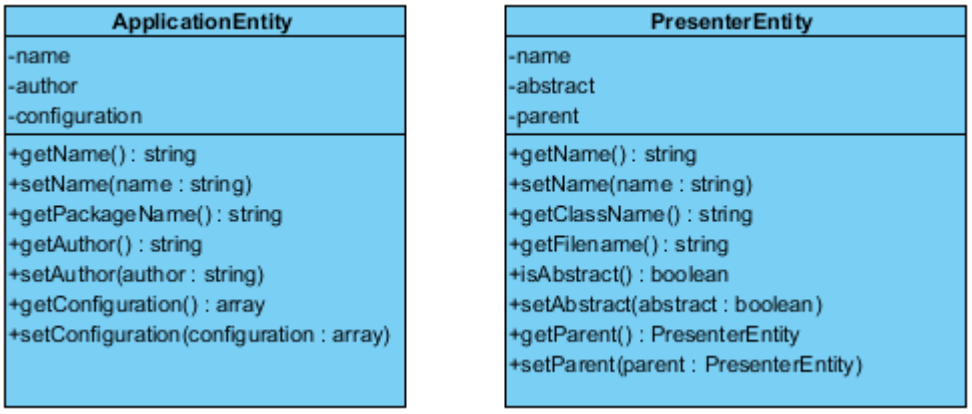
- lze také nastavit jméno aplikace a jejího autora; tyto údaje se použijí v komentářích,
- vygenerovaný sandbox bude obsahovat nabídku, která umožní procházet mezi jednotlivými presentery.

## 5.2 Analýza domény a tvorba modelu

V této aplikaci je doménový model velmi jednoduchý. Máme dvě doménové třídy:

- aplikaci,
- presenter.

Aplikace je vždy jedna, obsahuje nastavení a presentery. Presenterů může být několik, mohou od sebe dědit a mohou být označeny za abstraktní třídu.




5.1 Diagram entit modelové vrstvy

### 5.2.1 Modelová vrstva

Při přemýšlení o tom, jaký nástroj bude nejvhodnější pro tvorbu modelové vrstvy, jsem bral v první řadě v potaz druh úložiště. Od samého začátku jsem se klonil k SESSION úložišti[34], neboť mělo pro daný účel samé výhody:

- standardní součástí jazyka, netřeba žádný další systém (např. databáze),
- Nette poskytuje objektově orientované nástroje pro práci s ním,
- je vázán ke konkrétnímu uživateli (resp. jeho prohlížeči) – netřeba žádné další uživatelovi identifikace.

[34] Úložiště na straně serveru, identifikace probíhá na základě cookie zaslané klientem – [11]

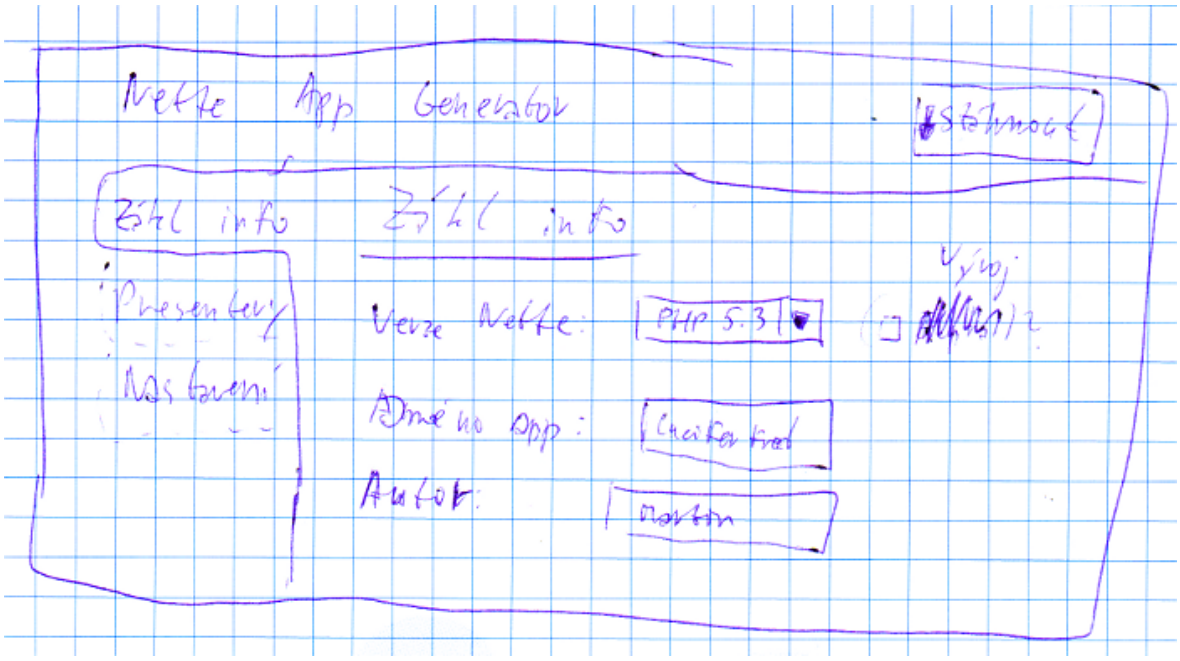
Protože jde o velmi specifické použití (obvykle se jako úložiště používají relační databáze postavené na jazyku SQL, ke kterým existují nástroje pro práci s daty – různé ORM systémy či ActiveRecordy), rozhodl jsem se napsat vlastní vrstvu pro práci s entitami. Ta vychází z pěti vrstev modelu [3], některé jeho součásti ovšem slučuje dohromady.

## 5.3 Tvorba uživatelského rozhraní

Uživatelské rozhraní se skládá z několika samostatných částí a liší se dle podpory JavaScriptu v prohlížeči. Kód je psán v HTML 5, vzhled je definován pomocí CSS.

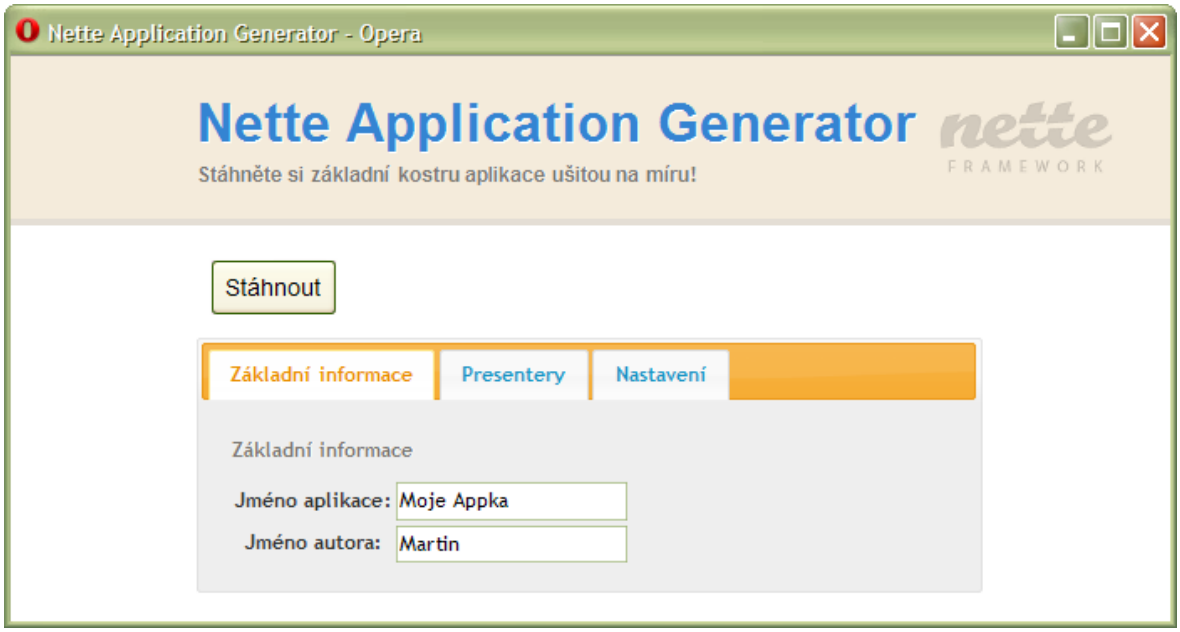
### 5.3.1 Wireframy

Před samotným psaním kódu uživatelského rozhraní je vhodné si nakreslit, jak by takové rozhraní mělo vypadat. To nám pomůže udělat si jasnou představu o tom, jak by aplikace měla vypadat a jak by se mohla používat. Také nám to může pomoci v ujasnění si samotné funkcionality, kvůli které aplikaci píšeme.




5.2 Wireframe úvodní stránky aplikace s vybranou první záložkou

Na obrázku 5.2 můžeme vidět první návrh aplikace. Je patrné, že do výsledné aplikace došlo ještě k několika změnám – byly přesunuty záložky ze svislé polohy do vodorovného řádku, výběr verze Nette se nakonec nerealizoval kvůli technické náročnosti.




5.3 První nakódovaný design aplikace

### 5.3.2 Bez JavaScriptu

Úvodní stránka umožňuje zadat informace o aplikaci a změnit její nastavení. Uživatel také vidí výpis presenterů, s kterým může dále pracovat. Důležitou součástí je tlačítko pro Stažení hotového sandboxu.

Uživatel může přidat nový presenter kliknutím na tlačítko „Přidat“, upravit stávající presenter kliknutím na jeho jméno či presenter smazat. Ve všech případech dojde k přesměrování na konkrétní stránku určenou dané akci. Po provedené této akce bude vrácen na úvodní stránku.

### 5.3.3 S JavaScriptem

S podporou JavaScriptu se jednotlivé části úvodní stránky přepnou pro větší přehlednost do záložek (jQuery UI – Tabs[35]). Zajímavější je ovšem AJAXové chování aplikace. To se týká všech operací okolo presenterů.

Při kliknutí na některou z těchto akcí (přidání, editace, smazání) se otevře nad aplikací okno (využívá se knihovny FancyBox[36]), do kterého se načte část stránky AJAXem. Z té samé stránky, jaká se uživateli zobrazí při vypnutém JavaScriptu. „Storno“ tlačítka toto okno zavřou, potvrzovací tlačítka odešlou pomocí AJAXu formulář (doplněk

[35] http://jqueryui.com/demos/tabs/
[36] http://fancybox.net/

Nette Frameworku pro AJAXové odesílání formulářů[37]) a zpracují výsledek – v případě chyby nahradí obsah okýnka novou stránkou, kde je chyba vypsána; v případě úspěchu se pomocí AJAXu načte tabulka presenterů (standardní součást Nette Frameworku pro práci s AJAXem[38]).

Všechen kód využívá frameworku jQuery[39].

## 5.4 Tvorba webové aplikace

### 5.4.1 Struktura aplikace

Zjednodušený popis struktury aplikace, který záměrně vynechává některé méně podstatné soubory:

```
/
|--app          # samotná aplikace
|   |--components    # komponenty
|   |   |--PresentersManagementControl.php     # vykresluje
|   |   |                                      # presentery a
|   |   |                                      # ovládací prvky
|   |   |                                      # k nim
|   |   |--PresentersManagementControl.latte   # šablona
|   |                                          # komponenty
|   |
|   |--models         # modelová vrstva
|   |   |--services         # „služby" (třídy plní úlohu více
|   |   |   |               # vrstev modelu)
|   |   |   |--ApplicationService.php      # stará se o entitu
|   |   |   |                              # aplikace
|   |   |   |--BaseService.php             # předek všech služeb
|   |   |   |--PresenterService.php        # stará se o entity
|   |   |   |                              # presenterů
|   |   |--ApplicationEntity.php     # entita aplikace
|   |   |--Entity.php                # předek všech entit
|   |   |--exceptions.php            # výjimky modelové vrstvy
|   |   |--PresenterEntity.php       # entita presenteru
|   |
|   |--presenters    # presentery
|   |   |--BasePresenter.php         # předek všech presenterů
|   |   |--ErrorPresenter.php        # chybový presenter se
|   |   |                            # stará o uživatelsky
|   |   |                            # přívětivé informování o
```

[37] http://addons.nette.org/cs/ajax-form-s-eventy
[38] Knihovnu bylo třeba lehce upravit kvůli požadavkům ostatních pluginů
[39] http://jquery.com/

```
|   |   |                              # chybě aplikace
|   |   |                              # (neočekávaná chyba,
|   |   |                              # nenalezená stránka,…)
|   |   |--HomepagePresenter.php       # úvodní presenter
|   |   |--PresentersPresenter.php     # presenter pro práci
|   |                                  # s presentery
|   |
|   |--templates     # šablony presenterů
|   |   |--Error
|   |   |--Homepage
|   |   |--Presenters
|   |   |   |--@notFound.latte   # šablona pro AJAXové chybové
|   |   |   |                    # hlášky
|   |   |   |--delete.latte      # smazání presenteru
|   |   |   |--edit.latte        # vytvoření nového či editace
|   |   |   |                    # stávajícího presenteru
|   |   |   |
|   |   |--@layout.latte         # šablona layoutu, používá se
|   |                            # ve všech presenterech
|   |
|   |--bootstrap.php # zaváděcí soubor
|   |--config.neon   # konfigurace aplikace
|
|--data
|   |--templates     # šablony pro generování částí sandboxu
|   |--sandbox.zip   # předpřipravený archiv sandboxu
|
|--libs         # používané knihovny/frameworky
|   |--Custom         # vlastní knihovny
|   |--Nette          # Nette Framework
|   |--NetteAddons    # rozšíření Nette Frameworku
|
|--log          # logy Laděnky
|
|--temp         # dočasná aplikační data
|   |--cache          # kompilované šablony,…
|   |--session        # soubory uživatelských session
|   |--zips           # generované sandboxy
|
|--www          # root webu (index.php,…)
|   |--css            # styly
|   |--images         # grafika
|   |--js             # JavaScripty
```

### 5.4.2 Presentery

Aplikace se skládá ze dvou základních presenterů:

- *HomepagePresenter* – pro práci s aplikací a generováním výsledného sandboxu,
- *PresentersPresenter* – pro práci s presentery.

Dále se používá *BasePresenter,* který má na starosti společné úlohy: automaticky promítá změny do úložiště při ukončení, vytváří komponentu pro práci s presentery, poskytuje zkratky pro přístup k službám modelové vrstvy,…

V aplikaci lze také nalézt *ErrorPresenter*. Se samotným účelem aplikace nemá nic společného – jeho smysl je ve zpracování případných neošetřených chyb, či při požadavku na neexistující adresu. Takové situace umí pomocí Laděnky zaznamenat a informovat uživatele přívětivou zprávou.

### 5.4.3 Šablony

V aplikaci existují dva druhy šablon:

1. vlastní vzhled aplikace (tj. šablona layoutu, šablony jednotlivých pohledů a šablony komponent),
2. šablony pro generování sandboxu.

V případě šablon pro vzhled aplikace jde o běžné použití šablonovacího systému Nette Frameworku. Využívají se zde makra[40], dědičnost, snippety[41],… Za zmínku stojí způsob, kterým se v jQuery přistupuje k formulářovým prvkům – v šabloně lze vypsat ID jakéhokoliv prvku formuláře. To nám ušetří práci v případě budoucího předělávání formulářů.

Zajímavější je použití šablon pro generování sandboxu. Máme zde šablonu pro konfigurační soubor, pro soubor presenteru a pro šablonu presenteru (pohled). Není zde výrazný rozdíl od běžného použití šablon, výjimku tvoří šablona pohledu – je třeba ošetřit, která makra se mají zpracovat a která se mají naopak vygenerovat. Přes tuto drobnou komplikaci mi přijde použití šablonovacího systému přehlednější než napřímo vepisované PHP.

[40] Rozšiřují funkcionalitu – podmínky, cykly, tvorba odkazů,… [9]
[41] Zjednodušují práci s AJAxem – umožňují označit část stránky, která se při změně pošle samostatně namísto posílání celé stránky najednou. [9]

### 5.4.4 Model

Modelové vrstvě jsme se již věnovali v samostatné kapitole. Z pohledu aplikace je práce s entitami řízena přes služby, které jsou registrovány v konfiguračním souboru. Tyto služby umožňují získávat entity z úložiště, pracovat s nimi a ukládat je. Poskytují i funkcionalitu nad rámec modelované domény: například dokáží zjistit všechny předky či potomky presenteru.

Entity jsou inspirovány frameworkem Doctrine 2[42] – pro získávání hodnot property se ve službách používá reflexe, která v PHP 5.3 umí číst i soukromé atributy. V entitách i službách se využívá novinka Nette Frameworku a tou je Dependency Injection [12] představovaný třídou NContext. Využívá se toho například ve chvílích, kdy entita presenteru potřebuje vygenerovat jméno souboru – využije služby IPresenterFactory, ktera do ní byla vložena v kontextu a nastavena z venku (v tomto konkrétním případě byla nastavena cesta k aplikační složce v rámci sandboxu).

[42] http://www.doctrine-project.org/projects/orm

# 6 Závěr

Cílem práce bylo shrnutí problematiky v oblasti tvorby webových aplikací. Přiblížili jsme si fungování internetu z pohledu webových technologií (v kapitole 3.1) a seznámili se se specifiky vývoje (v kapitole 3.3). Také jsme si stručně představili některé existující jazyky/platformy a jejich frameworky (v kapitole 3.2).

Následující část byla zaměřena na seznámení se s Nette Frameworkem. V kapitole 4 lze nalézt obecné informace o Nette a teoretický popis stěžejních součástí.

Dalším cílem práce bylo vytvořit aplikaci, která bude zmíněný framework využívat. Analýzu a popis aplikace lze nalézt v kapitole 5. Výslednou aplikaci pak lze nalézt na přiloženém CD.

# 7 Seznam použitých zdrojů

## 7.1 Tištěné zdroje


1. Kosek, J.: PHP - tvorba interaktivních internetových aplikací. Vydání první. Praha: Grada Publishing 1999. ISBN 80-7169-373-1


## 7.2 Internetové zdroje


2. HTTP - Hypertext Transfer Protocol [online]. [cit. 2011-03-28]. <http://www.w3.org/Protocols/>
3. Pět vrstev modelu [online]. [cit. 2011-03-28]. <http://www.phpguru.cz/clanky/pet-vrstev-modelu>
4. About Ruby [online]. [cit. 2011-03-28]. <http://www.ruby-lang.org/en/about/>
5. Nette coming soon! [online]. [cit. 2011-03-28] <http://latrine.dgx.cz/nette-coming-soon>
6. Prezentační vzory z rodiny MVC [online]. [cit. 2011-03-28]. <http://zdrojak.root.cz/clanky/prezentacni-vzory-zrodiny-mvc/>
7. Cross-Site Request Forgery [online]. [cit. 2011-03-28]. <http://php.vrana.cz/cross-site-request-forgery.php>
8. Cross Site Scripting [online]. [cit. 2011-03-28]. <http://php.vrana.cz/cross-site-scripting.php>
9. Dokumentace | Nette Framework [online]. [cit. 2011-03-28]. <http://doc10.nette.org/cs/>
10. Django: Autentizace a autorizace [online]. [cit. 2011-03-28]. <http://zdrojak.root.cz/clanky/django-autentizace-a-autorizace/>
11. PHP: Introduction - Manual [online]. [cit. 2011-03-28]. <http://cz.php.net/manual/en/intro.session.php>
12. Simplifying Dependency Injection [online]. [cit. 2011-03-28]. <http://blog.architexa.com/2010/04/simplifying-dependency-injection/>

# 8 Přílohy

## 8.1 Obsah přiloženého CD s aplikací

Výslednou aplikaci najdete spolu s touto bakalářskou prací umístěnou na přiloženém CD.

```
/
|--sandbox-generator          # aplikace
|--text
|   |--bakalarska-prace.pdf   # bakalářská práce
```