Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Bakalářská práce

Webový portál pro demonstraci skriptů psaných v Pythonu

Plzeň, 2014 Antonín Neumann

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 8. května 2014

Antonín Neumann

Poděkování

Chtěl bych poděkovat vedoucímu své práce panu Ing. Michalovi Camprovi za ochotnou pomoc při řešení potíží. Své přítelkyni za její podporu a za jazykové a stylistické rady. Rovněž bych chtěl poděkovat panu Ing. Vladimíra Lukeše, Ph. D. za jeho rychlou pomoc při řešení problémů. A v neposlední řadě také své mamince za podporu a zázemí.

Abstrakt

Tato práce popisuje tvorbu webového portálu pro demonstraci skriptů psaných v Pythonu. Na začátku práce se věnuji možnosti tvorby webových aplikací, speciálně pak použití webových frameworků.

Pokračuji návrhem databázového modelu a implementací toho návrhu ve zvoleném frameworku. Další částí práce je návrh dokumentu XML definující parametry pro generování vstupního Python skriptu. Realizace je zaměřena na knihovnu SfePy určenou pro řešení parciálních diferenciálních rovnic.

Abstract

The aim of this thesis is to describe the construction of a web portal that demonstrates scripts written in Python. In the first part, creation of web applications especially the use of web frameworks are presented.

The thesis then deals with proposal of database model and implementation of the proposal in the chosen framework. In another section the thesis provides design of XML document that defines the parameters for generating the input of the Python script. The implementation is focused on library SfePy designed to solve partial differential equations.

Obsah

1 Úvod 1
2 Technologie 2
2.1 Základní webové technologie 2
2.2 Skriptovací jazyk 2
2.2.1 Skriptování na straně klienta 3
2.2.2 Skriptování na straně serveru 3
2.3 Formát pro uložení a výměnu strukturovaných dat 4
2.4 Software pro správu verzí 5
2.5 Databáze 5
2.6 Webový server 6
3 Porovnání PHP frameworků vhodných pro tvorbu specializovaných webových aplikací 7
3.1 Výběrová kritéria 7
3.2 Kandidáti 8
3.3 Čisté PHP 8
3.4 WordPress a Drupal 8
3.5 Kohana framework 9
3.5.1 Stažení a instalace 9
3.5.2 Vlastnosti a přednosti 10
3.5.3 Nevýhody 10
3.5.4 Práce s databází 10
3.6 Nette Framework 10
3.6.1 Stažení a instalace 11
3.6.2 Vlastnosti a přednosti 11
3.6.3 Nevýhody 12
3.6.4 Práce s databází 12
3.7 Zend framework 2 12
3.7.1 Stažení a instalace 12
3.7.2 Vlastnosti a přednosti 12
3.7.3 Nevýhody 13
3.7.4 Práce s databází 13
3.8 Symfony 13
3.8.1 Stažení a instalace 13
3.8.2 Vlastnosti a přednosti 14
3.8.3 Nevýhody 14
3.8.4 Práce s databází 14

3.9 Srovnání výkonu frameworků při výběrovém dotazu ... 15
4 Popis vybraného řešení pro tvorbu aplikace ... 16
4.1 Technické a programové vybavení ... 16
4.1.1 Vývojové prostředí ... 16
4.2 Konfigurace ... 16
4.2.1 Nastavení aplikace ... 16
4.2.2 Konfigurace skriptu ... 17
4.3 Moduly použité v Symfony ... 17
4.3.1 Doctrine ... 17
4.3.2 FOSUserBundle ... 17
4.3.3 DoctrineFixturesBundle ... 18
4.3.4 Twig/extensions ... 18
4.3.5 GenemuFormBundle ... 19
5 Návrh databáze a jednotlivých entit ... 20
5.1 Popis entit ... 20
5.2 Entita Script ... 21
5.3 Entita Problem ... 21
5.4 Schéma databázového modelu ... 22
5.5 Ukázka definice entity v Doctrine2 ve formátu YAML ... 23
6 Definice a popis problémů ... 25
6.1 Návrh struktury XML ... 25
6.2 Implementace parseru pro převod XML ... 26
6.3 Generování formuláře ... 27
6.4 Generování Python skriptu a spuštění výpočtu ... 27
7 Závěr ... 29
Seznam literatury ... 30
Seznam obrázků ... 32
Seznam zkratek ... 33
Příloha A – definice entit v YAML ... 34
A.1 Definice entity Problem ... 34
A.2 Definice entity Script ... 35
Příloha B – definice početních problémů v XML ... 36
B.1 Vzor pro definici početních problémů ... 36
B.2 Definice problému Linear Elasticity ... 38
Příloha C – ukázky webového rozhraní pro práci s XML ... 40
C.1 Ukázka vygenerovaného formuláře ... 40
C.2 Ukázka správy XML u entity Problem ... 41
C.3 Ukázka vizualizace výsledku ... 42

1 Úvod

Ke vzniku této práce vedla motivace pana Ing. Vladimíra Lukeše, Ph. D. Ten chtěl studentům bakalářského studia na Katedře mechaniky, Fakulty aplikovaných věd na Západočeské univerzitě, ukázat možnosti knihovny SfePy[1].

SfePy je software vyvinutý pro řešení vázaných parciálních diferenciálních rovnic metodou konečných prvků ve 2D a 3D. [1]

Práce tedy vznikla pro potřeby snadné správy skriptů napsaných v programovacím jazyce Python a vizualizace výsledků vypočtených metodou konečných prvků pomocí knihovny SfePy.

Cílem této práce je poskytnout kompletní aplikaci postavenou na webových technologiích. Celá aplikace je poháněna frameworkem Symfony2, který je napsán v programovacím jazyce PHP a relační databází MySQL.

Aplikace, s pracovním názvem Masscom[2], umožní uživatelům spouštění a volbu některých parametrů výpočtu problémů, jež jsou předem nadefinovány pomocí XML souborů. Uživatelům s administrátorskými právy umožňuje vytvářet nové problémy a upravovat stávající, které je možné prostřednictvím skriptů řešit.

Při tvorbě bakalářské práce jsem použil především internetové zdroje, případně odbornou literaturu, zabývající se tvorbou webových stránek.

1 Simple Finite Elements in Python

2 MAnager of Script for Scientific COMputation, česky správce skriptů pro vědecké výpočty

2 Technologie

2.1 Základní webové technologie

Jak již bylo popsáno, cílem této práce je vytvořit webový portál. Webové stránky se vytvářejí ve značkovacím jazyce HTML[3] a pomocí kaskádových stylů CSS[4].

HTML, sloužící pro popis struktury WWW[5] dokumentu (tzv. hypertextu), je odvozený z univerzálního jazyka SGML[6] [2] stejně jako například jazyk XML[7]. Tim Berners-Lee jazyk vymyslel v roce 1989, když pracoval v CERNu [3], později založil World Wide Web Consortium, zkrácené W3C, které dnes spravuje mnohé standardy používané při tvorbě webových aplikací. HTML je aktuálně dostupné ve verzi 4.1, ale již se pracuje na standardu verze 5, která je z velké části webovými prohlížeči už podporována. Ve své aplikaci používám právě nejnovější verzi HTML5.

CSS se používá pro popis vzhledu webových stránek napsaných v HTML, XHTML nebo XML, může být ale použit na jakýkoliv typ dokumentu založeného na XML [4]. Původní verzi, označovanou jako Cascading HTML Style Sheet, vyvinul Håkon Wium Lie v roce 1994 a v roce 1996 bylo organizací W3C oficiálně představeno CSS1 [5].

2.2 Skriptovací jazyk

Samotné HTML a CSS pro vývoj komplexních webových stránek nestačí. Jelikož jsou HTML a CSS vlastně jen statické dokumenty, je potřeba použít ještě nějaký skriptovací jazyk, který dodá aplikaci složitější programovou logiku a dynamičnost.

Ve světě webových technologií, existují v zásadě dva typy skriptovacích jazyků. Jedny na straně klienta, jsou přímou součástí webového prohlížeče a umožňují skriptování bez potřeby komunikace s web serverem.

---

3 HyperText Markup Language
4 Cascading Style Sheets
5 World Wide Web
6 Standard Generalized Markup Language
7 eXtensible Markup Language

Druhé pak na straně webového serveru, které se vykonají ještě před odesláním HTTP[8] odpovědi serverem do webového prohlížeče. Pro jejich provedení je vždy nutné poslat požadavek na webový server.

2.2.1 Skriptování na straně klienta

Dnes se pro skriptování na straně klienta používá převážně JavaScript. Existují i jiné programovací jazyky, kupříkladu CoffeeScript, ovšem všechny jsou před použitím ve webových aplikací překompilovány do zmíněného JavaScriptu.

JavaScript je objektově orientovaný, interpretovaný jazyk, který vytvořil Brendan Eich v roce 1995. Tehdy pracoval ve společnosti Netscape Communications Corporation, na webovém prohlížeči Netscape Navigator [7].

JavaScript je v dnešní době hojně rozšiřován pomocí různých knihoven. Mezi nejznámější patří jQuery, Prototype nebo Moo Tools [8].

Dnes je nejpoužívanější knihovna jQuery. Dovoluje programátorovi mnohem snazší práci s DOM[9] prvky webové stránky, jednodušší obsluhu událostí nebo volání asynchronních HTTP požadavků pomocí technologie AJAX[10].

2.2.2 Skriptování na straně serveru

Serverových programovacích jazyků je na rozdíl od těch na straně klienta mnohem více. Mezi nejrozšířenější, a zároveň nejznámější jazyky patří PHP[11] a ASP.NET[12].

Ve své aplikaci jsem se rozhodl použít jazyk PHP, jelikož s ním mám již zkušenosti a je pro něj k dispozici více webových frameworků, než pro ostatní serverové skriptovací jazyky.

8 HyperText Transfer Protocol [6]
9 Document Object Model
10 Asynchronous JavaScript and XML
11 PHP: Hypertext Preprocessor, původně Personal Home Page
12 Active Server Pages, součást .NET Framework

Jazyk PHP byl původně vyvinutý v roce 1995 jako Perl/CGI skript Rasmusem Lerdorfem pro počítání návštěvnosti jeho životopisu. Později byl rozšířen o nové možnosti, přepsán do jazyku C a publikován jako PHP 2.0, též Personal Home Page/Form Interpret [9]. V současnosti je k dispozici již verze PHP 5.5. Verze PHP 5.3 přinesla velkou změnu v oblasti objektově orientovaného programování. Především jde o podporu tzv. jmenných prostorů, anonymních funkcí a mnoha dalších novinek [10].

ASP.NET je součástí .NET Frameworku vyvíjeného společností Microsoft. Tento framework je možné využít kromě tvorby webových stránek i při programování aplikací na platformě Windows. V ASP.NET lze vytvářet webové stránky pomocí tří frameworků, ASP.NET Web Forms, ASP.NET MVC nebo ASP.NET Web Pages. V každém z těchto tří frameworků lze vytvořit komplexní webovou aplikaci [11].

Dalšími programovacími jazyky pro tvorbu dynamických webových stránek jsou například RoR[13] založený na jazyce Ruby, JSP[14] založený na jazyce Java, Python nebo Node.js umožňující spouštění JavaScriptu na straně serveru.

2.3 Formát pro uložení a výměnu strukturovaných dat

Pro potřebu ukládání strukturovaných dat se ve webových aplikacích nejčastěji využívají formáty XML (případně z něj odvozené formáty), JSON[15] nebo YAML[16]. Existuje samozřejmě mnohem více formátů, například CSV, pro strukturované ukládání dat, ale tyto patří mezi nejpoužívanější při tvorbě webových stránek. Dokonce je možné vytvořit si velmi snadno vlastní textově orientovaný formát pro ukládání strukturovaných dat. K tomu je nutné vytvořit si vlastní „parser“, abychom zapsaná data mohli převádět mezi vytvořeným formátem a nějakým skriptovacím jazykem.

Všechny výše zmíněné formáty dokáží strukturovaná data ukládat rovnocenným způsobem, rovněž ke všem z nich existují „parsery“ pro různé programovací jazyky. Záleží tedy spíše na osobních preferencích, který z nich zvolíme.

13 Ruby on Rails
14 JavaServer Pages
15 JavaScript Object Notation
16 YAML Ain’t Markup Language

Drobné rozdíly mezi nimi přesto existují. Například formát JSON nemá implementovanou podporu pro komentáře, YAML dovoluje zase jen komentáře řádkové. Formát XML je naproti tomu složitější na strojové zpracování a generování.

Čitelnost a srozumitelnost těchto formátů pro lidi je velmi subjektivní a každému člověku vyhovuje nějaký styl více než jiný, ale po určité krátké době se dokáže zorientovat a používat kterýkoli z nich.

2.4 Software pro správu verzí

Hlavním cílem všech programů pro správu verzí je usnadnit spolupráci více lidí na jednom projektu. Přesto se však hodí i pro projekty, na kterých pracuje jen jeden programátor. Ten sice nevyužije všech možností, které tyto programy nabízejí, avšak minimálně návrat ke starší verzi zdrojového kódu nebo možnost synchronizace při práci na více počítačích považuji za velký přínos.

Nejběžnějšími programy pro správu verzí jsou SVN[17], Git a Mercurial. Zatímco SVN byl vyvinut jako nástupce systému CVS[18]. Git a Mercurial započali svůj vývoj jako náhrada BitKeeperu. Ten byl používám pro správu verzí linuxového jádra.

Vývojové prostředí NetBeans, které používám pro tvorbu mé aplikace, podporuje všechny tři výše zmíněné programy pro správu verzí. Důvod pro výběr systému Git, který pro verzování zdrojových kódů používám, byl hlavně služba GitHub.com. Tato služba umožňuje snadné zřízení bezplatného veřejně dostupného repositáře pro jakékoliv projekty s otevřeným zdrojovým kódem, tzv. open source.

2.5 Databáze

Pro ukládání dat se při vývoji webových stránek používají relační databáze, které pro ukládání dat a vztahů mezi nimi používají tabulky.

Mezi nejznámější relační databáze využívané pro tvorbu webových aplikací patří MySQL a PostgreSQL. Další možností je použití MSSQL vyvíjené společností Microsoft, SQLite nebo MariaBD, která je odvozená z MySQL.

17 Apache Subversion

18 Concurrent Versions System – software určený pro správu verzi, vyvinutý Dickem Grunem v roce 1986 [12]

Aplikace si všechny potřebné údaje udržuje v databázi a umožní administrátorovi jejich snadnou úpravu. Je tedy zapotřebí nějaká databázová vrstva, aby nemusel s databází komunikovat jen pomocí základních funkcí, které nabízí PHP. Databázové vrstvy nabízejí programátorovi pohodlnější práci s databází, ošetřují nežádoucí průniky do databáze jako, SQL injection[19], a často nabízejí podporu pro ORM[20], které převádí data z databáze do objektů v programovacím jazyce, tzv. entit. Tento přístup k databázovým datům jako k objektům není nutný, ale urychluje vývoj a zjednodušuje práci s daty.

2.6 Webový server

Webový server je počítač s programem, který se stará o vyřizování HTTP (případně HTTPS aj.) požadavků. Klient, většinou webový prohlížeč, si prostřednictvím HTTP požadavku vyžádá nějaký dokument (nejčastěji HTML nebo PHP soubor). [13]

Server požadavek vyhodnotí, ověří jeho dostupnost, oprávněnost a vrátí jako svoji odpověď požadovaný dokument nebo chybový kód. [14]

Mezi nejpoužívanější servery patří Apache HTTP Server, Internet Information Services (zkráceně IIS) a Nginx. Programy Apache a Nginx jsou dostupné pro většinu známých operační systémů jako GNU/Linux, BSD, Solaris, Mac OS a Windows. ISS je vyvíjeno společností Microsoft a je dostupný pouze pro systém Windows.

Podle posledního měření společnosti Netcraft z dubna 2014 je provoz všech internetových stránek rozdělen podle následující tabulky.

Tabulka 1: Zastoupení webových serverů na celkovém internetovém provozu

Webový server	**Procentuální zastoupení [%]**
Apache HTTP Server	37,74
Internet Information Services	33,04
Nginx	15,25

Zdroj: Netcraft [15]

19 SQL injection je technika napadení databáze pomocí podsunutí vlastního škodlivého SQL příkazu přes neošetřený vstup.

20 Object-relational mapping, česky objektově relační zobrazení

3 Porovnání PHP frameworků vhodných pro tvorbu specializovaných webových aplikací

Na začátku je nutné si ujasnit k čemu budu framework ve své práci potřebovat. Výsledkem mé práce bude webová aplikace používaná pro správu a prezentaci skriptů napsaných v jazyce Python, primárně pro knihovnu SfePy sloužící pro řešení parciálních diferenciálních rovnic.

Webové rozhraní umožňuje administrátorovi definovat různé problémy pomocí XML konfiguračních souborů. Na základě těchto souborů by měl systém vygenerovat formulář, do kterého uživatel bude zadávat hodnoty konkrétního výpočtu. Následně musí systém vložené hodnoty validovat, vygenerovat vstupní soubor, spustit s ním výpočet skriptu a nakonec vizualizovat výsledky.

3.1 Výběrová kritéria

Před samotným porovnáváním různých frameworků, je potřeba stanovit kritéria, která má použitý framework splňovat.

- Podpora architektury MVC[21],
- podpora databázové abstrakce,
- snadná instalace,
- jednoduché programování dodatečných funkcí,
- kvalitní dokumentace.

21 Model-view-controler je softwarová architektura, která rozděluje datový model, uživatelské rozhraní a řídící logiku do tří samostatných celků. Zjednodušeně řečeno model se stará o data, view zobrazuje data například na monitoru a controller řídí tok příkazů a propojuje model s view. [16]

3.2 Kandidáti

Na základě porady s několika lidmi znalými v programování webových stránek, a také po prostudování několika diskuzních fór, jsem dospěl k těmto 7 možným kandidátům:

- Symfony2,
- Zend framework 2,
- Nette Framework,
- Kohana framework,
- CMS[22] WordPress,
- CMS Drupal,
- a čisté PHP.

3.3 Čisté PHP

Při použití čistého PHP není dostupná žádná MVC architektura. Vytvoření vlastní architektury by zabralo dlouhou dobu a výsledek by pravděpodobně stejně nedosahoval stejných kvalit jakou nabízejí několik let vyvíjené frameworky.

PHP samo neřeší ochranu proti různým bezpečnostním hrozbám a neošetřeným vstupům a výstupům, kterou většina frameworků poskytuje.

Výhodou oproti ostatním řešením by byla rychlost celé aplikace, protože bychom zahrnuli jen ty funkce, které jsou v aplikaci skutečně potřeba.

3.4 WordPress a Drupal

Systém WordPress, stejně jako Drupal, nejsou plnohodnotným PHP frameworkem. Jedná se o kompletní systém pro správu obsahu webových stránek.

Tvorba stránek v těchto systémech je velice jednoduchá. Stačí celý systém stáhnou, nakopírovat zdrojové kódy do webové složky a nakonfigurovat celý systém. Nastavení lze provést ve velice jednoduchém webovém instalátoru, který je samo popisný.

22 Content management system (česky systém pro správu obsahu) je souhrnné označení pro software, který umožňuje uživateli vytvářet převážně webový obsah bez znalosti programování v HTML, CSS nebo PHP. Ke správě obsahu stačí základní znalosti ovládání počítače, práce na internetu a práce s textovým editorem.

Oba systémy poskytují velké množství funkcí přes vytváření webový stránek až po různé komplexní statistiky a souborové manažery. Rovněž se tyto systémy drží modelu MVC.

Přidání funkcionality, kterou tyto systémy dosud nenabízejí, se řeší pomocí tzv. pluginů. Plugin je zásuvný modul, tedy samostatný balíček, který potřebnou funkcionalitu do systému dodává. Tento plugin se stará o obsluhu databáze, zobrazování dat a o jejich správu přes administrační rozhraní.

Vytvoření zásuvného modulu vyžaduje prostudování tvorby pluginu pro daný systém. Rovněž je třeba využívat například implementovaný šablonový systém, databázovou abstrakci a respektovat konvence určené autory systému.

3.5 Kohana framework

Kohana je vyvíjena od roku 2007 a její současná verze je 3.3.1. Celý framework je licencován pod BSD license a využívá architekturu HMVC[23]. Kohana je původně forkem, odvozeninou, projektu CodeIgniter. Celý projekt je v angličtině.

Dokumentace [17] k projektu je rozdělená podle jednotlivých vývojových verzí. Je dobře strukturovaná, přehledná a nabízí množství postupů včetně ukázek zdrojového kódu.

3.5.1 Stažení a instalace

Požadavky pro správnou funkčnost frameworku jsou minimálně PHP 5.3.3 s rozšířeními „iconv“ a „ctype“. Obě tyto rozšíření podporuje i většina freehostingů.

Stažení Kohana frameworku je možné provést přímo ze stránek projektu nebo také z veřejného repositáře na stránkách GitHubu.

Po stažení a přesunutí zdrojových kódu do webové složky, je nutné celý framework nastavit. Nastavení je velice snadné a podrobně s ním seznamuje dokumentace. Jde zejména o správné nastavení časové zóny a nastavení podadresáře, ve kterém je framework umístěn. Dále pak o změnu práv pro složky cache a logs, případně nastavení řetězce pro cookie.

23 Hierarchical MVC je softwarová architektura, velmi podobná PAC (Presentation-abstraction-control) architektuře . Jedná se o variantu klasického MVC modelu, kdy jeden kontrolér má pod sebou další kontroléry a tvoří takto stromovou strukturu. Vrchní kontrolér deleguje požadavky na odpovídající nižší kontroléry.

Poté si zobrazíme kořenovou webovou stránku, kde bychom měli vidět report o správné funkčnosti. Pokud na této stránce není žádná chyba, podařilo se nám úspěšně zprovoznit framework Kohanu. Nakonec ještě smažeme soubor install.php. Kde již uvidíme výstup implicitního kontroleru.

3.5.2 Vlastnosti a přednosti

Zajímavou vlastností toho frameworku je podpora hiearchického MVC, které může někomu vyhovovat více, než běžněji dostupná obyčejná MVC architektura.

3.5.3 Nevýhody

Za velkou nevýhodu lze považovat nekompatibilitu verzí 2.x a 3.x. Přestože oficiální podpora pro verzi 2.x byla ukončena v roce 2013, jde stále o velmi rozšířenou verzi. Je tedy nutné při hledání návodů dávat dobrý pozor, pro kterou vývojovou větev frameworku jsou určeny.

3.5.4 Práce s databází

Kohanu má pro práci s databází specíální modul, který podporuje ovladače MySQL i PDO[24]. Framework nabízí možnost nastavení vlastního modulu s databázovou abstrakcí, například Doctrine2 [18] nebo Propel [19].

3.6 Nette Framework

Nette Framework je vyvíjen přibližně od roku 2008 a v současné době se nachází poměrně čerstvě ve verzi 2.0.12. Tento framework je licencován pod New BSD license a GNU GPL version 2 i GNU GPL version 3. Lze si tedy vybrat licenci, která lépe vyhovuje projektu. Celý projekt je v češtině s aktivní českou komunitou.

Dokumentace k projektu je přehledná a kvalitní. Je k dispozici mnoho ukázkových postupů včetně zdrojových kódů a rovněž API reference.

24 PHP Data Objects je rozšíření standardního PHP o abstrakční vrstvu, která dovoluje přistupovat k různým databázím pomocí stejných funkcí.

3.6.1 Stažení a instalace

Aktuální stabilní verzi frameworku můžete stáhnout přímo na stránkách projektu nebo pomocí nástroje zvaného Composer [20]. Vývojovou verzi je možné stáhnout z veřejného Git repositáře na stránkách GitHubu.

Nette lze provozovat na PHP verze 5.2 i 5.3, přestože vyšší verze PHP je doporučována, jsou si obě funkčně rovnocenné. Pomocí aplikace nazvané Requirements Checker si můžeme ověřit splnění minimálních požadavků pro správné fungování frameworku.

Nette framework obsahuje velmi propracovanou adresářovou strukturu, která poskytuje různé více či méně užitečné nástroje. Rovněž obsahuje několik ukázkových aplikací, které si před samotnou tvorbou naší aplikace můžeme projít.

3.6.2 Vlastnosti a přednosti

Hlavní předností frameworku je bezesporu obsáhlá a aktivní česká komunita.

Knihovna Nette\Diagnostics\Debugger známá rovněž pod názvem „Laděnka“ je další velkou předností. Tento ladící nástroj si programátoři na Nette cení, velmi pomáhá při testování aplikace a poskytuje programátorovi nadstandardní ladící informace. Nette je velmi otevřený framework a je koncipován tak, že jednotlivé části je možné využít i mimo samotný framework – tedy i „Laděnku“.

Nette poskytuje velmi dobrou ochranu před různými typy útoků na webovou stránku, například CSRF[25] nebo XSS[26], což se děje prostřednictvím vlastního šablonového systému nazývaného Latte.

Také se řídí zásadami „méně je více“ (KISS – Keep It Sort and Simple) a „neopakuj sám sebe“ (DRY – Don't repeate yourself). [23]

Pro všechny konfigurační soubory se nyní používá systém NEON [24], který je velmi podobný zápisu ve formátu YAML. Jedná se o datovou strukturu nezávislou na konkrétní platformě, která slouží pro přenos dat.

25 Cross-Site Request Forgery. Podstata útoku spočívá v tom, že uživatele přimějeme navštívit stránku napadané aplikace, která provádí nějakou akci, aniž by o tom uživatel věděl. [22]

26 Cross Site Scripting „je způsob narušení webových stránek, kdy je útočník modifikuje tak, že se v jejich kontextu provede podstrčený JavaScriptový kód.“ [21]

3.6.3 Nevýhody

Výše popsané výhody, například šablonový systém Latte a konfigurační systém NEON jsou zároveň i nevýhodou Nette, protože nejsou standardizovány a může dojít k jejich nečekané změně nebo zrušení. Rovněž lze za určitou nevýhodu považovat fakt, že framework Nette je vyvíjen pouze jedním člověkem, přestože je celý systém otestován komunitou.

3.6.4 Práce s databází

Pro práci s databází nabízí Nette vlastní knihovnu Nette\Database\Connection. Jedná se o obálku nad standardní PHP knihovnou PDO. Pro Nette ovšem existuje množství doplňků, které umožňují používat například Doctrine 2 nebo Dibi.

3.7 Zend framework 2

Zend framework je vyvíjen od roku 2005 a jeho současná verze 2 byla vydána v roce 2010. Framework je licencován pod New BSD license.

Zend je plně objektově orientovaný framework, který vyžaduje pro svůj běh nejnovější verzi PHP 5.3.

3.7.1 Stažení a instalace

Zend framework 2 je možné stáhnout mnoha způsoby, například přímo ze stránek projektu, pomocí Composeru nebo také ze stránek projektu na Githubu.

Při stažení frameworku se stáhne i tzv. application skeleton, tedy kostra aplikace. Po základním nastavení web serveru je nutné vytvořil modul, který tvoří základ aplikace. Celý postup je detailně popsán v dokumentaci, včetně různých možných problémů.

3.7.2 Vlastnosti a přednosti

Výhodou Zend je plně objektový zdrojový kód, který využívá nové vlastnosti PHP verze 5.3.

Další vlastností je návrhový vzor MOVE[27], který se od běžnějšího MVC modelu liší především způsobem, jakým přistupuje k událostem vyvolaným například interakcí uživatele. Některým programátorům může tento model vyhovovat více, než klasická MVC architektura.

Silné možnosti přizpůsobení jsou pro zkušené vývojáře skutečně velkou výhodou, avšak začátečníkům se tím některé úkony naopak ztíží.

3.7.3 Nevýhody

Největší nevýhodou je podpora dvou vývojových verzí Zend 1 a Zend 2. Starší verze Zend 1 bude podporována nejméně do roku 2014.

Jako nevýhodu považuji též velkou rozsáhlost celého frameworku, protože obsahuje mnoho modulů mezi kterými je často obtížné najít ten, který zrovna potřebujeme.

3.7.4 Práce s databází

Zend framework 2 používá jako základní modul pro práci s databází Zend\Db\TableGateway\TableGateway. Nicméně je možné připojit i jiné databázové abstrakce, jako Doctrine 2, Propel nebo také NotORM modely.

3.8 Symfony

Symfony framework je vyvíjen od roku 2005, jeho současná verze je 2.4.0. Je licencovaný pod svobodnou licencí MIT license.

Symfony využívá mnoho open source projektů, tím je zaručena široká základna vývojářů, kteří se podílejí na vývoji a bezpečnostních aktualizacích.

3.8.1 Stažení a instalace

Framework je možné stáhnout přímo ze stránek projektu nebo pomocí správce závislostí Composeru.

27 Models-Operation-Views-Events (česky modely-operace-pohledy-události) je softwarová architektura, která rozděluje aplikaci do čtyř různých celků.
Model obsahuje pouze informace, getry, setry, a případně i kontrolní funkce. Ale neumí data ukládat do databáze nebo na jiné externí úložiště. Operace jsou odpovědné za změnu dat v modelech, správné zobrazení pohledů a odpovídají reakci na události vyvolané interakcí uživatele. Pohledy vizualizují data například na monitor, předávají vzniklé události ke zpracování operacím a čekají na novou událost, podle které se přizpůsobí. [25]

Celý projekt si překopírujeme do webového adresáře a pomocí příkazové řádky vytvoříme nový projekt. Při jeho vytváření si můžeme nechat vygenerovat i základní kostru aplikace.

Nyní již můžeme aplikaci skrze webový prohlížeč spustit a zobrazit implicitně vygenerovanou stránku. Nastane-li nějaký problém, například špatně nastavená přístupová práva pro některé adresáře nebo chybějící modul v PHP, ladící stránka nás o tom informuje a nabídne i možné řešení.

3.8.2 Vlastnosti a přednosti

Za hlavní přednost považuji u Symfony možnost vytvoření, nastavení a spravování aplikace pomocí nástroje zvaného console. Jedná se o program napsaný v PHP určený pro příkazovou řádku. Tento nástroj můžou o svou funcionalitu obohati i jednotlivé rozšiřující moduly tzv. "bundles" (viz kapitola 4.3) a přidat další možnosti správy naší aplikace.

Veškerá nastavení jsou uložená v konfiguračních souborech ve formátech YAML, XML nebo pomocí standardních PHP souborů. Můžeme si tedy vybrat formát, který nám vyhovuje.

Symfony obsahuje debugovací nástroj, který se zcela jistě vyrovná například výše zmíněné Laděnce z frameworku Nette. Dokonce poskytuje i výkonostní informace o aplikaci, například využití paměti, počet a dobu běhu SQL dotazů nebo celkovou dobu načítaní stránky.

3.8.3 Nevýhody

Výše zmíněný program pro správu aplikace pomocí příkazové řádky můžeme považovat i za jeho nevýhodu. Jde o ojedinělý nástroj pro správu frameworku a je nutné se v něm nejprve naučit pracovat.

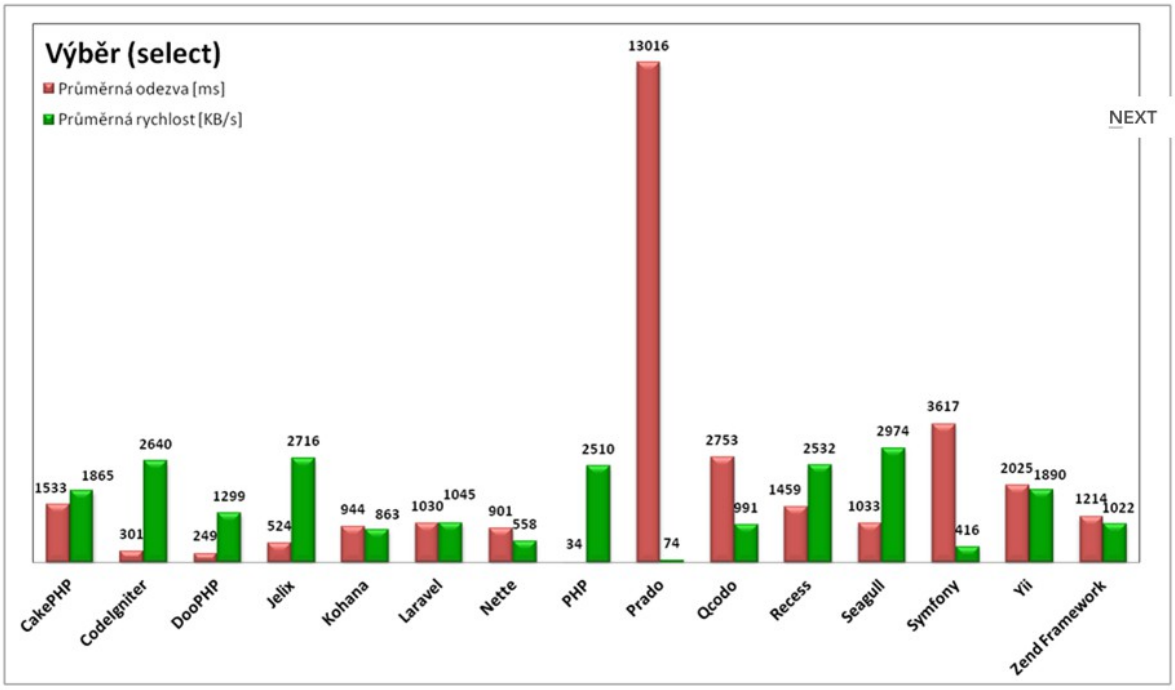
3.8.4 Práce s databází

Symfony framework využívá pro práci s databází Doctrine ORM. Avšak jde pouze o volitelné doporučení. Není problém používat, jak čisté PDO poskytované přímo PHP, tak i jinou databázovou vrstu.

3.9 Srovnání výkonu frameworků při výběrovém dotazu

Na serveru Zdroják vyšel seriál se srovnáním rychlosti různých frameworků při práci s databází, srovnání pochází z roku 2013. [26] Přestože pro můj projekt není rychlost databáze stěžejním faktorem, výsledky testu mohou posloužit jako dobré vodítko při rozhodování, který framework zvolit.

Obrázek 1: Srovnání výběrového dotazu SELECT v různých webových frameworcích



Zdroj: zdroják.cz

Při testování výkonu výběrového dotazu SELECT, každá aplikace napsaná v jednom z testovaných frameworků provedla 3000 požadavků na databázi. Stejné měření proběhlo vždy třikrát.

Průměrná odezva ukazuje čas, který potřebuje aplikace pro vykonání všech databázových požadavků. Průměrná rychlost pak uvádí, jak velké množství informace bylo přeneseno během jedné vteřiny. Nejlepších hodnot dosáhne aplikace s co nejnižší dobou odezvy a nejvyšší průměrnou rychlostí.

4 Popis vybraného řešení pro tvorbu aplikace

4.1 Technické a programové vybavení

Pro realizaci webové aplikace je použit PHP framework Symfony. Tato volba byla učiněna po vypracování srovnání frameworků vedoucím práce panem Ing. Michalem Camprem. Framework Symfony byl zvolen hlavě z důvodu, že vyniká rychlostí vývoje a celkovou robustností výsledné aplikace.

4.1.1 Vývojové prostředí

K provozu frameworku je zapotřebí webový server, programovací jazyk PHP a relační databáze MySQL. Tyto technologie byly instalovány v balíčku EasyPHP 5.4.6 pro operační systém Windows.

Pro vývoj je použit verzovací software Git. Ten je součástí NetBeans IDE určeného pro vývoj v PHP, který byl pro vývoj aplikace také použit. Jako Git repositář je použita služba Github.

Pro správu databáze byl použit nástroj phpMyAdmin [27], který je napsaný jako PHP aplikace a je dostupný skrze webový prohlížeč.

4.2 Konfigurace

Ve své aplikaci používám konfigurační soubory ke dvěma účelům. Jedním z nich jsou konfigurační soubory samotné aplikace a frameworku, na kterém je postavena. Druhým je popis jednotlivých problémů, který slouží pro generování vstupních Python skriptů.

4.2.1 Nastavení aplikace

Téměř všechna nastavení mé aplikace i samotného frameworku se provádí přes konfigurační soubory. Framework Symfony umožňuje zápis těchto souborů ve formátech XML, YAML nebo pomocí PHP polí. Ve své aplikaci jsem se rozhodl použít podle mého nejpřehlednější, a zároveň výchozí způsob zápisu ve formátu YAML.

Jedinou výjimkou je konfigurační soubor pro správce závislostí zvaného Composer. Ten vyžaduje pro jeho používání konfigurační soubor ve formátu JSON.

4.2.2 Konfigurace skriptu

Pro popis jednotlivých početních problémů jsem použil formát XML. V tomto formátu jde podle mého nejlépe vytvořit vlastní struktura, která je přehledná a srozumitelná. Uživatel, který by chtěl vytvořit vlastní popis problému, se tak v této XML struktuře velmi rychle zorientuje a popis bude schopný vytvořit.

4.3 Moduly použité v Symfony

Celé Symfony verze 2 se skládá z tzv. „bundles“, to jsou rozšiřující balíčky, které do frameworku přidávají nové funkce. O správu těchto balíčků se stará výše zmíněný nástroj pro správu závislostí Composer, který zajistí stažení všech definovaných balíčků. Rovněž se stará o jejich aktuálnost.

4.3.1 Doctrine

Framework Symfony k používání ORM nenutí, ale nabízí velmi snadnou možnost integrování s Doctrine [28], Propel [29] a dalšími. S žádným ORM jsem dosud nepracoval, zvolil jsem tedy Doctrine, který je v Symfony implicitně přednastaven.

4.3.2 FOSUserBundle

Tento balíček slouží ke správě uživatelů, obsluhuje přihlášení, registraci a další. Vyvíjí jej skupina nazývaná Friends of Symfony, zkráceně FOS. Po instalaci FOSUserBundle si vytvoříme vlastní entitu User, která je odděděná od třídy FOS\UserBundle\Model\User.

```
class User extends FOS\UserBundle\Model\User
```

Původní třída nabízí spoustu atributů, jako heslo, e-mail, uživatelské jméno atp. Rovněž se stará o šifrování hesel a správu uživatelských rolí pro možnost přístupu k různým částem aplikace. V naší třídě User vytvoříme jen ty atributy, které chceme mít přístupné navíc, oproti rodičovské.

Vytvoření nového uživatele je možné provést, pomocí nástroje „console“, registrací přes webový prohlížeč nebo pomocí tzv. data fixtures.

4.3.3 DoctrineFixturesBundle

DoctrineFixturesBundle poskytuje snadný nástroj pro nahrání prvotních nebo testovacích dat do databáze.

Umožňuje vytvořit několik záznamů téže entity a dovoluje definovat referenci na entitu, kterou je možné použít v jiných data fixtures jako odkaz na cizí klíč.

Níže uvedený zdrojový kód je funkce, která se postará o přidání jednoho uživatele do databáze.

```
public function load(ObjectManager $em) {
    $userManager = $this->container->get('fos_user.user_manager')
    // vytvoření nového uživatele
    $user = $userManager->createUser();
    // nastavení povinných atributů
    $user->setUsername('admin');
    $user->setName('Antonín');
    $user->setLastname('Neumann');
    $user->setEmail('admin@example.com');
    $user->setPlainPassword('SomeSecretPassword');
    $user->setEnabled(true);
    $user->setRoles(array('ROLE_SUPER_ADMIN'));
    // vytvoření reference pro možnost odkazování v jiných
       Data Fixtures třídách
    $this->addReference('admin', $user);
    //uložení uživatele do databáze
    $em->persist($user);
    $em->flush();
}
```

4.3.4 Twig/extensions

Twig je šablonovací systém pro PHP. Symfony umožňuje použití i standardních PHP šablon, ale Twig je rychlejší a bezpečnější. Jeho výhodou je například automatické escapování[28].

28 Escapování vstupního nebo výstupního řetězce je proces, při němž dochází k převodu znaků, které mají v daném kontextu specíální význam. [30]

Umožňuje také používání cyklů, řídících struktur nebo různých filtrů. Filtry je možné vytvářet i vlastní, v Symfony například filtr „trans“. Ten je určený pro snadný překlad prvků stránky do více jazyků a jeho možností využívám i ve své aplikaci. Ačkoli je prozatím dostupný jen anglický jazykový balíček, budoucí vytvoření českého překladu bude velmi jednoduché a rychlé.

4.3.5 GenemuFormBundle

Tento balíček rozšiřuje možnosti formulářů. Přidává nové datové typy, které Symfony implicitně nezná. Do své aplikace jsem jej přidal pro potřebu generovat ve formulářích samostatný textový popisek. Symfony tuto možnost standardně neumožňuje, a je proto nutné sáhnout po rozšíření, které to umožní.

5 Návrh databáze a jednotlivých entit

K vytvoření modelu databáze jsem použil program MySQL Workbench 6.0 CE. Základní model byl v průběhu práce ještě několikrát upraven v závislosti na potřebách aplikace.

Entita je objekt s množinou určitých hodnot – atributů, které jej popisují a množinou funkcí, které můžeme nad objektem používat. Tabulka je potom převedení entity do kontextu relačních databází. Oba tyto pojmy lze do značné míry zaměnit, jelikož oba představují popis reálného objektu.

5.1 Popis entit

Hlavními entitami jsou popis skriptů a problémů, ty jsou nezbytné pro generování formuláře, a následné spuštění výpočtu. Jeden skript umí řešit více různých problémů, takže vztah mezi těmito tabulkami je 1:N.

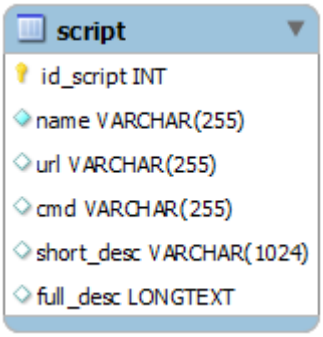
Pro možnost zobrazení textového obsahu byla použita entita Page, která představuje obecné stránky, a entita Doc, jež reprezentuje uživatelskou dokumentaci. Rozdělení obsahu do dvou tabulek je provedeno záměrně, protože dokumentace má být reprezentována ve stromové struktuře. Kvůli chybě v rozšíření Gedmo prozatím k implementaci nedošlo a stránky dokumentace jsou prezentovány stejně jako obecné stránky aplikace.

K zajištění správy webové aplikace je zapotřebí ještě entita User. Ta umožní oprávněným uživatelům možnost přidávat nebo upravovat jednotlivé stránky, scripty i problémy. Běžným uživatelům zajistí možnost ukládání jednotlivých výpočtů pro pozdější použití. Realizace této entity je zajištěna rozšířením entity User z bundlu FOSUserBundle.

Entita Usersave umožňuje přihlášeným uživatelům uložit si vygenerované Python scripty, které lze již přímo použít pro zahájení opakovaného výpočtu, bez nutnosti znovu vyplňovat hodnoty ve formuláři.

Pro možnost jednoduchých statistik je použita entita Log, která umožňuje administrátorovi zobrazit přehled o tom jaký uživatel, kdy a jaký problém spustil a jestli byl proveden úspěšně nebo neúspěšně.

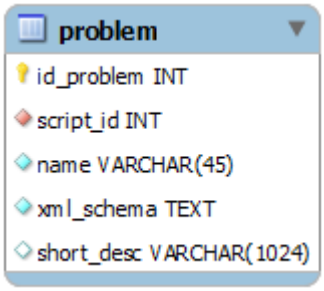
5.2 Entita Script



Tato entita popisuje jednotlivé skripty. Ty od sebe odlišuje jméno (*name*), které slouží jako identifikátor při definici problému. Krátký (*short_desc*) a dlouhý (*full_desc*) popis jsou určeny pro prezentování informací, co daný skript počítá, jaké vyžaduje parametry nebo v čem je jiný, než ostatní.

URL[29] adresa definuje. kde lze skript nalézt. V případě běhu skriptu na stejném serveru, jako webová aplikace, se uvádí localhost. Atribut cmd (*command*) je příkaz pro spuštění skriptu, včetně všech potřebných vstupních parametrů.

5.3 Entita Problem



Každý problém je závislý na nějakém skriptu, který jej řeší. Tabulka proto obsahuje cizí klíč s referencí na entitu Script. Atribut jméno (*name*) slouží obdobně jako u skriptů ke vzájemnému odlišení jednotlivých problémů. Atribut *short_desc* slouží pro krátký popis a může uživateli pomoci s výběrem správného problému.

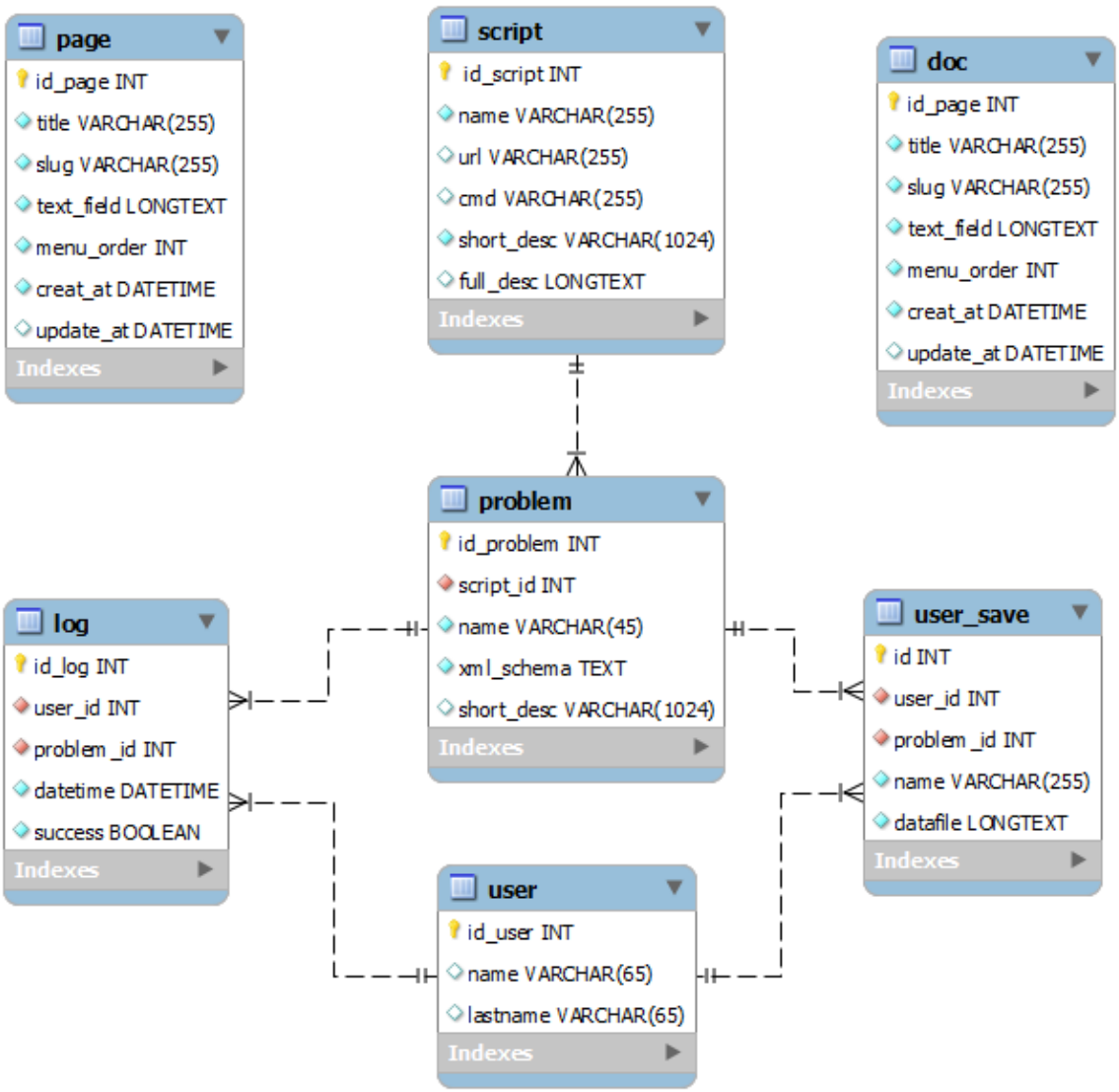
Klíčovým je však atribut *xml_schema*, ten obsahuje popis problému ve formátu XML použitého pro generování formuláře a výsledného vstupního Python skriptu.

29 Uniform Resource Locator slouží k jednoznačné identifikaci a přístupu ke zdrojům na internetu. [31]

5.4 Schéma databázového modelu

Výsledné schéma (viz obrázek 2) popisuje datový model databáze včetně všech relačních vztahů. Identifikátory uvozené symbolem klíče představují primární klíč. Pro cizí klíč je použit symbol červeného kosočtverce.

Obrázek 2: Datový model databáze



Zdroj: Vlastní, 2014

5.5 Ukázka definice entity v Doctrine2 ve formátu YAML

Plné definice entit Problem a Script jsou uvedeny v příloze A, zde je pouze ukázka, jak definice entity vypadá a jaké jsou její hlavní části.

```
Muj\VlastniBundle\Entity\Ukazka:
    type: entity
    table: ukazka
        #zde je definice primárního klíče
    fields:
        nazev_sloupce:
            type: string
            length: 255
            unique: true
            nullable: true
    oneToMany:
        nazev_sloupce:
            targetEntity: NazevCiloveEntity
            mappedBy: ukazka
            joinColumn:
                onDelete: CASCADE
    manyToOne:
        nazev_sloupce:
            targetEntity: NazevCiloveEntity
            inversedBy: ukazka_id
            joinColumn:
                name: nazev_fk_sloupce_id
                referencedColumnName: id
                onDelete: CASCADE
```

Při definování entity ve formátu YAML je důležité dodržet formát zápisu. Každá další úroveň musí být o té předchozí oddělena dvěma značkami pro tabulátor. Hodnota se píše na stejný řádek jako její identifikátor.

Na první úrovni je celé jméno entity včetně jmenného prostoru. Druhá úroveň obsahuje typ předchozí úrovně. Ten je zde proto, že v jednom YAML souboru můžeme definovat více entit najednou. Identifikátor „table“ určuje jméno tabulky v databázi a dále jméno a vlastnosti primárního klíče. „Fields“ obsahuje definice sloupců tabulky. Každý sloupec má svůj datový typ, a případně doplňující informace, jako rozsah datového typu, unikátnost obsahu nebo nutnost vyplnění hodnoty. Posledními identifikátory jsou „oneToMany“ a „manyToOne“, jež představují relační vlastnosti tabulky, tedy propojení této tabulky s jinými.

6 Definice a popis problémů

Jednotlivé početní problémy jsou popsány jazykem XML. Tento popis je uložen u každé entity Problem, jako její atribut a je možné jej upravovat přes webové administrační rozhraní.

6.1 Návrh struktury XML

Protože si v XML můžeme definovat vlastní značky tzv. tagy, existuje několik pravidel, která je potřeba dodržet, aby našemu XML rozuměly programy určené pro zpracování XML souborů.

Splňuje-li XML všechna následující pravidla, nazýváme jej správně strukturovaný (anglicky well-formed). Tyto pravidla jsou uvedena v XML specifikaci publikované W3C [32] [33].

- Každý XML dokument začíná deklarací,
- každý XML dokument obsahuje jeden kořenový element a všechny ostatní elementy jsou obsaženy v jeho těle,
- každý XML tag je řádně ukončen,
- je-li element obsažený v těle jiného elementu, pak je zde obsažen celý,
- jednotlivé tagy se nesmějí křížit.

Mnou navržený XML dokument (vzor viz příloha B.1) pro definici problému lze spravovat přes webové administrační rozhraní (ukázka viz příloha C.2). Definice obsahuje kořenový element <*problem*> jehož hodnotou jsou právě dva další elementy <*variables*> a <*form*>.

Kořenový element obsahuje dva atributy. Jedním z nich je jméno (*name*) problému a druhý (*script*) obsahuje název skriptu, který umí daný problém vyřešit.

V elementu <*variables*> jsou obsaženy jednotlivé proměnné, které definují daný problém pod značkou <*var*>. Tato značka obsahuje atribut *type*, definující datový typ proměnné. Přípustné datové typy jsou celé číslo (*int*), reálně číslo (*float*), textový řetězec (*str*) a řetězec pro import knihoven (*import*). Dále atribut *name*, který definuje jméno proměnné. Jednotlivé elementy <*var*> mohou být seskupovány, pokud je hodnota jejich atributu *name* stejná.

Element <*form*> slouží k definici formuláře, který uživatelům umožní volbu definovaných vstupních parametrů problému. Může obsahovat dva další elementy <*text*> a <*field*>.

Tag <text> je použit pro zobrazení obyčejného textu nebo podnadpisu ve formuláři. Obsahuje pouze atribut *type* nabývající hodnot *simple* pro obyčejný text a *subtitle* pro podnadpis. Hodnota tohoto elementu je poté zobrazena textem a formátována podle definovaného typu.

Značka <*field*> definuje samotná formulářová pole a obsahuje atributy *type* a *name*. Hodnota elementu je pak ve formuláři použita jako popis (HTML tag <*label*>) tohoto pole.

Atribut *type* může nabývat hodnot, celé číslo (*int*), reálné číslo (*float*), textový řetězec (*str*), logická hodnota (*bool*), trojrozměrný vektor (*vect*) nebo seznam hodnot (*combo*). Typ určuje výslednou podobu formulářového pole, například *vect* se zobrazí jako tři vstupní pole akceptující reálné číslo. A rovněž určuje datový typ vstupní hodnoty tohoto pole, které je při zpracování formuláře kontrolováno, například pro typ *int* není možné zadat hodnotu „text“.

Je-li typ definovaný jako *combo*, obsahuje tento element ve svém těle ještě tagy <*opt*> s definicí přípustných hodnot a celý element je interpretován jako výběrové tlačítko tzv. „select box“ (též „combo box“).

6.2 Implementace parseru pro převod XML

Pro převod problémů definovaných v XML, podle výše uvedené struktury do proměnných použitelných v programovacím jazyce PHP, jsem použil knihovnu Ganon [34], se kterou jsem pracoval na základě předchozích zkušeností s ní. Ganon umožňuje velmi snadný přístup k jednotlivých elementům a jejich atributům. Mezi elementy lze i vyhledávat podle různých kritérií.

Knihovna Ganon není objektově orientovaná, takže pouze stačí načíst XML soubor pomocí metody *str_get_dom($xml)*. Převod XML dokumentu sestává ze tří základních částí. Nejdříve získám hodnoty všech atributů elementu *problem*. V druhé získám hodnoty všech atributů a obsah elementu *var*. Třetí část slouží pro získání obsahu a atributů polí *field* a *text*. Druhá i třetí část je procházena v cyklu pro všechny elementy daného typu. Ve třetí části je potřeba dát si pozor na elementy typu *combo*, jejichž obsah tvoří ještě další elementy a je nutné procházet je v dalším cyklu.

Každá část je nakonec předána jako samostatné pole do třídy starající se o generování samotného formuláře. K těmto třem polím je přidána ještě proměnná obsahující *id* problému, ze kterého byla pole „parsována“.

6.3 Generování formuláře

Proměnná s *id* problému, jméno skriptu a všechny prvky pole získaného z původního XML elementu *<variables>,* jsou generována jako skrytá pole a použita až při zpracování formuláře. V HTML se skryté pole definuje jako *<input type='hidden'>*.

Prvky pole s hodnotami definujícími jednotlivá formulářová pole, jsou v cyklu procházeny a interpretovány podle hodnoty prvku s klíčem *type*. Klíč odpovídá původní hodnotě téhož jména v XML dokumentu.

Náhled kompletního vygenerovaného formuláře je možné vidět v příloze C.1.

6.4 Generování Python skriptu a spuštění výpočtu

Po vyplnění hodnot formuláře uživatelem, jsou všechny hodnoty zkontrolovány. Pokud je obsah nějakého pole vyplněn špatně, například zadaná hodnota neodpovídá definovanému datovému typu, je uživatel přesměrován zpět na formulář a je mu oznámena chyba, které se při vyplňování dopustil.

Je-li kontrola provedena úspěšně, je z těchto dat vygenerován odpovídající vstupní Python skript, který je uložen do souboru input_HASH.py. *HASH* představuje jednoznačný identifikátor vygenerovaný funkcí *md5()*. Vstupem této funkce je kombinace aktuálního času, získaného funkcí *time()* a obsahu skriptu.

Vytvořený soubor je předán k výpočtu příkazem *"\$HOME/run_sfepy HASH“* na server s adresou uvedenou v atributu *url* v entitě Script. Jelikož na serveru (počítači), na kterém běží knihovna SfePy není dostupný žádný webový server, probíhá přihlašování k tomuto počítači prostřednictvím protokolu ssh. Server má rovněž přístup k počítači, na kterém běží webová aplikace, a dokáže si vstupní soubor s názvem *input_HASH.py* ze složky */volume1/web/masscom/sfepy_input/* vyzvednout sám. Do stejné složky v případě úspěšného dokončení výpočtu uloží obrázek *output_HASH.png*, který ukazuje výsledek výpočtu. Tento obrázek je následně zobrazen uživateli přímo ve webovém prohlížeči (viz příloha C.3).

7 Závěr

V úvodu práce se zabývám rozborem technologií umožňujících tvorbu webových aplikací. Hlavní důraz je kladen na výběr technologií, které jsou pro aplikaci použity.

V práci jsem měl navrhnout databázi potřebnou k online správě prezentaci Python skriptů a tuto databázi poté implementovat ve frameworku Symfony2 jako rozšiřující balíček, tzv. „bundle“. Návrh i výsledná implementace je plně funkční a splňuje všechny požadavky, které byly pro webové rozhraní definované.

Implementace měla být zaměřena primárně na knihovnu SfePy s možností pozdějšího přidání dalších knihoven. Při použití jiné knihovny bude potřeba upravit „parser“ XML schémat tak, aby generoval vstupní formát akceptovaný touto knihovnou nebo přizpůsobit vstupní soubor knihovny, aby odpovídal formátu, který již aplikace generuje.

Aplikace byla otestována na lokálním webovém serveru v operačním systému Windows 7 a GNU/Linux Ubuntu, kde fungovala dle zadání. Nyní běží v produkčním nasazení na NAS[30] od společnosti Synology. Rovněž byla aplikace otestována pomocí základních unit testů napsaných panem Ing. Michalem Camprem, které testovaly zejména dostupnost a autorizaci pro jednotlivé stránky.

Při realizaci mé bakalářské práce jsem získal množství zkušeností s tvorbou webových aplikací ve frameworku Symfony2, s objektově relačním modelem práce s databází v podobě Doctrine2 a prohloubil své znalosti o XML jazyce.

Aplikace, stejně jako text práce, jsou volně přístupné prostřednictvím Git repositáře na stránkách Github [35].

30 Network Attached Storage

Seznam literatury

[1] Robert Cimrman a kolektiv, Sfepy [online]. Dostupné z <http://sfepy.org/doc-devel/index.html> [cit. 5. 3. 2014]

[2] On SGML and HTML [online]. Dostupné z <http://www.w3.org/TR/REC-html40/intro/sgmltut.htmhttp://www.w3.org/TR/REC-html40/intro/sgmltut.html> [cit. 3. 4. 2014]

[3] Tim Berners-Lee, Tim Berners-Lee [online]. Dostupné z <> [cit. 2014]

[4] HTML & CSS [online]. Dostupné z <http://www.w3.org/standards/webdesign/htmlcss#whatcss> [cit. 7. 4. 2014]

[5] Ian Pouncey, Richard York, Beginning CSS: Cascading Style Sheets for Web Desig,John Wiley & Sons , 2011. ISBN 9781118121788

[6] R. Fielding a kolektiv, Hypertext Transfer Protocol -- HTTP/1.1 [online]. Dostupné z <https://tools.ietf.org/html/rfc2616; https://cs.wikipedia.org/wiki/HTTP> [cit. 7. 4. 2014]

[7] Kolektiv, A re-introduction to JavaScript (JS Tutorial) [online]. Dostupné z <https://developer.mozilla.org/en-US/docs/Web/JavaScript/A_re-introduction_to_JavaScript> [cit. 2014]

[8] JavaScript Libraries [online]. Dostupné z <http://www.w3schools.com/js/js_libraries.asp> [cit. 12 .4. 2014]

[9] W. Jason Gilmore, Beginning PHP and MySQL: From Novice to Professional,Apress , 2010. ISBN 9781430231141

[10] PHP 5.3 change log [online]. Dostupné z <http://docs.php.net/ChangeLog-5.php#5.3.0> [cit. 2009]

[11] Get Started with ASP.NET Web Sites [online]. Dostupné z <http://www.asp.net/get-started/websites> [cit. 2014]

[12] Derek R. Price, Overview [online]. Dostupné z <http://ximbiot.com/cvs/manual/cvs-1.12.12/cvs_1.html> [cit. 2014]

[13] Jan Štráfelda, Server [online]. Dostupné z <http://www.adaptic.cz/znalosti/slovnicek/server/> [cit. 24. 4. 2014]

[14] Hypertext Transfer Protocol -- HTTP/1.1 [online]. Dostupné z <http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html> [cit. 24. 4. 2014]

[15] Netcraft, April 2014 Web Server Survey [online]. Dostupné z <http://news.netcraft.com/archives/category/web-server-survey/> [cit. 5. 5. 2014]

[16] Chris Pitt, Pro PHP MVC,Apress , 2012. ISBN 9781430241652

[17] Kohana documentation [online]. Dostupné z <http://kohanaframework.org/documentation> [cit. 12. 4. 2014]

[18] Andrew Coulton, Doctrine 2 [online]. Dostupné z <https://github.com/ingenerator/kohana-doctrine2> [cit. 17. 4. 2014]

[19] Samnan ur Rehman Akhoond, Propel [online]. Dostupné z <https://github.com/Samnan/Kohana-Propel> [cit. 17. 4. 2014]

[20] Nils Adermann, Jordi Boggiano, Composer: Dependency Manager for PHP [online]. Dostupné z <https://getcomposer.org/> [cit. 5. 5. 2014]

[21] Jakub Vrána, Cross Site Scripting [online]. Dostupné z <http://php.vrana.cz/cross-site-scripting.php> [cit. 4. 5. 2014]

[22] Jakub Vrána, Cross-Site Request Forgery [online]. Dostupné z <http://php.vrana.cz/cross-site-request-forgery.php> [cit. 4. 5. 2014]

[23] David Grudl, Nette Framework [online]. Dostupné z <http://nette.org/cs/> [cit. 4. 5. 2014]

[24] David Grudl, Neon [online]. Dostupné z <www.ne-on.org> [cit. 4. 5. 2014]

[25] Conrad Irwin, MVC is dead, it's time to MOVE on [online]. Dostupné z <http://cirw.in/blog/time-to-move-on> [cit. 14. 3 2014]

[26] Jaroslav Jakoubě, Seriál: ORM test PHP frameworků [online]. Dostupné z <http://www.zdrojak.cz/serialy/test-php-frameworku/> [cit. 4. 5. 2014]

[27] Bringing MySQL to the web [online]. Dostupné z <http://www.phpmyadmin.net/> [cit. 1. 5. 2014]

[28] Doctrine Project [online]. Dostupné z <http://www.doctrine-project.org/> [cit. 5. 5. 2014]

[29] Propel ORM [online]. Dostupné z <http://propelorm.org/> [cit. 5 .5. 2014]

[30] David Grudl, ESCAPOVÁNÍ - DEFINITIVNÍ PŘÍRUČKA [online]. Dostupné z <http://phpfashion.com/escapovani-definitivni-prirucka> [cit. 5. 5. 2014]

[31] Tim Berners-Lee, Uniform Resource Locators [online]. Dostupné z <http://www.w3.org/Addressing/URL/url-spec.txt> [cit. 21 March 1994]

[32] Tim Bray a kolektiv, Extensible Markup Language (XML) 1.0 (Fifth Edition) [online]. Dostupné z <http://www.w3.org/TR/REC-xml/> [cit. 6. 5. 2014]

[33] Jiří Kosek, XML pro každého,Grada Publishing , 2000. ISBN 80-7169-860-1

[34] A. D. Niels , Ganon [online]. Dostupné z <https://code.google.com/p/ganon/> [cit. 6. 5. 2014]

[35] Antonín Neumann, Zdrojové kódy aplikace Masscom [online]. Dostupné z <https://github.com/tonda13/bp.git> [cit. 7. 5. 2014]

Seznam obrázků

Obrázek 1: Srovnání výběrového dotazu SELECT v různých webových frameworcích…15
Obrázek 2: Datový model databáze.....22

Seznam tabulek

Tabulka 1: Zastoupení webových serverů na celkovém internetovém provozu.....6

Seznam zkratek

HTML	HyperText Markup Language
CSS	Cascading Style Sheets
WWW	World Wide Web
SGML	Standard Generalized Markup Language
XML	eXtensible Markup Language
HTTP	HyperText Transfer Protocol
DOM	Document Object Model
AJAX	Asynchronous JavaScript and XML
PHP	PHP: Hypertext Preprocessor, původně Personal Home Page
ASP	Active Server Pages
RoR	Ruby on Rails
JSP	JavaServer Pages
JSON	JavaScript Object Notation
YAML	YAML Ain’t Markup Language
SVN	Apache Subversion
CVS	Concurrent Versions System
ORM	Object-relational mapping
MVC	Model-view-controler
HMVC	Hierarchical MVC
PDO	PHP Data Objects
CSFR	Cross-Site Request Forgery
XSS	Cross Site Scripting
MOVE	Models-Operation-Views-Events
URL	Uniform Resource Locator
NAS	Network Attached Storage

Příloha A – definice entit v YAML

A.1 Definice entity Problem

```
sfepy\MasscomBundle\Entity\Problem:
    type: entity
    table: problem
        id:
            id:
                type: integer
                generator: { strategy: AUTO }
    fields:
        name:
            type: string
            length: 255
            unique: true
        xml_schema:
            type: text
        short_desc:
            type: string
            length: 1024
            nullable: true
    oneToMany:
        user_save:
            targetEntity: Usersave
            mappedBy: problem
            joinColumn:
                onDelete: CASCADE
        log:
            targetEntity: Log
            mappedBy: problem
            joinColumn:
                onDelete: CASCADE
    manyToOne:
        script:
            targetEntity: Script
            inversedBy: problem_id
            joinColumn:
                name: script_id
                referencedColumnName: id
                onDelete: CASCADE
```

A.2 Definice entity Script

```
sfepy\MasscomBundle\Entity\Script:
    type: entity
    table: script
        id:
            id:
                type: integer
                generator: { strategy: AUTO }
    fields:
        name:
            type: string
            length: 255
            unique: true
        url:
            type: string
            length: 255
            nullable: true
        cmd:
            type: string
            length: 255
            nullable: true
        short_desc:
            type: string
            length: 1024
        full_desc:
            type: text
            nullable: true
    oneToMany:
        problem_id:
            targetEntity: Problem
            mappedBy: script
            joinColumn:
                onDelete: CASCADE
```

Příloha B – definice početních problémů v XML

B.1 Vzor pro definici početních problémů

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--
    NAME_OF_PROBLEM - jméno problému je definováno jako kombinace
    problému a geometrie

    NAME_OF_SCRIPT - jméno skritpu, ke kterému je daný problém
    přidružen
-->

<problem name="NAME_OF_PROBLEM" script="NAME_OF_SCRIPT">
    <!--
    definice proměnných, které patří k danému problému
    např. diferenciální rovnice:

    type - datový typ proměnné
        int - integer [celé číslo]
        float - float/double [reálné číslo]
        str - string [řetězec]
        import - pro potřeby importování knihoven do Python
scriptů       (tato proměnná by měly být uváděny jako první
unvitř tagu      <variables>)

    name - pouze [A-Z,a-z,0-9,_,-]

    VALUE - hodnota proměnné
    -->

    <variables>
        <!-- proměnná typu import -->
        <var type="import">from utils import input_fce</var>

        <!-- proměnná typu string -->
        <var type="str" name="equations">neco * neco -
neco...</var>

        <!-- proměnná typu integer -->
        <var type="int" name="">VALUE</var>

        <!-- proměnná typu float/double -->
        <var type="float" name="">VALUE</var>
    </variables>
```

```
<!--
A) definice formulářových polí
    type - datový typ proměnné, může nabývat hodnot:
        int - integer [celé číslo]
        float - float/double [reálné číslo]
        str - string [řetězec]
        bool - boolean [logická proměnná]
        vect - vector (x, y, z)
        combo - výběrové tlačítko

    LABEL - popisek, který se zobrazí před formulářovým políčkem

    name - proměnná, kterou se řídí zpracovávající skript
           (bez háčků a mezer - pouze [A-Z,a-z,0-9,_,-])


B) definice obyčejného textu
    type - typ textu
        subtitle - zvětšený text, slouží jako podnadpis
jednotlivých formulářových polí
        simple - obyčejný text
-->

    <form>
        <field type="" name="">LABEL</field>

        <!-- příklad celočíselné proměnné -->
        <field type="int" name="alfa">Nějaký úhel</field>

        <!-- příklad podnadpisu -->
        <text type="subtitle">Toto se zobrazí jako
podnadpis</text>

        <!-- příklad obyčejného textu -->
        <text type="simple">Toto se zobrazí jako obyčejný
        text.</text>

        <!--
        příklad comboboxu / selectboxu
            LABEL – text který v select boxu uvidí uživatel
            value - tato hodnota se pošle do scriptu
        -->
        <field type="combo" name="volba" label="Nějaká volba">
            <opt value="1">Volba 1</opt>
            <opt value="2">Volba 2</opt>
            <opt value="3">LABEL</opt>
        </field>
    </form>
<problem>
```

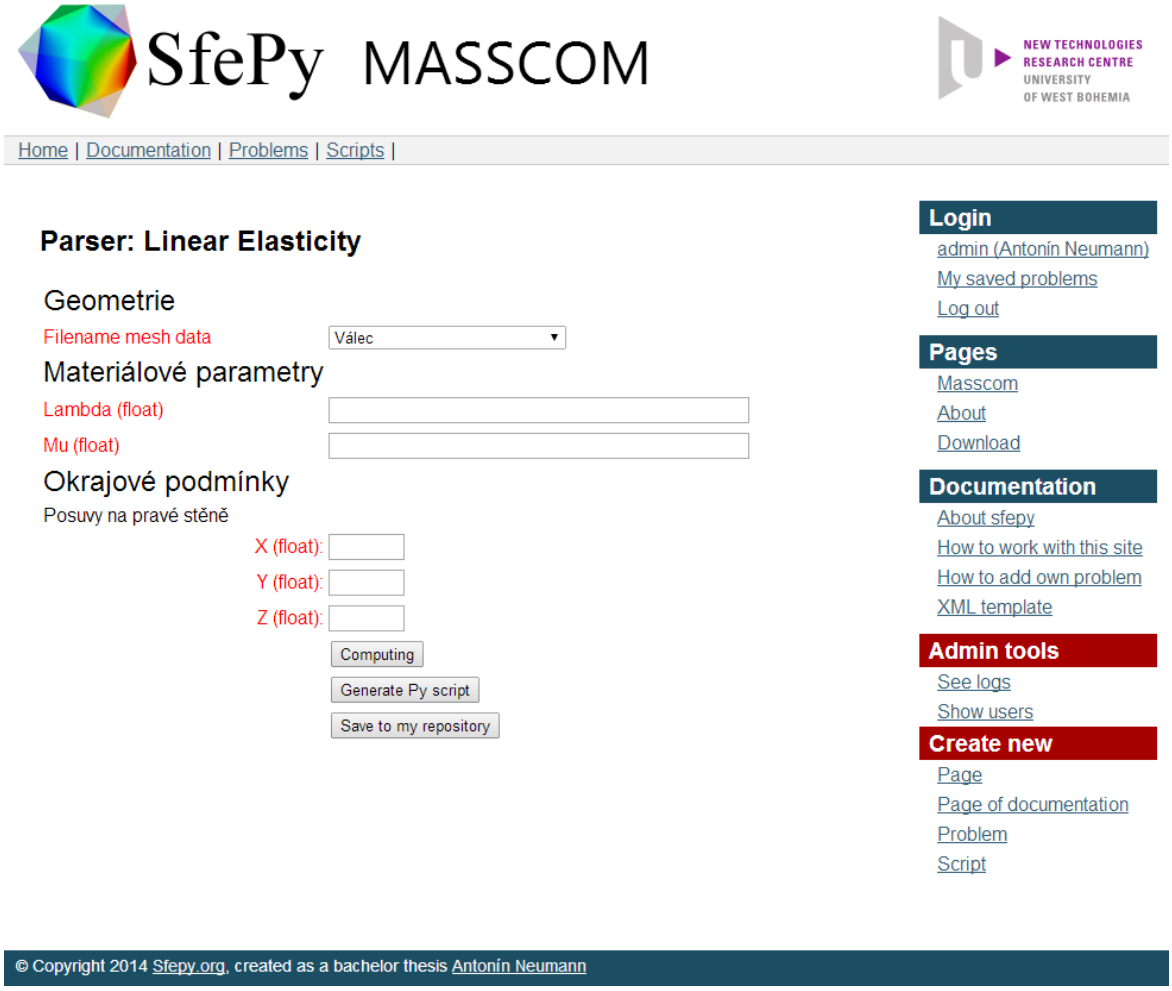
B.2 Definice problému Linear Elasticity

```
<?xml version="1.0" encoding="UTF-8" ?>
<problem name="Linear Elasticity" script="linear_elastic">
   <variables>

      <!-- oblasti -->
      <var type="str" name="regions">'Omega': 'all'</var>
      <var type="str" name="regions">
         'Left': ('vertices in (x < %region_left)',
'facet')</var>
      <var type="str" name="regions">
         'Right': ('vertices in (x > %region_right)',
         'facet')</var>

      <!-- materiál -->
      <var type="str" name="materials">
         'solid: ({'lam': %lambda, 'mu': %mu},)</var>

      <!-- pole (posuvy, rychlosti, teplota, ...) -->
      <var type="str" name="fields">
         'displacement': ('real', 'vector', 'Omega', 1)</var>

      <!-- neznámé, testovací a parametrické proměnné -->
      <var type="str" name="variables">
         'u': ('unknown field', 'displacement')</var>
      <var type="str" name="variables">
         'v': ('test field', 'displacement', 'u')</var>

      <!-- definice integrálů -->
      <var type="str" name="integrals">'i': ('v', 1)</var>

      <!-- okrajové podmínky -->
      <var type="str" name="ebcs">
         'Fixed': ('Left', {'u.all': 0.0})</var>
      <var type="str" name="ebcs">
         'Displaced': ('Right', {'u.0': %u_rightX,
         'u.1': %u_rightY, 'u.0': %u_rightZ})</var>

      <!-- rovnice problému -->
      <var type="str" name="equations">
         'balance_of_forces': """dw_lin_elastic_iso.i.Omega
         ( solid.lam, solid.mu, v, u ) = 0"""</var>

      <!-- použité řešiče -->
      <var type="str" name="solver">
         'ls': ('ls.scipy_direct', {})</var>
```

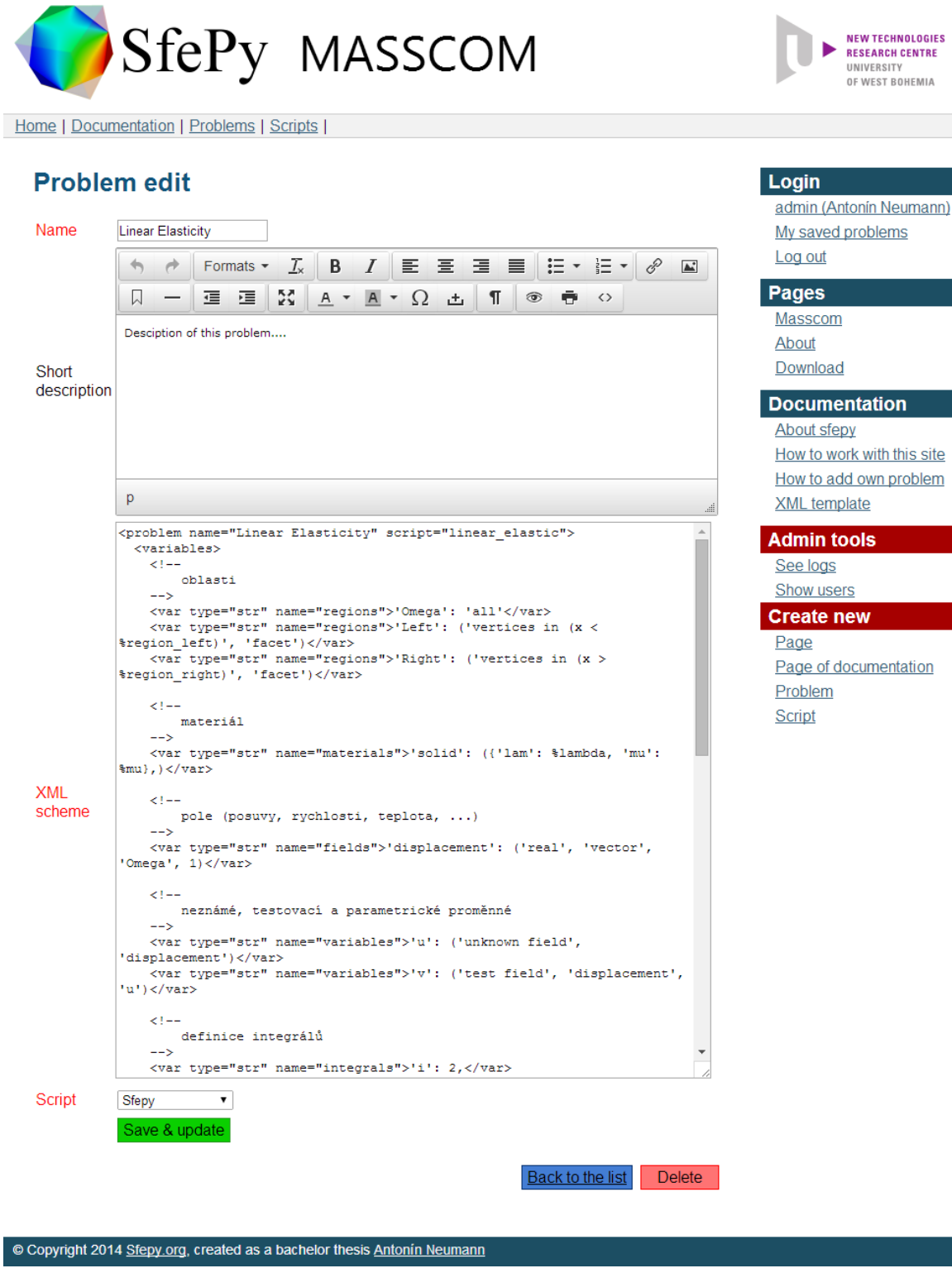
```
        <var type="str" name="solver">
            'newton': ('nls.newton', {'eps_a': 1e-10, 'eps_r':
1.0,          'check': 0, 'problem':'nonlinear'})</var>
    </variables>

    <form>
        <text type="subtitle">Geometrie</text>

        <field type="combo" name="filename_mesh"
label="Geometrie">
            <opt value="meshes/3d/cylinder.mesh"
region_left="0.001"             region_right="0.099">Válec</opt>

            <opt value="meshes/3d/cube_medium_hexa.mesh"
            region_left="-0.499"
region_right="0.499">Krychle</opt>

            <opt value="meshes/3d/block.mesh" region_left="-4.99"
            region_right="4.99">Kvádr</opt>
        </field>

        <!-- volba materiálových parametrů -->
        <text type="subtitle">Materiálové parametry</text>
        <field type="float" name="lambda">Lambda</field>
        <field type="float" name="mu">Mu</field>

        <!-- okrajové podmínky - posuv na pravé hranici -->
        <text type="subtitle">Okrajové podmínky</text>
        <field type="vect" name="u_right">
            Posuvy na pravéstěně</field>
    </form>
<problem>
```

Příloha C – ukázky webového rozhraní pro práci s XML

C.1 Ukázka vygenerovaného formuláře



C.2 Ukázka správy XML u entity Problem



C.3 Ukázka vizualizace výsledku