VYSOKÁ ŠKOLA EKONOMICKÁ V PRAZE

Fakulta informatiky a statistiky

Katedra informačního a znalostního inženýrství



Analýza nedostatků populárních content management systémů a návrh jejich řešení

Bakalářská práce

Vojtěch Kohout

Vedoucí práce: Ing. Tomáš Kliegr

prosinec 2010

Prohlášení

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a že jsem uvedl všechny použité prameny a literaturu, ze kterých jsem čerpal.

Ve Vimperku dne 6. 12. 2010

Vojtěch Kohout

Abstrakt

Cílem práce je poukázat na nedostatky vybraných populárních CMS systémů a navrhnout a naimplementovat vlastní jádro rozšiřitelného CMS systému. Takový systém, postaven na moderním PHP frameworku, může být při zakázkové tvorbě webů pro webdesignery výhodnější než populární volně dostupné CMS systémy, degradované mnohými nedostatky.

V první části práce jsou analyzovány tři populární volně dostupné CMS systémy: Drupal, CMS Made Simple a TYPOlight, a to z několika nezávislých pohledů.

V úvodu druhé části jsou prezentovány výhody, které vyplývají z použití moderního PHP frameworku a následně je navržen koncept základu CMS systému ve frameworku Nette.

Přílohou práce je kompletní implementace navrženého systému.

Návrh a implementace CMS systému ve frameworku Nette je originálním přínosem práce.

Abstract

The main goal of this thesis is to reveal significant imperfections of the popular open-source Content Management Systems (CMS) and to design and implement a core of a custom extensible CMS system. Such a system, based on a modern PHP framework, could be a better choice than open-source CMS systems for webdesigners using it for commercial purposes.

There are three open-source CMS systems analyzed from various perspectives in the first chapter: Drupal, CMS Made Simple and TYPOlight.

In the second chapter there are introduced benefits from using a modern PHP framework and a custom CMS system is designed. It is completely implemented in the appendix of the thesis.

The most important and the most original contribution of this thesis is a design and implementation of a core of a custom CMS system in Nette Framework.

Poděkování

Rád bych poděkoval Ing. Tomáši Kliegrovi za velmi vstřícné vedení mé bakalářské práce. Jsem mu zavázán za množství užitečných rad a podnětných připomínek.

Dále bych chtěl poděkovat své manželce za její bezednou trpělivost a nikde nekončící láskyplnost, kterou mě po dobu psaní této práce zahrnovala. V neposlední řadě bych chtěl poděkovat také svým dvěma půlročním dětem za nevyčerpatelný zdroj energie, kterým mi doslova ve dne i v noci byly jejich úsměvy, a také Bohu za to, že je mi pevnou skalou pod nohama.

Obsah

Úvod 7
1. Cíle práce 8
1. Populární webové CMS systémy 10
1.1. Prameny častých nedostatků 10
1.1.1. PHP – jak z neobjektového objektový jazyk vzešel 11
1.1.2. Technické možnosti versus požadavky na CMS systémy: spojené nádoby 13
1.2. Charakteristika vybraných populárních systémů 14
1.2.1. Komentář k výběru 14
1.2.2. Metodika hodnocení systémů 15
1.2.3. Drupal 17
1.2.4. CMS Made Simple 26
1.2.5. TYPOlight 30
2. Vlastní CMS systém jako řešení 32
2.1. Výhody systémů postavených na robustních frameworcích 33
2.2. Nástin implementace CMS systému v Nette Frameworku 35
2.2.1. Navrhujeme systém 36
2.2.2. Schéma datového úložiště 38
2.2.3. Využití Nette prezenterů 40
2.2.4. Router s napojením na databázi 41
2.2.5. Struktura prezenterů a modulů v administraci 43
2.2.6. Detail implementace 44
2.2.7. Závěr 45
3. Závěr 46
Literatura 48
A. DVD s implementací CMS systému podle návrhu 51

Úvod

Kdybychom přenesli skupinu dnešních absolventů vysokých škol řekněme o dvě generace nazpět v čase a poslali je na pohovor do personální agentury, jejich specializace a zaměření by ve tvářích personalistů zajisté zanechaly nemálo údivů. A přední příčku v „podivínství“ by pravděpodobně obsadil informatik, z jehož výčtu ovládaných jazyků, platforem, frameworků, databázových systémů, návrhových vzorů a mnoho dalšího by personalisté porozuměli snad jen spojkám a slovesům.

Informační technologie jsou bezesporu jednou z nejprogresivněji se vyvíjejících disciplín dnešní doby. A ruku v ruce s nimi se rozvíjí i Internet, informační médium 21. století, internetové technologie a zejména díky komerční sféře i požadavky na ně.

Velmi důležitou roli ve světě dnešního Internetu hrají webové Content Management Systémy, zkráceně CMS systémy, což jsou systémy pro správu obsahu webových prezentací. Ty ve svém nejběžnějším pojetí umožňují pohodlnou správu obsahu webových stránek a zároveň se tento úkon snaží učinit co možná nejjednodušším a přístupným co nejširší řadě webmasterům. Různé implementace CMS systémů se poté vyznačují mimořádnou variabilitou – jak z hlediska technického, tak i administrátorského.

Pro zvolené téma práce jsem se rozhodl na základě zkušeností, kterých jsem nabyl během několika posledních let strávených v pozici webdesignera na volné noze a externího PHP vývojáře a kodéra jedné nejmenované společnosti. Během své praxe jsem poznal, že podstatná většina všech zadání, se kterými se setkávám, vede na určitou variaci CMS systému a není vždy jednoduché rozhodnout se, zda pro daného klienta pouze nasadit a nakonfigurovat některé z hotových a volně dostupných CMS řešení, anebo zda pro jeho potřeby připravit řešení na míru.

1. Cíle práce

Prvním cílem práce je poukázat na to, že i přes svou značnou popularitu mají nejrozšířenější volně dostupné webové CMS systémy množství podstatných nedostatků. Neexistuje mezi nimi „ideální systém“, a proto není pro subjekty (firmy i jednotlivce) zabývající se webdesignem (dále jen pro „wedesignery“) vůbec jednoduché zvolit hotový systém, který by optimálně řešil běžné požadavky jejich klientů. V práci poukážeme nejen na samotné nedostatky, ale v obecné rovině se zamyslíme i nad jejich možnými příčinami.

Analyzovány budou tři populární volně dostupné webové CMS systémy, a to ze třech úhlů pohledu. Systémy byly vybrány tak, aby každý představoval poněkud odlišný typ (zejména z hlediska celkové koncepce).

Poznámka

Ve své práci jsem se rozhodl zabývat pouze systémy napsanými v jazyce PHP. Rozhodl jsem se tak v první řadě z důvodu jeho velké popularity a suverénně největší rozšířenosti minimálně na doménách cz. Podle průzkumu z února 2010 je totiž PHP využíváno na 48% cz domén, což mu uděluje velký náskok před druhou nejrozšířenější platformou .NET s 16%. Pro zajímavost, jazyk Python je podle studie využíván na 3% domén a Java na pouhém 1%[1].

O odhalenými nedostatky se lze vypořádat různými způsoby v závislosti na jejich charakteru. Práce si bere za cíl prezentovat jedno značně univerzální řešení, ke kterému se množství webdesignerů utíká, a to vývoj vlastního, dobře rozšiřitelného jádra CMS systému. Toto radikální a zdánlivě nákladné řešení může díky vyspělosti, kvality a rychlosti vývoje v dnešních PHP frameworcích představovat efektivní řešení a může být pro webdesignery dobrou investicí do budoucnosti.

Práce si dále klade za cíl stručně pojednat o obecných výhodách, které postavení CMS systému na některém z moderních PHP frameworků přináší, a navrhnout možnou architekturu takového řešení.

[1] PHP triky - Verze PHP v ČR – únor 2010: http://php.vrana.cz/verze-php-v-cr-unor-2010.php

Práce bude v příloze obsahovat kompletní ukázku implementace navrženého systému za použití frameworku Nette.

1. Populární webové CMS systémy

Bráno v měřítcích Internetu, webové CMS systémy mají za sebou již poměrně zasloužilou historii (první vznikaly zejména jako komerční produkty již v 90. letech) a jejich vývoj rozhodně neustává, naopak neustále vznikají systémy nové. Jejich význam je totiž značný – už jen to, že umožňují často kompletní správu webové prezentace běžným webmasterům se základními znalostmi, tzn. nejen kodérům a programátorům, například v komerční sféře šetří náklady, zvyšuje flexibilitu a napomáhá klientům udržovat si jistou nezávislost na webdesignové společnosti. A zajdeme-li ještě dále, v některých případech umožňují až tvorbu webové prezentace svépomocí.

Přestože ve svých počátcích byly webové CMS systémy především komerčními produkty, postupem času začalo vznikat nepřeberné množství systémů šířených pod některou ze svobodných licencí a některé z nich kolem sebe dokonce i dokázaly vytvořit rozsáhlou komunitu webmasterů a vývojářů, čímž si vybudovali značnou popularitu a dobré jméno. Avšak, jak už tomu bývá, mnoho možností znamená i mnoho nároků na správnou volbu.

Volba vhodného CMS systému jako základu pro zakázkovou tvorbu webových prezentací není vůbec jednoduchá. A to nejen z důvodu různorodosti potřeb klientů, ale také proto, protože svým způsobem žádný z dnešních populárních CMS systémů není dokonalý a bez nedostatků.

1.1. Prameny častých nedostatků

Některé problémy, se kterými se populární CMS systémy na platformě PHP potýkají, mají komplexnější pozadí a nejsou jednoduše důsledky špatných návrhů či koncepcí ze strany jejich tvůrců. Představme si alespoň některé z těchto faktorů.

1.1.1. PHP – jak z neobjektového objektový jazyk vzešel

PHP je svým způsobem poněkud neobvyklý jazyk. Vytvořil jej v roce 1994 dánský programátor Rasmus Lerdorf jako vcelku nenápadnou pomůcku pro potřeby své vlastní domovské stránky (odtud prvotní význam zkratky: *Personal Home Page*), čímž se mu podařilo nahradit několik skriptů napsaných v jazyce Perl, plnících vesměs nenáročné úlohy (jednoduché měření návštěvnosti a podobně)[1]. Avšak projekt si nenechal jen pro sebe a 8. června roku 1995 jej ve verzi označené jako PHP 2 představil veřejnosti. Za zmínku stojí, že již v této rané verzi jeho funkcionalita v základě připomínala verze dnešní.

Základ PHP 3 vznikl v roce 1997, kdy se do vývoje zapojili izraelští programátoři Zeev Suraski a Andi Gutmans, kteří pro účely PHP 3 mimo jiné kompletně přepsali parser. Ve verzi 3 se PHP také poprvé masově rozšířilo (uvádí se, že bylo „motorem“ pro statisíce internetových stránek). Radikální vývoj ale neustál a pro účely verze 4 Andi Gutmans a Zeev Suraski pro změnu kompletně přepsali jádro jazyka, nově poháněného Zend Enginem (název podle křestních jmen tvůrců Zeev a Andi). PHP 4 bylo vydáno 22. května 2000 a představovalo první verzi, ve které bylo možné využívat již některých základních nástrojů objektově orientovaného programování.

Zásadním přelomem bylo vydání verze 5 v roce 2004, od které je možné na PHP nahlížet jako na plnohodnotný objektově orientovaný jazyk, přičemž aktuální minor verze 5.3.2 (informace k 8. březnu 2010) zahrnuje již i dlouho postrádanou podporu jmenných prostorů, přináší closures a řeší množství problémů předchozích verzí (například problém s late static binding).

Co je však velmi důležité, PHP ve svých raných verzích nebylo objektově orientovaným jazykem a nabylo tohoto přívlastku až časem. A to na něm pochopitelně nemohlo nezanechat stopy, především při snaze o zachování zpětné kompatibility. Můžeme jmenovat hned několik souvisejících rysů, za které je PHP poměrně oprávněně kritizované:

[1] PHP: History of PHP: http://www.php.net/manual/en/history.php.php

- Většina základních funkcí jazyka není definována v třídách, nýbrž nachází se „kdesi“ v globálním prostoru a jejich používání v objektově psaných aplikacích vede ke směsi objektového a neobjektového kódu.[2]

- Výše zmíněné funkce nevyužívají elegantního mechanismu výjimek, nýbrž vzniklé problémy řeší návratovými hodnotami a generováním chybových událostí (které se v závislosti na nastavení interpretu mohou zobrazovat na výstupu a ukládat do logů). Tyto chyby lze sice sekundárně překládat na výjimky (například pomocí funkce `set_error_handler`), avšak toto nelze považovat za elegantní řešení a nevyrovná se nativnímu vyhazování specifických výjimek, na které jsou programátoři zvyklí z vyspělejších objektově orientovaných jazyků[3].

- Objektový model se vyvíjel velmi postupně a zejména ve verzi 4 působil oproti modelům obsaženým v jiných moderních jazycích (Java, C#) spíše jako experiment. Jako plnohodnotný se většinou vnímá až od verze 5.0, kdy ale například jménné prostory byly do jazyku zahrnuty až ve verzi 5.3 vydané 30. června 2009[4].

Dopady na CMS systémy

Při zkoumání dopadů vývoje PHP na CMS systémy si musíme uvědomit, že historie řady populárních PHP CMS systémů sahá do doby před PHP 5 (které je svým způsobem revoluční) a některé dokonce až deset let nazpět do období, kdy bylo sotva vpuštěno do světa PHP 4. Jelikož skoro každý takový systém alespoň nějakým způsobem dbá na zpětnou kompatibilitu, z principu nelze od těchto systémů očekávat, že jejich architektura bude využívat nejmodernějších vlastností jazyka PHP, především pak těch, které souvisí s OOP.

[2]PHP: Function reference: http://www.php.net/manual/en/funcref.php

[3]PHP: Error Handling: http://www.php.net/manual/en/book.errorfunc.php

[4]PHP: Classes and Objects: http://www.php.net/manual/en/language.oop5.php

1.1.2. Technické možnosti versus požadavky na CMS systémy: spojené nádoby

Pokrok webových technologií probíhá mílovými kroky a paralelně na mnoha frontách. Vyvíjí se jak značkovací jazyky (například nově připravované HTML 5[5]), tak i programovací jazyky na straně serveru i klienta, databázové systémy, prohlížeče a se vzrůstajícími možnostmi pochopitelně vzrůstají i nároky uživatelů: jak Internetu obecně, tak i administrátorů pracujících s CMS systémy. Jako příklad můžeme uvést nedávný prudký nárůst obliby JavaScriptových aplikací, známých také pod zkratkou RIA (Rich Internet Application), které prakticky boří typické paradigma webu jako souboru stránek vzájemně provázaných odkazy. S použitím technologií, jako například AJAXu (Asynchronous JavaScript and XML), dokáží v prostředí prohlížeče vytvořit robustní aplikaci schopnou uchovávat si svůj stav a přenášet mezi klientem a serverem pouze nezbytně nutná data.

Poznámka

Za zmínku stojí, že ačkoliv samotný vznik termínu AJAX a jeho významné rozšíření se datuje do roku 2005 (tehdy Jesse James Garrett publikoval článek *Ajax: A New Approach to Web Applications*[6]), schopnosti prohlížečů, kterých AJAX využívá, byly přítomny již v Internet Exploreru 5 při jeho uvedení na trh 18. března roku 1999[7].

Tato technologie velmi záhy dorazila i do oblasti CMS a vzniklo hned několik systémů, které prakticky neobsahují typickou „administrátorskou sekci“, nýbrž správa celé prezentace v nich probíhá prostřednictvím robustní JavaScriptové vrstvy, která se nachází přímo nad stránkami v takové podobě, jak je vidí návštěvník. Jako příklad takto koncipovaných systémů můžeme uvést například americký Weebly[8], či populární český Webnode[9].

[5]HTML5: http://dev.w3.org/html5/spec/Overview.html

[6]adaptive path » ajax: a new approach to web applications: http://www.adaptivepath.com/ideas/essays/archives/000385.php

[7]About Native XMLHTTP: http://msdn.microsoft.com/en-us/library/ms537505(VS.85).aspx

[8]Weebly - Create a free website and a free blog: http://www.weebly.com

[9]Webové stránky zdarma - snadná tvorba www stránek zadarmo: http://www.webnode.cz

Dopady na CMS systémy

Aktivní propagace nejrůznějších „web 2.0“ technologií má jeden zmínění hodný efekt. Zásadně ovlivňuje široké povědomí webmasterů o tom, co vše je možné realizovat a co vše mohou od webových aplikací očekávat. Uživatelská rozhraní, o kterých by uživatelé ještě před několika lety nevěřili, že jsou realizovatelná, dnes od webových aplikací běžně vyžadují. A tento trend se týká i CMS systémů.

V důsledku toho je udržování CMS systémů moderními náročnou a z principu nikdy nekončící prací. Jako příklad systému, který se snaží držet krok, můžeme uvést například Drupal, který v připravované verzi 7 podle všech informací přichází s úplně novým, moderním, uživatelským rozhraním v plné míře využívajícím možností skriptování na straně klienta[10]. To představuje oproti aktuální verzi 6 v zásadě radikální inovaci.

1.2. Charakteristika vybraných populárních systémů

1.2.1. Komentář k výběru

V následující kapitole se zaměříme na tři populární webové CMS systémy zvolené tak, aby každý z nich byl typickým zástupcem některé z následujících „pomyslných“ kategorií (pomyslných proto, protože webové CMS systémy se v současné době žádným standardním způsobem nekategorizují):

1. kategorie: robustní systémy s dlouhou tradicí a rozsáhlou komunitou

Názvy systémů z této kategorie jsou v obecném povědomí webdesignerů. U některých z nich se uvádí, že pohání až stovky tisíc webů a často jsou tím prvním, na co webmaster při průzkumu trhu s webovými CMS systémy narazí. Jedná se o systémy s historií, jejichž kvalita je bohatě prověřená a za dobu jejich existence kolem nich vznikla rozsáhlá komunita webmasterů a vývojářů neustále vyvíjejících nová rozšíření. Jejich dokumentace bývají na velmi dobré úrovni a množství existujících doplňků nepřeberné.

[10]Drupal 7 Alpha Preview: http://www.youtube.com/watch?v=DaofScfCF1w

Typickými zástupci jsou *Drupal*, *Joomla!* či *WordPress*, kdy my se budeme více zabývat Drupalem.

2. kategorie: minimalistické systémy založené na jednoduchosti a uživatelské přívětivosti

Tyto systémy se snaží konkurovat robustním systémům zejména svou jednoduchostí (vnitřní i vnější) a jakousi nenáročností. Jejich tvůrci jsou si dobře vědomi časté těžkopádnosti svých robustních konkurentů a kráčí cestou minimalismu. Takovéto systémy často ve svém jádru obsahují pouze velmi základní funkcionalitu, kterou je pochopitelně opět možné rozšiřovat doplňky – buďto již existujícími, anebo napsanými svépomocí. V jaké míře a s jakým pohodlím lze tyto systémy rozšiřovat dosti záleží na jejich architektuře a způsobu fungování obecně.

Podíváme se na zástupce této kategorie zvaného *CMS Made Simple*.

3. kategorie: systémy s důrazem na vývojářskou přívětivost a moderní architekturu

Do této kategorie, asi nejméně ostře ohraničené, můžeme zařadit PHP CMS systémy, které plně využívají objektového modelu a obecně nejmodernějších technik, jaké PHP ve svých aktuálních verzích nabízí. Silnou stránkou těchto systémů je, že jejich kvalitně navržené zástupce je pro programátory velmi pohodlné rozšiřovat.

Z této kategorii podrobíme analýze populární systém *TYPOlight*.

1.2.2. Metodika hodnocení systémů

Součástí každého hodnocení bude stručná charakteristika systému a náhled na něj z hlediska:

1. komunity, podpory a dostupnosti doplňků,
2. administrátorského,
3. vývojářského.

Přibližme si, co tato hlediska znamenají a proč se budeme zabývat právě jimi.

1.2.2.1. Komunita, podpora a dostupnost doplňků

Nehledě na konkrétní licenci, kolem většiny kvalitních a volně dostupných systémů vzniká s postupem času početná komunita uživatelů a vývojářů. Její existence je pro projekt samotný a jeho rozvoj často až klíčová: komunita systém neustále testuje (ne nadarmo se říká, že nejlepší testeři jsou skuteční uživatelé), usměrňuje, přináší podněty k jeho rozšiřování (či jej rovnou rozšiřuje formou volně dostupných doplňků), propaguje, poskytuje podporu a mnoho dalšího.

V neposlední řade komunita dává systému stabilitu a snižuje riziko, že by v případě svého odchodu hlavní vývojáři neměli nástupce a další vývoj by tak byl ohrožen.

U systémů s početnou komunitou můžeme očekávat kvalitní dokumentaci, velmi dobrou podporu a velké množství doplňků – je-li systém postaven modulárně. Naopak u systémů bez významné komunity můžeme očekávat řadu neznámých, zejména týkajících se podpory a vývoje v budoucnosti.

1.2.2.2. Administrátorské hledisko

Administrátorským hlediskem budeme rozumět *použitelnost systému* pro administrátora, který bude nasazený systém spravovat (tj. pro webmastera). To je velmi důležité hledisko, protože v krajním případě, kdy je systém pro administrátora velmi špatně použitelný (například pro zbytečnou komplexnost administrátorského rozhraní), v podstatě přestává plnit svůj základní účel.

Poznámka

V praxi jsem se osobně skutečně setkal s klienty, kteří poptávali kompletní vyhotovení webu jen z toho důvodu, že jejich předchozí CMS systém nebyli schopni pro jeho přílišnou složitost používat.

1.2.2.3. Vývojářské hledisko

Z pohledu vývojáře je důležité, jak je daný systém koncipován a jak pohodlné a efektivní je jeho nasazování a rozšiřování.

Zejména v případech, kdy je nezbytné programovat rozšíření na míru, jsou systémy využívající moderních prostředků jazyka PHP pro současné vývojáře podstatně přívětivější, než

systémy založené na starších postupech. To, jaký kód nutí systém vývojáře v takové situaci produkovat, je důležité.

1.2.3. Drupal

Drupal je jedním z předních zástupců robustních, volně dostupných CMS systémů. Množina typů webů, které lze na Drupalu rozumně provozovat, je nadstandardně široká – systém vyhoví požadavkům e-shopů, korporátních sítí, blogů, komunitních webů a řady dalších typů webových prezentací. Jeho autoři jej oprávněně nazývají „CM frameworkem“.

Historie Drupalu sahá až do roku 2000, kdy Dries Buytaert, student Antverpské univerzity, postrádal jednoduchý komunitní systém pro sdílení informací mezi přáteli v rámci kolejní sítě. Začal tedy požadovaný systém vyvíjet svépomocí a na doméně drop.org zprovoznil na něm založenou „sociální síť“. V lednu roku 2001 se rozhodl systém pohánějící drop.org (tenkrát ve verzi označované 1.0.0) vydat veřejně pod názvem Drupal[11].

V současnosti se Drupal vyvíjí ve dvou větvích: 5 a 6. Větev 5 má počátky v roce 2007 a aktuální verze v ní nese označení 5.22, větev 6 se oddělila v roce 2008 a aktuální verze v ní nese označení 6.16[12]. Aktivně rovněž pokračuje vývoj nové větve 7, která bude podle všech informací a dostupných alfa verzí značně revoluční (zejména co se uživatelského rozhraní týče), a souběžně s vydáním první stabilní verze v ní vývojáři plánují ukončit podporu větve 5.

1.2.3.1. Komunita, podpora a dostupnost doplňků

Komunita

Komunita kolem Drupalu je soustředěna na doménách drupal.org a groups.drupal.org a jejímu fungování prakticky není co vytknout. Jen na doméně drupal.org (tj. neuvažujeme nyní subdomény groups.drupal.org a další) se nachází více než 365 000 stránek sestávajících se z aktualit, požadavků na nové funkcionality, bug reportů, patchů, témat vzhledu, modulů, vláken fóra a

[11] History | drupal.org: http://drupal.org/about/history

[12] Informace aktuální k 5. dubnu 2010.

dokumentace[13]. Samotné oficiální fórum obsahuje přes 200 000 vláken, z nichž podstatná část se týká podpory a běžná prodleva odpovědi na položený dotaz je v řádech minut až jednotek hodin.

Dokumentace

Vlastní dokumentace je velmi dobře zpracovaná a tématicky v podstatě vyčerpávající. Je značně rozsáhlá a z toho přirozeně vyplývá snad jediná její slabina – pro nově příchozího může být z počátku obtížné dostat se právě k těm informacím, které hledá. Naprosto nepostradatelnou pomůckou se tak stává funkce vyhledávání.

Dostupnost doplňků

V současné době existuje přes 2 400 doplňků pro Drupal 5.x a přes 3 600 doplňků pro Drupal 6.x. V tomto množství lze nalézt doplněk prakticky pro každý standardní požadavek.

Velkým plusem je, že všechny jsou dostupné a kategorizované propracovaným způsobem přímo na doméně drupal.org. Propracovaným proto, protože nad každým doplňkem mají uživatelé možnost rozvést diskuzi, pomocí ViewVC lze on-line prohlížet CVS úložiště s množství verzí podle tagů či větví, jsou dostupné konkrétní dokumentace a v neposlední řadě jsou dostupné i velmi detailní statistiky počtů stažení.

To má jeden velmi kladný důsledek – lze tak relativně snadno odhadnout, které doplňky jsou kvalitní, stabilní, populární a je možné je bez větších obav použít i v kritických částech vyvíjené webové aplikace.

Jediným problémem, který je však společným jmenovatelem snad všech CMS systémů rozšiřovatelných doplňky, zůstává skutečnost, že velmi specifická zadání často není možné vyřešit ani nadstandardně konfigurovatelnými doplňky. To není problém v případě, že specifikace od zadavatelů jsou volné a vývojáři sami mohou navrhnout řešení ušité na míru dostupným doplňkům, avšak v případě, kdy zadavatel webu má o jeho fungování velmi přesnou představu,

[13]Community Initiatives | drupal.org: http://drupal.org/community-initiatives

často nezbude než množství součástí naprogramovat na míru – a zdaleka tomu nemusí být tak jen u rozsáhlých webů.

1.2.3.2. Systém z pohledu administrátora

Drupal má pověst komplikovaného systému a z hlediska použitelnosti představuje takřka učebnicový příklad systému, který navrhovali programátoři.

Uživatelská přívětivost a použitelnost jsou aspekty značně subjektivní a obtížně se objektivně hodnotí. Avšak možná právě pověst, kterou Drupal má, vedla tým vedený Beccou Scollan k vypracování studie přímo na téma použitelnosti Drupalu[14].

Výsledky této vynikající studie, ve které ale Drupal nedopadl vůbec dobře, pokorně přijala i drupalovská komunita a sám Dries Buytaert prohlásil, že si ji „vytiskl, přilepil na zeď a že nevydá Drupal 7 dříve, než bude vyřešeno alespoň 90% z identifikovaných problémů“[15]. Přílohou studie je i vynikající audiovizuální záznam, na kterém je přímo patrné, s čím měli respondenti problémy a jak značná byla místy míra jejich „frustrace“[16].

Webmasteři měli například následující problémy:

Zmatení plynoucí z neoddělené administrační sekce

Správa webu je v Drupalu řešena velmi specifickým způsobem. Administrátor není po přihlášení přesměrován do žádné „administrátorské sekce“, ale zůstává v prostředí vlastní prezentace a jsou mu pouze v hlavním menu navíc zpřístupněny položky související se správou obsahu. Sekce, ze kterých se obsah webu spravuje, není ani žádným způsobem graficky odlišená. Na základě reakcí respondentů můžeme prohlásit, že pro uživatele nepatří tato koncepce mezi zrovna nejpřehlednější a nejintuitivnější.

V základní distribuci chybí WYSIWYG editor

Všichni respondenti při práci s obsahem pátrali po vizuálním editoru, který se ale v základní distribuci Drupalu nenachází. Jeho doplnění je otázkou instalace a nastavení některého od-

[14]Drupal Usability Research Report: http://groups.drupal.org/files/DrupalUsabilityResearchReport.pdf

[15]Usability, usability, and usability | Dries Buytaert: http://buytaert.net/usability-usability-and-usability

[16]University of Baltimore Drupal Usability Study: http://video.google.com/videoplay?docid=-3942663057311698508#

povídajícího rozšíření. Toto lze v dnešní době chápat jako nedostatek, protože webmasteři jsou zkrátka na přítomnost WYSIWYG editorů u podobných systémů zvyklí. Z audiovizuálního záznamu je patrné, že je ani nenapadlo, že by v Drupalu mohl chybět a že by jej mohlo být zapotřebí speciálně instalovat. Namísto toho jej zdlouhavě hledali.

Terminologie

Pro respondenty bylo obtížné pochopit celou řadu pojmů (jako příklad můžeme uvést pojem „primary links“). Rovněž měli problém s jejich nekonzistencí: například úvodní stránka je někde nazývána „home page“, jinde „front page“ a podobně.

Vážných problémů, na které respondenti narazili, je ale ještě více.

Závažné je, že někteří respondenti nebyly schopni jisté úkoly vůbec splnit. Tím Drupal selhává v jedné ze základních myšlenek CMS systémů, a to ve snaze zpřístupnit tvorbu a následnou správu obsahu webové prezentace „laikům“.

1.2.3.3. Systém z pohledu vývojáře

Drupal je silně modulární systém. Konkrétní nasazení Drupalu si můžeme představit jako soubor Drupalovského jádra, rozšiřujících modulů, motivů vzhledu, nastavení a vlastních dat.

Samotné jádro Drupalu není tvořeno ničím jiným, než souborem modulů poskytujících základní funkcionalitu. Momentálně se jich ve standardní distribuci nachází něco málo přes 30, přičemž nejzákladnější, pro běh systému nezbytné, jsou následující:

Block

Tento modul obstarává logiku spojenou s vykreslováním a správou „bloků“. Ty představují v terminologii Drupalu jakési útržky obsahu (typickým příkladem bloku je například přihlašovací formulář pro registrované uživatele, přehled návštěvníků on-line a podobně), které se vykreslují na stránce ve zvolených, šablonou definovaných „regionech“ (například v záhlaví, zápatí a podobně). Množina bloků, se kterými lze v konkrétním nasazení Drupalu pracovat, je definována použitými moduly (například pomyslný modul pro správu anket může definovat blok „ankety za poslední týden“ a podobně).

Filter

Modul, který umožňuje nastavit filtry pro textová vstupní pole. S jeho pomocí lze například odstranit ze vstupu potenciálně nebezpečné (X)HTML/JavaScript zápisy.

Node

Jedním ze základních pojmů v terminologii Drupalu je „node“ (česky „uzel“), který představuje základní jednotku obsahu. Každý „node“ má své vlastní unikátní číslo (NID), verzi (VID), svá metadata, titulek a obsah. Můžeme se setkat i s konstatováním, že „node“ zastupuje de facto „Content“ ze zkratky „CMS“. Stejnojmenný modul obstarává logiku s uzly spojenou.

System

Modul, který poskytuje řadu z elementárních funkcionalit Drupalu, spravuje množství základních nastavení, umožňuje spouštění naplánovaných úloh a zajišťuje základní kešování vygenerovaných stránek.

User

Modul umožňuje registraci, přihlašování a odhlašování uživatelů, zastřešuje správu rolí a oprávnění a individuálních uživatelských nastavení.

Jak již bylo zmíněno, kromě těchto základních modulů je již ve standardní distribuci dostupné množství dalších, které do systému například přidávají podporu pro generování navigace (modul „Menu“), novinek (modul „Blog“) a obecně vzato rozšiřují systém o funkcionality, které současné webové aplikace běžně vyžadují.

Struktura modulu a jeho interakce s okolím

Modul má podobu složky obsahující všechny soubory, které k němu bezprostředně patří. Z nich jsou nezbytně nutné pouze následující dva: soubor *jmenomodulu.info*, který obsahuje veškerá metadata potřebná k tomu, aby Drupal modul „viděl“ a byl s ním schopen pracovat, a soubor *jmenomodulu.module*, který obsahuje vlastní PHP kód modulu. Další soubory, které se ve složce modulu běžně nacházejí, obsahují například knihovny funkcí (většinou soubory s příponou *inc*), šablony (běžně soubory s příponou *tpl.php*), manuály k použití a licenční ujednání.

Vskutku důmyslný je mechanismus, kterým konkrétní modul interaguje s ostatními (tedy i s moduly tvořícími samotné jádro Drupalu). Drupalistům je známý pod pojmem „hooks“ (tento terminus technicus můžeme přeložit poněkud kostrbatě jako „háky“) a představuje mocný nástroj k zachytávání událostí, které v celém procesu generování požadované stránky nastávají, a k reagování na ně. Drupal samotný totiž definuje velké množství událostí[17], na které se může modul „přichytit“ a svým kódem ovlivnit jejich průběh.

Jako příklad si můžeme představit modul, který má za úkol vložit na konec každého článku reklamní prvek. Modul sleduje v systému událost, při které dochází k získání obsahu článku z databáze, a bezprostředně po ní k získanému obsahu připojí požadovaný kód reklamy.

Možnosti této koncepce jsou opravdu velké – moduly běžně přidávají své vlastní položky do konkrétních menu (která se generují v události „hook_menu“), v potřebný moment definují dostupné bloky (událost „hook_block“), typy uzlů (událost „hook_node_info“) a dalo by se dlouze pokračovat.

Druhá strana mince

Přestože je Drupal koncepčně navržen velmi dobře, to, jak jsou v něm myšlenky naimplementovány, bývá velmi častým terčem kritiky. Přibližme si alespoň něco z toho, co na Drupalu řadě vývojářů právem nevyhovuje.

Styl programování v Drupalu silně zaostává za moderními technikami

Při programování modulů v Drupalu se mnohý současný PHP vývojář může cítit, jako kdyby se přenesl hned o několik let nazpět v čase. Jako ukázku typického Drupalovského kódu si můžeme rozebrat například kód relativně malého modulu *Insert Block*[18]. Toto rozšíření doplňuje do systému jednu velmi užitečnou funkcionalitu – umožňuje jednoduché vložení bloku definovaného některým modulem do kontextu prakticky libovolné textové oblasti na stránce. Mějme například následující situaci: na míru naprogramovaný modul „kontakt“ definuje blok s kontaktním formulářem. Za použití základních modulů dodávaných

[17]Hooks | Drupal API: http://api.drupal.org/api/group/hooks

[18]Insert Block | drupal.org: http://drupal.org/project/insert_block

s Drupalem by jej bylo možné umístit do některého bočního sloupce či do jiného vyhrazeného regionu v šabloně, avšak nikoliv již například doprostřed hlavního textového obsahu stránky.

Struktura kódu modulu v podobě hlaviček funkcí bez jejich těl vypadá následovně:

```
<?php

/**
 * Implementation of hook_filter_tips().
 */
function insert_block_filter_tips($delta, $format, $long = FALSE);

/**
 * Implementation of hook_help().
 */
function insert_block_help($section = 'admin/help#insert_block', $args ▶
= array());

/**
 * Implementation of hook_filter().
 */
function insert_block_filter($op, $delta = 0, $format = -1, $text = '');

function _insert_block_substitute_tags($text);

/**
 * Implementation of hook_theme().
 */
function insert_block_theme();

function theme_insert_block_block($block);
```

Přestože analyzovaný modul můžeme co se do rozsahu kódu týče označit za malý, svou strukturou se prakticky nijak neliší od kódu modulů podstatně rozsáhlejších. Obsahuje prakticky jen implementace potřebných „hooks“ a podpůrné funkce, které se v implementacích háků volají.

Co je podstatné, prakticky všechen kód je psán procedurálně[19], značně nemoderně a zejména vývojáři jdoucí s dobou mohou mít problémy takovýto styl programování přijmout za svůj. V Drupalu musí zapomenout na velký komfort OOP, musí zapomenout na principy tříd, dědičnosti, na práci s rozhraními, jmennými prostory a opět přijmout za své, že vše se děje ve funkcích nadefinovaných kdesi v globálním prostoru, jejichž názvy musí mít samozřejmě přesně definovanou podobu, a mohli bychom dlouze pokračovat.

V tomto směru zkrátka Drupal nezapře, že letos oslavil již desáté narozeniny a po dobu své zasloužilé existence z řady pochopitelných důvodů (například kvůli alespoň nějaké zpětné kompatibilitě) bohužel nemohl pružně reagovat na vývoj možností jazyka PHP.

Deployment aneb kterak se do databáze šablony a útržky aplikačního kódu ukládají

Autoři Drupalu se databázi přidělenou aplikace rozhodli využívat opravdu masivně. Drupal si do databáze ukládá vše možné a dokonce i řadu entit, které bychom v databázi pravděpodobně nehledali. Jedná se zejména o útržky (X)HTML kódu šablon či útržky PHP kódu, které systém následně vykonává voláním funkce `eval()` či vylepšené `drupal_eval()`.

To zásadně znesnadňuje deployment, efektivní používání verzovacích nástrojů a synchronizaci vývojářů ve vývojovém týmu. V databázi Drupalovského systému se nachází řada informací, které vývojáři z jiných platforem běžně verzují a uchovávají v úložišti. Kdykoliv je zapotřebí, může si kterýkoliv vývojář stáhnout *head* revizi a pracovat s aktuální verzí aplikace (samozřejmě bez aktuálního obsahu, ale to ve vývojářské praxi většinou ničemu nevadí). Takto jednoduše ale bohužel v Drupalu získat kopii aktuální vývojové verze nelze. Vývojáři v Drupalu musí totiž jako bonus řešit, kde jsou v databázi jaké fragmenty šablon, v horším případě i fragmenty PHP kódu.

[19] Až na pár výjimek, kdy se objekty v Drupalu sice používají, ale slouží spíše jen jako jakási rozšířená verze datového typu pole.

Prakticky ze stejného důvodu je obtížný i deployment. Je to fakt, který nepopírají ani zastánci Drupalu a hezky o této problematice pojednává například prezentace od Jeffa Eatona[20], Drupalovského vývojáře ze společnosti Lullabot, jejíž 67 slidů popisujících, jak správně provést deployment, začíná následujícím:



V současné době sice existuje rozšíření, které si bere za úkol deployment usnadnit[21], avšak v době psaní této práce je zatím stále ještě ve fázi *aktivního* vývoje a navíc ve výsledku usnadní deployment pouze základních modulů. Deployment libovolného modulu třetí strany bohužel z principu vyřešit nemůže.

Často tedy deployment dopadá tak, že vývojář si doplní nové moduly do své lokální vývojové verze, v administraci je nastaví, naplní daty, vše otestuje a následně stejnou posloupnost událostí provede na produkčním serveru. Toto je však postup v dnešní době již zdaleka překonaný.

1.2.3.4. Závěr

Drupal je bohatě prověřený systém, který vyniká svou flexibilitou, dostupností doplňků a rozsáhlou komunitou.

[20]Drupal Deployment: http://www.slideshare.net/eaton/drupal-deployment-presentation

[21]Deployment | drupal.org: http://drupal.org/project/deploy

Na druhou stranu styl, jakým je systém naimplementován, odpovídá stylu, jakým se programovaly PHP aplikace před několika lety nazpět a zejména pro moderní PHP vývojáře může být problém tento styl přijmout za svůj.

Nakonec ale možná ještě závažnějším nedostatkem je nepříliš dobrá použitelnost a zbytečná komplikovanost pro webmastery. Při odkazovaných testech použitelnosti došlo k tomu, že někteří respondenti jisté úkoly nebyli schopni vůbec splnit, což je krajní situace, která představuje selhání CMS systému v jeho základních myšlenkách.

1.2.4. CMS Made Simple

CMS Made Simple je příkladem systému, který si neklade za cíl pokrytí potřeb rozsáhlých korporátních webů, ale o to více klade důraz na uživatelskou přívětivost a praktičnost při použití na webech menšího rozsahu.

Systém začal vyvíjet Ted Kulp pro své vlastní potřeby poté, co nenašel žádný dostupný CMS systém, který by splňoval jeho očekávání (testování existujících pro něj bylo jeho vlastními slovy „frustrující“[22]).

První veřejná verze (0.1) byla vydána v červenci roku 2004 a o úspěchu systému vypovídá to, že jeho počet stažení (všech verzí dohromady) v říjnu 2009 překročil hranici 750 000[23].

1.2.4.1. Komunita, podpora a dostupnost doplňků

Komunita

Komunita je soustředěna na doméně cmsmadesimple.org a hlavní fórum se nachází na její subdoméně forum.cmsmadesimple.org. V současné době čítá bezmála 15 000 registrovaných uživatelů a 40 000 vláken. Kladně můžeme hodnotit přítomnost sekce „International Discussi-

[22]CMS Made Simple - About Us: http://www.cmsmadesimple.org/about-link/about-us/

[23]CMS Made Simple - Blog: http://blog.cmsmadesimple.org/2009/10/22/750-000-downloads/

ons“[24], která poskytuje dále strukturovaný prostor pro diskuzi v dalších 14 jazycích, včetně češtiny.

Česká komunita spravuje doménu cmsmadesimple.cz, která je s její mezinárodní a rozsáhlejší „sestrou“ velmi úzce provázaná. Vlastní česká podkategorie fóra obsahuje přes 170 vláken. Běžná prodleva odpovědi na položený dotaz je v řádech jednotek hodin, což lze hodnotit jako plně vyhovující. Jen v české podkategorii lze nalézt dotazy i velmi starého data, které jsou stále bez odpovědi.

Dokumentace

Dokumentace k systému je řešena formou otevřené wiki[25] a má velmi dobrou úroveň. Pokrývá všechna potřebná témata a má promyšlenou koncepci: například hned v úvodu je členěna podle typu uživatele na informace pro správce systému, redaktory, designery a podobně.

Dostupnost doplňků

Na doméně dev.cmsmadesimple.org existuje bezmála 700 projektů a toto číslo se prakticky rovná součtu doplňků, které nějakým způsobem rozšiřujících funkčnost jádra CMS Made Simple, a dostupných překladů (existuje i český). Spolu s bezmála 9000 registrovanými uživateli podprojektu „CMS Made Simple Forge“ to svědčí o nemalé popularitě systému i mezi vývojáři. Přestože velké systémy typu Drupal v obou těchto metrikách vedou, i u CMS Made Simple lze za použití dostupných doplňků nějakým způsobem pokrýt prakticky všechny požadavky běžné webové prezentace.

Existence množství doplňků se štítkem například „e–commerce“[26] dokládají flexibilitu systému a to, že je v něm nějakým způsobem realizovatelný například i e-shop.

Zde můžeme mít pouze výtku k organizaci doplňků: 695 dostupných doplňků je označeno změtí 817 štítků (jeden projekt může být označen více štítky), kdy řada štítků zaštiťuje pouze

[24]CMS Made simple Forum: http://forum.cmsmadesimple.org/index.php#3

[25]CMS Made Simple Documentation: http://wiki.cmsmadesimple.org/index.php/Main_Page

[26]CMS Made Simple - Forge: http://dev.cmsmadesimple.org/project/list_tagged/e-commerce

jeden projekt (pro který jej autor projektu pravděpodobně založil) a zároveň řadu štítků můžeme označit za duplicitní (například e-comm, e-commerce, e-commerse a eCommerce). Je pravděpodobné, že označení projektů není nijak moderované a již se lehce vymklo kontrole.

1.2.4.2. Systém z pohledu administrátora

Dobrá použitelnost, uživatelská přívětivost a jistá minimalističnost jsou bezesporu klíčovými vlastnostmi systému (svědčí o tom i jeho samotný název) a pravděpodobně právě díky nim je dnes tak populární.

V základní instalaci nabízí administrační rozhraní jen několik dobře srozumitelných a intuitivních sekcí: správu obsahů, rozložení stránek, správu uživatelů a uživatelských skupin, extenzí, nastavení webu a nastavení administrátorského účtu. Úvodní stránka administrace je realizovaná formou rozcestníku, ve kterém nechybí žádná z důležitých sekcí a podsekcí, a bylo myšleno i na takové detaily, jako například na možnost nastavit si do bočního sloupce libovolný počet odkazů na nejrůznější akce. To vše činí administraci dobře použitelnou, pochopitelnou a rychlou.

V administraci je vhodně využíván JavaScript (například změnu pořadí stránek lze realizovat pohodlným tažením myší), při editaci stránky se v kontextové navigaci setkáme s využitím přepínatelných záložek (které jsou z hlediska použitelnosti obecně považovány za dobré), v základní instalaci je přítomen populární WYSIWYG editor TinyMCE a administrátor má možnost ještě před jejich potvrzením vidět náhled provedených změn.

Cíl umožnit rychlé a snadné vytváření webových prezentací různých typů tak CMS Made Simple z pohledu administrátora úspěšně naplňuje.

1.2.4.3. Systém z pohledu vývojáře

Z hlediska architektury je CMS Made Simple hezkým příkladem směsi objektového a procedurálního kódu, což je něco, na co lze u projektů na platformě PHP velmi často narazit.

Celý životní cyklus aplikace se ukrývá ve skriptu index.php, který analyzuje požadovanou URL, načítá externí knihovny objektově i neobjektově psaného kódu, deleguje sestavování výstupního kódu a odesílá jej klientovi. Systém využívá šablonovacího systému Smarty, který je velmi populárním a rozšířeným šablonovacím systémem pro platformu PHP. Šablonová vrstva

je tedy kvalitně oddělená. Je to ale prakticky jediné rozdělení do vrstev v rámci aplikace, například vrstva, která by měla na starost business logiku, oddělená není (v důsledku čehož jsou například SQL dotazy hojně rozprostřeny napříč všemi zdrojovými kódy).

Systém je modulární. Každý modul je zastoupen složkou se zdrojovými soubory a šablonami jasně dané struktury a hierarchie. Každý modul obsahuje definici třídy dědící z `CMSModule` a mimo jiné deleguje načítání a vykonávání akcí uložených v souborech `action.*.php`. Ty obsahují jakési útržky kódu spouštěné v kontextu potomka `CMSModule`.

Celkově se jedná o jednoduchou koncepci. Na druhou stranu podobnou směs objektového a neobjektového kódu nelze nazvat za elegantní a lze se v ní setkat s konstrukcemi, které příliš neoslní. Jedná se například o enormně hojné využívání klauzule global, která je v čisté základní instalaci použita bezmála 550 krát v celkem 165 souborech. Vesměs jde o předávání instance *$gCms*, což by šlo bezpochyby zařídit lépe například použitím návrhového vzoru *dependency injection*[27]. Dále bychom mohli koncepci vytknout například vkládání útržků kódu do kontextu tříd. V samotných útržcích se poté totiž hojně pracuje s proměnnými a instancemi, které v něm nejsou inicializované a pro programátora může být problematické dohledat, kam se daný útržek vkládá a co je obsahem používaných proměnných. Mnohem elegantnějším způsobem by byl například použití dědičnosti a při vykonávání kódu modulu ověřovat existenci volitelně definovatelných metod a eventuálně je v pravý čas vykonávat.

1.2.4.4. Závěr

CMS Made Simple si klade za cíl být uživatelsky přívětivým, ale přesto robustním CMS systémem s dobrou rozšiřitelností. To se mu daří, a proto si dokázalo vybudovat nemalou popularitu. Z věcí, které mu lze vytknout, je architektura. Vzhledem k tomu, že systém vyžaduje PHP 5, mohl by jeho možností více využívat a být například napsán čistě objektově a využívat osvědčených návrhových vzorů.

[27] Inversion of Control Containers and the Dependency Injection pattern: http://martinfowler.com/articles/injection.html#InversionOfControl

1.2.5. TYPOlight

TYPOlight (který byl 15. května přejmenován na Contao[28]) je systém, který začal vyvíjet v roce 2004 německý vývojář Leo Feyer. Rozsahem a možnostmi jej lze bezesporu zařadit mezi robustní CMS systémy typu Drupal nebo Joomla! a slovy autorů se snaží být ideálním systémem pro ty, kteří vyžadují profesionální prezentaci, která bude zároveň i snadno spravovatelná. Přidáme-li si k tomu skutečnost, že celý systém je psán s plným využitím OOP a podle návrhového vzoru MVC, jedná se bezesporu o zajímavý systém.

1.2.5.1. Komunita, podpora a dostupnost doplňků

Komunita kolem systému je lehce roztříštěna do dvou fór: jedno se nachází přímo na doméně contao.org[29] a druhé na nezávislé doméně contao-community.org[30]. První zmíněné fórum je rozsáhlejší, zejména pak jeho německá verze. Z té je dobře patrné, že systém má německý původ a těší se velké popularitě právě ve své rodné zemi. Celkově vzato jsou obě fóra hrubě členěná a hledání informací v nich není nejpohodlnější. Další nevýhodou je také fakt, že už členění fór je z podstatné části duplicitní (obě fóra obsahují například sekci „Developer's corner“ a podobně).

Dokumentace systému je pro koncového uživatele obsáhlá, avšak pro vývojáře rozšíření je velmi strohá. Má charakter referenční příručky a neobsahuje žádný tutoriál, který by vývojáře uvedl do architektury systému a do způsobu jeho rozšiřování, a ani žádné ukázky kódu. Proto snad nejužitečnějším zdrojem informací zůstávají již hotová rozšíření, ze kterých je různě komplexním způsobem patrné, jak lze TYPOlight rozšiřovat. V kombinaci s mírně nepřehledným fórem se jedná vskutku o slabou stránku systému.

Dostupnost doplňků je z rozebíraných systémů nejslabší, avšak stále uspokojivá. Na oficiálním webu TYPOlight je v době psaní této práce více než 450 doplňků. Ty jsou organizovány podle štítků a dobrým pomocníkem je fulltextové vyhledávání, které snadno umožňuje najít optimální doplněk.

[28]TYPOlight becomes Contao - Contao Open Source CMS (fka TYPOlight): http://www.contao.org/news/items/typolight-becomes-contao.html

[29]Contao forum - Contao Open Source CMS (fka TYPOlight): http://www.contao.org/forum.html

[30]The Contao Community • Index page: http://www.contao-community.org [http://www.contao-community.org/]

1.2.5.2. Systém z pohledu administrátora

TYPOlight sám sebe prezentuje jako uživatelsky přívětivý a snadno spravovatelný systém. Realizací se nachází na pomyslné ose použitelnosti někde mezi dříve recenzovanými systémy Drupalem a CMS Made Simple. Administrační sekce je podstatně komplexnější než u CMS Made Simple, avšak díky přehledné koncepci není tak náročná na zorientování, jako například administrace v Drupalu.

Přesto je rozhraní již natolik obsáhlé, že prakticky není možné, aby všechny položky byly dokonale samovysvětlující a nezkušený administrátor se tak často může setkat s potřebou experimentovat. Z tohoto důvodu bychom při nasazování systému s předvyplněným obsahem (častý postup v komerční sféře) zároveň měli buďto poskytnout klientovi školení, anebo mu alespoň připravit oddělenou kopii pro experimentování a seznámení se se systémem.

1.2.5.3. Systém z pohledu vývojáře

TYPOlight je napsán čistě objektově a plně využívá možností jazyka PHP 5 (ke svému běhu potřebuje minimálně PHP 5.2.x). Jeho architektura vychází z prezentačního návrhového vzoru MVC. Tím se TYPOlight odlišuje od dříve recenzovaných systémů.

Dalo by se tedy očekávat, že práce s ním a jeho rozšiřování bude pro nově příchozího díky standardní architektuře a čistě objektovému kódu relativně snadná. Bohužel velmi to znesnadňuje již zmíněná velmi strohá dokumentace. Chybí tutoriály, návody, „best practices“. Vývojáři nezbude, než se učit z kódu již hotových rozšíření a podrobně studovat API referenci systému.

1.2.5.4. Závěr

TYPOlight se snaží být robustním a zároveň uživatelsky přívětivým systémem. Díky robustnosti jádra má velký potenciál a aktivně těží i z kvalitního objektového návrhu a návrhového vzoru MVC. Nejslabší stránkou je v současné době dokumentace, a to konkrétně ta část, která se týká rozšiřování systému a psaní doplňků podle specifických zadání. Dobře nepůsobí ani rozdělení komunity do dvou podobně rozsáhlých diskuzních fór.

2. Vlastní CMS systém jako řešení

Přestože v současné době v oblasti CMS systémů existuje nepřeberné množství volně dostupných řešení[1], není neobvyklé, že se webdesignér rozhodne nevyužívat žádného z nich a upřednostní své vlastní řešení. Motivací ke zdánlivému „vynalézání kola“ mu mohou být nejen nedostatky, kterými volně dostupné systémy trpí a kterými jsme se podrobněji zabývali v první části práce, ale například i následující skutečnosti:

Zejména v komerční sféře může proprietární software působit lépe

Zejména v komerční sféře často působí řešení na míru dojmem větší kvality, zvláště pak řešení na míru od některé z renomovaných softwarových společností. Toto je velmi komplikované téma, protože na druhou stranu je mnohdy právě prověřenost a otevřenost volně dostupných řešení brána jako zásadní výhoda oproti řešením na míru. Celá věc závisí na mnoha faktorech, kterými jsou například zkušenosti, schopnosti a renomé dodavatele, finanční možnosti zadavatele, otázka podpory a další.

Nicméně faktem zůstává, že i některé volně dostupné systémy je pro jejich konkrétní licenci skutečně obtížné v komerční sféře použít.

Detailní znalost kódu má cenu zlata

Každý vývojář (či skupina vývojářů například v nějaké firmě) pochopitelně nejlépe zná kód, jehož je autorem. Toto autorství přináší jeden podstatný benefit – autor kódu jej dokáže mnohem snadněji než kdokoliv jiný rozšiřovat, optimalizovat a opravovat (zejména lokalizace a oprava logických chyb je pro autora kódu většinou snazší než pro kohokoliv jiného).

[1] Na portálu http://php.opensourcecms.com je v době psaní této práce v evidenci více než 280 systémů.

Pro účely druhé části práce přijměme roli webdesignéra, který se pro nedostatky volně dostupných systémů rozhodl pro vývoj vlastního řešení a zvolil si pro jeho realizaci framework Nette a databázový systém MySQL.

2.1. Výhody systémů postavených na robustních frameworcích

Pozastavme se nad tím, proč bude náš systém založen na PHP frameworku a nebude, jak se tak říká, napsán „na zelené louce“. PHP je velmi obsáhlým jazykem, ve kterém existují funkce výstižně řečeno snad na všechno. S těmito možnostmi je ale zapotřebí umět dobře pracovat a to je zrovna něco, v čem klade čisté PHP na programátory vcelku vysoké nároky. PHP samo o sobě totiž nepředkládá programátorům žádná vodítka, jak optimálně řešit různá zadání a jak navrhnout kvalitní architekturu aplikace.

Mimo jiné i z tohoto důvodu vzniklo za posledních několik let množství PHP frameworků[2]. Z výhod, které PHP frameworky přinášejí, můžeme jmenovat především následující:

Usnadňují a zefektivňují práce

Cílem každého frameworku je odlehčit programátorovi od rutinních aspektů programování a v důsledku mu umožnit věnovat více času hlavním myšlenkám aplikace. Při používání frameworku se významně zkracuje čas, který uplyne od vzniku myšlenky až po její realizaci.

Vedou k bezpečnému a kvalitnímu kódu

Je známo, že při programování v čistém PHP je nutné například neustále řešit zabezpečení aplikace – jedná se o problematiku escapování generovaného kódu[3], zabezpečení autentizace uživatelů, obrany proti CSRF[4] a mnoho dalšího. Většina kvalitních frameworků toto řeší za programátora automaticky a opět tak výrazně šetří jeho čas a produkční kapacitu. Z pohledu klienta je velmi přínosné to, že u aplikace postavené na frameworku je všeobecně mnohem větší pravděpodobnost, že bude kvalitní a dobře zabezpečená.

[2]PHP Frameworks: http://www.phpframeworks.com

[3]Escapování - definitivní příručka » phpFashion: http://phpfashion.com/escapovani-definitivni-prirucka

[4]Co je Cross-Site Request Forgery a jak se mu bránit - Zdroják: http://zdrojak.root.cz/clanky/co-je-cross-site-request-forgery-a-jak-se-branit/

PHP frameworky v tomto směru demonstrují efektivní dělbu práce. Vývojáři frameworku se soustředí na kvalitní řešení těchto nízko-úrovňových záležitostí, zatímco vývojář používající framework se může soustředit na vysoko-úrovňové záležitosti, jakými jsou zejména hlavní myšlenky vyvíjené aplikace.

Vedou ke kvalitní architektuře

Dnešní PHP frameworky zpravidla nabízejí programátorům nějaký koncept i co se celkové architektury aplikace týče. To je velmi záslužné, protože, jak již bylo zmíněno, čisté PHP žádný takovýto koncept nenabízí a navrhnout aplikaci tak, aby se její architektura nestala při jejím dalším růstu a úpravách v budoucnosti omezující, vyžaduje nemalé programátorské zkušenosti.

Řada dnešních PHP frameworků využívá prezentačního návrhového vzoru MVC (který popsal již v roce 1979 Trygve Reenskaug[5]), který aplikaci rozděluje do třech vrstev a programátora vede k dodržování určitých pravidel, které souhrnně zajišťují dobrou spravovatelnost aplikace, zvyšují znovupoužitelnost kódu, vedou k dobré testovatelnosti a mohli bychom dlouze pokračovat.

Navíc i k zadáním, ke kterým nedefinují frameworky přímo pravidla, často přicházejí alespoň se souhrnem jistých doporučení (známými též mezi programátory jako „best practices“).

Zlepšují přenositelnost aplikací mezi vývojáři i celými vývojářskými týmy

Zejména u rozsáhlých aplikací napsaných v čistém PHP bez použití některého ze známých frameworků může být pro nově příchozího programátora zorientování se v projektu velmi náročné. U projektu postaveného na některém z populárních frameworků je pro nově příchozího programátora s jeho znalostí zorientování se v aplikaci o mnoho snazší. Toto je pro vývojářské týmy velmi podstatné, protože u větších a dlouhodobějších projektů je určitá fluktuace vývojářů naprosto běžná.

[5]Trygve/MVC: http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html

Argumentů proti používání frameworků není příliš, a proto se dnes těší velké popularitě. Kromě argumentu o nutnosti „učit se něco navíc“ se můžeme setkat ještě s tvrzením, že frameworky zvyšují náročnost aplikace na systémové prostředky, avšak to je jednoznačná pravda pouze u nejméně rozsáhlých aplikací (u kterých může být vhodnost nasazení frameworku skutečně na pováženou). Faktem zůstává, že vývojáři kvalitních frameworků zpravidla dbají i na otázku výkonu.

2.2. Nástin implementace CMS systému v Nette Frameworku

Nette Framework[6] je PHP framework vyvíjený Nette Foundation[7]. Počátky jeho vývoje sahají do roku 2004, načež první veřejná verze byla vydána v červenci roku 2008. To, že byl framework relativně dlouho vyvíjen jako „closed-source“, prezentuje jeho hlavní autor jako velmi šťastné rozhodnutí, protože tak nebyl zatížen potřebou zachovávat zpětnou kompatibilitu.

Framework Nette patří mezi nejpopulárnější frameworky české PHP scény. V únoru roku 2010, tj. 19 měsíců po jeho oficiálním vydání, poháněl weby již na 592 cz doménách[8], což znamená, že se rozšiřoval úctyhodným tempem jedné cz domény denně. V současné době existuje kolem frameworku početná komunita vývojářů a oficiální diskuzní fórum[9] má více než 2000 členů.

Bylo by velmi náročné až nemožné jednoznačně posoudit, zda je Nette Framework nejvhodnějším existujícím frameworkem pro implementaci vlastního CMS systému a rozhodně není to cílem práce. Uveďme jen, že jsme jej vybrali především pro jeho popularitu v ČR a také pro jeho výkonnost, která byla nezávisle zhodnocena serverem Root.cz[10].

[6]Nette Framework | Nette Framework: http://nette.org

[7]About the Foundation | Nette Foundation: http://nettefoundation.com

[8]PHP triky - Verze PHP v ČR – únor 2010: http://php.vrana.cz/verze-php-v-cr-unor-2010.php

[9]Nette Framework forum: http://forum.nette.org/cs/

[10]Velký test PHP frameworků: Zend, Nette, PHP a RoR - Root.cz: http://www.root.cz/clanky/velky-test-php-frameworku-zend-nette-php-a-ror/

2.2.1. Navrhujeme systém

2.2.1.1. Požadavky na systém

Před samotným návrhem implementace si musíme ujasnit, co od implementovaného systému budeme očekávat.

1. Nenáročnost na implementaci

   Od vyvíjeného systému budeme očekávat, že jej bude možné vyvinout a používat za rozumnou dobu. Pokud by náklady na jeho vývoj byly neúměrné, nemohli bychom jej brát jako vhodnou alternativu k volně dostupným CMS systémům a jeho vývoj by nepředstavoval řešení našeho problému.

   Z kontextu tohoto požadavku si od použití frameworku budeme slibovat kratší dobu vývoje oproti řešení „na zelené louce“. Architekturu systému navrhneme tak, abychom co nejefektivněji využívali možností frameworku a aby byly respektovány frameworkem dané „best practices“.

2. Široký záběr

   Od systému budeme očekávat, že v něm bude možné snadno realizovat širokou škálu klientských požadavků, zkrátka že se nestane jednoúčelným.

   I v tomto požadavku nám bude framework dobrým pomocníkem. Klíčové bude systém navrhnout dostatečně otevřený tak, aby do něj bylo možné doprogramovat prakticky libovolnou funkcionalitu na míru. U té lze díky použití frameworku očekávat kratší dobu vývoje, než bez jeho použití.

3. Uživatelská přívětivost

   Budeme se snažit, aby bylo uživatelské rozhraní administrační části systému intuitivní a dobře použitelné. Budeme dbát na výstižné popisky a budeme se vyhýbat zavádění termínů, které by pro běžné uživatele nemusely být srozumitelné (jako například „primary links“ v Drupalu či „slug“ ve WordPressu). Potenciálně obsáhlé formuláře, jako například formulář pro založení nové stránky, zrealizujeme formou průvodce o několika krocích.

4. Výkon

Jádro systému budeme optimalizovat i z výkonnostního hlediska. Jako příklad uveďme potřebu pracovat se stromovými strukturami uloženými v relační databázi (například se stromem stránek). Existuje několik přístupů, jak s takto uloženými daty pracovat, a my budeme volit takový, který nepovede ke zbytečnému využívání rekurzivních, výkonnostně náročných algoritmů. Dále budeme dbát na to, aby náročnější výpočty byly kešovány.

Ani v problematice kešování nás Nette Framework neponechá našemu vlastnímu osudu a nabídne nám svou vestavěnou podporu práce s dočasnou pamětí, která umožňuje například i pokročilé hromadné zneplatňování více položek v cache na základě tagů.

5. Multijazyčnost

Budeme chtít podporu pro vícejazyčnost, která je dnes pro spoustu prezentací nutností.

Zde využijeme nejen vestavěné podpory frameworku, ale i volně dostupné nadstavby GettextTranslator[11], která je implementací UNIXového gettextu.

6. SEO

Od systému budeme očekávat, že bude „SEO ready“, a to možností ovlivňovat z administrace on-page faktory. U každé stránky budeme požadovat, aby měla nastavitelné hlavičky *title*, *description* a *keywords* a zároveň aby bylo možné ovlivnit URL stránky.

2.2.1.2. Vlastní koncepce systému

Stránky

Základní entitou systému bude stránka určitého typu. Typ stránky si bude vybírat správce při jejím zakládání a bude jím definováno zejména rozložení hlavních obsahů a možnost sdílení vybraných obsahů (viz dále). Určité typy stránek budou moci být zastoupeny ve struktuře webu pouze jednou (například stránka typu „úvodní stránka“).

[11] GettextTranslator | Nette Framework: http://addons.nette.org/cs/gettext-translator

Hlavní obsahy

Každá stránka bude moci obsahovat typem stránky definovaný počet hlavních obsahů. Jedná se o obsahy, které budou pro danou stránku jedinečné a které mají být nositeli většiny ústředních informací. Rozložení hlavních obsahů v rámci stránky bude předem dané šablonou.

Sdílené obsahy

V systému bude možné předdefinovat tzv. sdílené obsahy, což budou jakési útržky obsahu, které budou moci být sdílené napříč více stránkami. Jedná se například o obsahy různých bočních sloupců. To, jaké sdílené obsahy bude moci daná stránka sdílet, bude řízeno přes typ stránky.

Globální obsahy

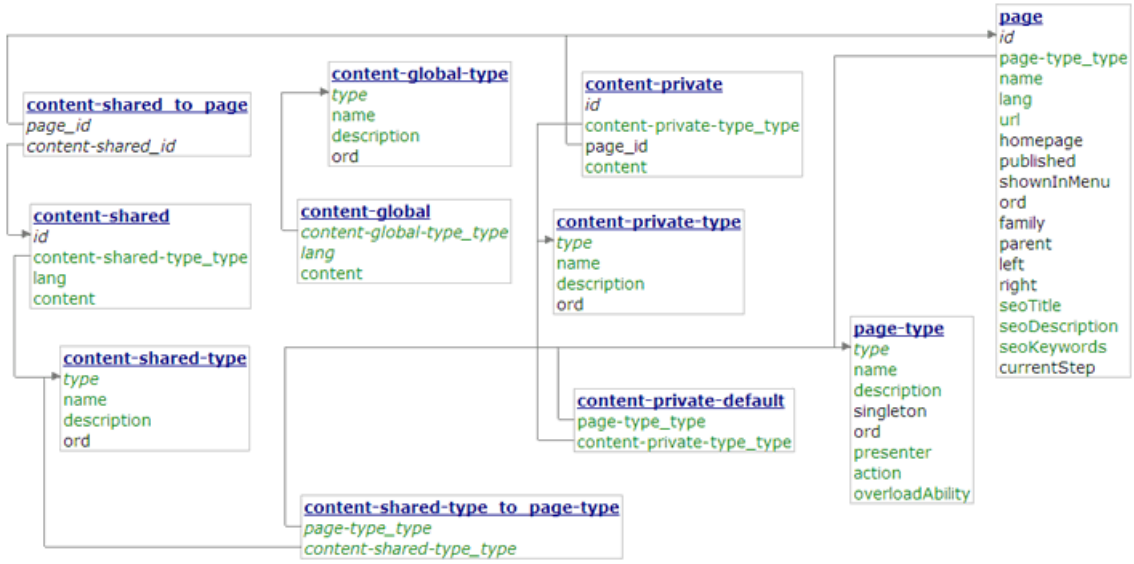
Globálními obsahy budeme rozumět obsahy, které budou společné pro všechny stránky z určité jazykové verze webu. Jako typický příklad uveďme zápatí stránek, slogan v záhlaví webu a podobně.

Takto koncipované obsahy by měly ze své podstaty pokrýt většinu v úvahu připadajících zadání. Speciální sekce, jako například novinky, budou řešeny následovně: bude pro ně existovat speciální typ stránky, například „přehled novinek“, ve kterém bude automaticky generován například přehled novinek s možností stránkování a zobrazení jejich detailů. Novinky samotné budou spravované přes speciální sekci v administraci.

Tímto způsobem lze do systému zakomponovat řadu dalších běžných funkcionalit, jakými mohou být různé formuláře (kontaktní, registrační), fotogalerie a další.

2.2.2. Schéma datového úložiště

Uvedená koncepce vede k následující struktuře MySQL databáze:



Obrázek 2.1. Schéma datového úložiště

Za komentář stojí tabulka *content-shared-type_to_page-type* definující, které stránky *mohou sdílet* které sdílené obsahy, a tabulka *content-shared_to_page* řídící, které stránky které sdílené obsahy *aktuálně sdílí*.

Tabulka *page* obsahuje na první pohled nepříliš zřejmé položky *family*, *left* a *right*. Jedná se o položky potřebné pro uložení stromové struktury a práce s ní pomocí tzv. „traverzování kolem stromu“[12].

Nepříliš samovysvětlující je položka *page.currentStep*. Ta slouží k uchování informace, ve kterém kroku zakládání se daná stránka nachází. Jak jsme navrhli již dříve v této kapitole, vytváření stránky bude z důvodu lepší použitelnosti rozděleno do několika dílčích kroků a právě tato položka bude sloužit k uchování informace, ve které fázi se proces tvorby nachází.

Položky *page-type.presenter*, *page-type.action* a *page-type.overloadAbility* souvisejí s fungováním frameworku a budou popsány později v této kapitole.

[12]MySQL :: Managing Hierarchical Data in MySQL: http://dev.mysql.com/tech-resources/articles/hierarchical-data.html

2.2.3. Využití Nette prezenterů

Nette využívá prezentačního návrhového vzoru MVP, ve kterém je prezentační vrstva tvořena hierarchickou strukturou tříd odvozených od základní třídy `Presenter`. Ta v jistém smyslu v Nette představuje instanci webové stránky (v historických verzích frameworku se tato třída dokonce nazývala přímo `Page`). My se budeme snažit této koncepci co nejvíce podřídit a nebudeme zakládat žádné nové třídy pro CMS stránky. Jednoduše každý typ stránky bude mít určeno, který potomek třídy `Presenter` jej reprezentuje.

V Nette může mít každý `Presenter` několik akcí, tzv. `actions`. Pro pochopení této skutečnosti mějme například dvě prakticky shodné stránky lišící se pouze v rozložení hlavních obsahů (řekněme jeden versus dva obsahové sloupce). Bylo by zbytečné mít pro každou z těch stránek vlastní prezenter. Vhodnější je reprezentovat ji jedním prezenterem a nadefinovat mu dvě akce. Tohoto budeme využívat. To je zároveň odpovědí na otázku, k čemu slouží v databázi položky *page-type.presenter* a *page-type.action*: skrze ně je pro každý typ stránky nadefinována *výchozí* kombinace prezenteru a akce. Pro uvedený příklad s dvěma rozloženími obsahů bychom mohli mít prezenter *Default* s akcemi *oneColumn* a *twoColumns*.

Řekli jsme si, že pro každý typ stránky nadefinujeme *výchozí* kombinaci. Další klíčovou vlastností našeho systému bude cosi, co bychom mohli nazvat „přetížitelností akce“. Mějme typ stránky „přehled aktualit“. Od tohoto přehledu budeme očekávat nejen možnost procházet nadpisy a perexy aktualit, ale i možnost zobrazení detailu vybrané položky. Z hlediska prezenteru budeme mít dvě akce: jednu, nazvanou řekněme *default*, která bude použita pro vygenerování přehledu aktualit, a druhou, řekněme *detail*, která bude použita pro zobrazení detailu aktuality.

Udělejme si nyní zdánlivou odbočku. Všechny uvažované stránky, tj. přehled aktualit i všechny jednotlivé detaily, přirozeně potřebují svou vlastní URL adresu. Jako vhodné řešení se nabízí použít URL `novinky` pro přehled novinek a URL `novinky/detail/nase-prvni-novinka` pro detail novinky „Naše první novinka“ a analogicky pro všechny další. Životní cyklus Nette aplikace začíná tím, že součást frameworku zvaná router převádí URL adresu na vnitřní požadavek (zapouzdřený do instance třídy `Nette\Application\PresenterRequest`) obsahující název prezenteru a další parametry včetně požadované akce.

Jak si detailně přiblížíme později, budeme používat vlastní router, který bude schopen hledat stránky uložené v databázi (takovýmto routerem Nette v základní distribuci nedisponuje). V našem příkladu bude router fungovat následovně: vyžádá-li si klient (klientem je zde myšlen návštěvníkův prohlížeč) adresu `novinky`, router nalezne v databázi stránku s touto URL a zjistí, že k ní patří prezenter `NewsPresenter` a výchozí akce *default*. Vytvoří tedy instanci třídy `Nette\Application\PresenterRequest`, v parametrech jí předá název prezenteru `NewsPresenter` a akci *default* a předá ji k dalšímu zpracování.

Co se ale stane, vyžádá-li si klient URL `novinky/detail/nase-prvni-novinka`? Přímo tu v databázi router nenalezne. A zde přichází na scénu položka v databázi *page-type.overloadAbility*. Bude-li mít pro stránku s URL `novinky` podobu `<action>/<id>`[13], router požadavek vyhodnotí jako požadavek na existující stránku s URL `novinky` s tím, že v parametrech instance třídy `Nette\Application\PresenterRequest` předá namísto výchozí hodnoty akce akci `detail`, a k tomu všemu navíc aplikaci předá i proměnnou *id* naplněnou hodnotou `nase-prvni-novinka`.

2.2.4. Router s napojením na databázi

Router je ta část frameworku, která zajišťuje překlad URL na interní požadavek aplikace. Jinými slovy router na základě analýzy klientem požadované URL rozhoduje o tom, co klient od aplikace požaduje (řečeno slovy Nette která akce kterého prezenteru a s kterými parametry se má vykonat).

Router, který by uměl pracovat s databází, není nativní součástí Nette (což je vzhledem k nezávislosti frameworku na žádném databázovém systému logické). Hlavní vestavěný router pracuje tak, že požadovanou URL za použití regulárních výrazů postupně porovnává s vývojářem nadefinovanými maskami, a to až do nalezení takové, která požadované adrese tzv. vyhovuje (anebo dokud nevyčerpá všechny dostupné masky, což eventuálně vede na HTTP chybu 404). Router URL adresu rozdělí podle schématu obsaženého ve vyhovující masce na jednotlivé čini-

[13] Zápis `<action>/<id>` představuje předpis, který určuje, že daný fragment URL bude rozdělen podle lomítka na dvě části, z nichž první bude brána jako název akce a druhá jako hodnota parametru `id`. To je následně formou parametrů uvnitř instance `Nette\Application\PresenterRequest` předáno prezenteru.

tele (typicky prezenter, akci a id) a vrátí aplikaci inicializovanou instanci třídy `Nette\Application\PresenterRequest`.

Router, který bude klientem požadovanou URL porovnávat s adresami stránek uložených v databázi si musíme naimplementovat sami. Jelikož se bezesporu jedná o jednu z nejzajímavějších nadstaveb frameworku v rámci naší aplikace, nastiňme si podrobněji, jak bude takový router fungovat.

Překlad URL na `Nette\Application\PresenterRequest`

1. Router si z dodané instance `Nette\Web\IHttpRequest` zjistí relativní URL požadavku. Pokud je relativní URL prázdná, je vyžadována úvodní stránka.

2. Router otestuje, zda se v databázi nachází stránka s URL shodnou s požadovanou. Pokud ano, nalezli jsme hledanou stránku.

3. Pokud se ve struktuře webu stránka s URL shodnou s požadovanou nenachází, je ještě možné, že se jedná o nějakou „přetíženou verzi“. Při ověřování této možnosti budeme postupovat následovně: z databáze vybereme všechny stránky, jejichž URL je začátkem URL požadavku. Jinými slovy bude-li požadovaná URL `novinky/detail/nase-prvni-novinka`, vybereme z databáze stránku s URL `novinky`, protože požadovaná URL začíná textem „novinky“.

   Následně rozdělíme podle lomítek URL požadavku *podle obsažených lomítek* na jednotlivé činitele (ve zmíněném příkladě na tři: `novinky`, `detail` a `nase-prvni-novinka`). Poté postavíme stranou *posledního* činitele (v našem příkladu `nase-prvni-novinka`) a zbylé spojíme za použití lomítek nazpět do jednoho řetězce. Zkontrolujeme, zda máme ve struktuře webu stránku s URL „o jednoho činitele kratší“ (v našem příkladu stránku s URL `novinky/detail`). Pokud ji máme, zkontrolujeme u nalezené stránky hodnotu položky *overloadAbility*. Není-li `null` (tzn. je-li výchozí akce daného typu stránky „přetížitelná“), nalezli jsme hledanou stránku. Nakonec již jen zanalyzujeme hodnotu *overloadAbility*, abychom věděli, zda činitel postavený stranou představuje akci (`<action>`) či například *id* (`<id>`).

Představme si ale nyní, že jsme v našem příkladu ani v tomto kroku ještě neuspěli a stránku s URL `novinky/detail` jsme ve struktuře webu nenalezli. Budeme tedy pokračovat. Postavíme stranou dalšího činitele a budeme ve struktuře webu hledat stránku s URL `novinky`. Tu nalezneme a jelikož má položku *overloadAbility* hodnotu `<action>/<id>`, víme, že z „odstavených" činitelů má být `detail` názvem akce a `nase-prvni-novinka` hodnotou parametru *id*. Nalezli jsme tedy hledanou stránku a správně rozpoznali její „přetížení".

Kompletní ukázka implementace takového routeru je k vidění v aplikaci přiložené k této práci. Oproti popsané funkcionalitě navíc ještě disponuje schopností určit z požadované URL jazykovou verzi, která může být předána v podobě zkratky o dvou znacích na začátku URL (oddělené od případných dalších částí adresy standardně lomítkem).

2.2.5. Struktura prezenterů a modulů v administraci

Frontend systému je koncepčně triviální. Díky routeru s napojením na databázi můžeme pro všechny typy stránek na webu jednoduše vytvořit potřebné prezentery s libovolnými akcemi. Ty mohou využívat libovolných modelových tříd a ve výsledku jsou možnosti omezené prakticky jen možnostmi frameworku (či možnostmi jazyka PHP). Všechny prezentery frontendu budou v modulu `Front`.

Všechny prezentery backendu, tj. celou administraci, umístíme do modulu `Admin`. Zde si však dovolíme zavedení ještě jedné úrovně modulů. Již dříve jsme zmínili, že jednotlivé typy stránek budou zastoupeny různými potomky třídy `Presenter`. Tak k tomu framework přímo navádí.

Vytváření, úpravu a mazání stránek můžeme koncepčně vyřešit následovně: pro každý prezenter přítomný ve frontendu a zastupující určitý typ stránky bude existovat i analogický prezenter v backendu. Zatímco prezenter ve frontendu bude mít za úkol vykreslit stránku daného typu tak, jak je určena pro návštěvníka webu, prezenter v backendu bude mít za úkol vykreslení vstupních formulářů pro její vytvoření a úpravu. Tyto prezentery v backendu (které mají logickou návaznost na prezentery ve frontendu) pro přehlednost sdružíme do submodulu `Pages`. To proto, že jemu nadřazený modul `Admin` bude obsahovat více prezenterů než pouze ty, které zastupují

konkrétní typy stránek. Jejich vyčlenění do samostatného modulu tedy přidá systému na přehlednosti.

2.2.6. Detail implementace

Příklad kompletní implementace systému podle navrženého konceptu je přílohou práce.

V přiložené implementaci je také znázorněna snadná rozšiřitelnost diskutovaného základu. Oproti samotné správě stránek a jejich (X)HTML kódu obsahuje totiž ukázka následující moduly:

Novinky

Modul novinkami s možností rozdělení každé novinky na perex a pokračování. Na úvodní stránce webu se zobrazuje perex poslední novinky a odkaz na stránku s přehledem novinek. Přehled je řazen chronologicky a využívá stránkování. Dále je generována stránka s detailem novinky obsahující novinku v jejím úplném znění.

Kontextová fotogalerie

Každá obsahová stránka může volitelně obsahovat fotogalerii, která se zobrazí pod hlavním obsahem. K zobrazení náhledů je použit posuvný pás implementovaný na straně klienta a detaily fotek se otevírají ve vlastní překryvné vrstvě. Ta umožňuje zároveň i přechod mezi jednotlivými snímky. Na posuvném pásu s náhledy jsou dále ještě odkazy na kontaktní stránku a na stránku s objednávkovým formulářem – adresy obou stránek jsou dynamicky generovány z hierarchie webu.

V administraci je možné snímky libovolně nahrávat, mazat i uživatelsky přívětivým tažením myší přeskupovat.

Kontaktní a objednávkové formuláře

Na míru vytvořené formuláře, v ukázce odesílané na e-mail nastavitelný v administraci. Z technického hlediska by nebylo o nic náročnější prezentovat data z odeslaných formulářů v administraci. E-mail z objednávkového formuláře je pro lepší přehlednost a ergonomii ve formátu HTML.

Vývoj ceny hliníku

Modul se stará o dvě věci: za prvé automaticky stahuje a parsuje ze vzdáleného zdroje aktuální cenu hliníku a za druhé vykresluje graf vývoje ceny hliníku z hodnot zadaných administrátorem v administraci. Jedná se o příklad velmi specifického modulu, jehož předchozí existenci bychom nemohli očekávat u žádného hotového volně dostupného CMS systému.

Všechny tyto moduly bylo díky rychlosti vývoje v Nette Frameworku velmi nenáročné vyvinout a do systému zakomponovat. Zároveň, díky kvalitnímu návrhu, ke kterému Nette Framework vývojáře vede, vynikají svou znovupoužitelností. Lze je velmi snadno použít v prakticky jakémkoliv dalším Nette projektu.

2.2.7. Závěr

V kontextu práce je poměrně důležitá náročnost takové implementace, která byla 100 hodin čistého vývojářského času: 80 hodin zabrala implementace jádra systému a 20 hodin poté implementace ukázkových speciálních modulů. To je pro mnoho firem plně přijatelná investice a vývoj vlastního řešení se tedy pro ně stává přijatelnou alternativou k použití některého zdarma dostupného CMS systému a řešením mnohých jejich nedostatků.

Zejména pak mají-li klienti firmy velmi často vskutku specifické požadavky, může být pro firmu nastíněné řešení díky jeho efektivní rozšiřitelnosti velmi výhodné.

Závěrem bychom měli zdůraznit, že představený vlastní systém bychom neměli prezentovat jako systém, který je ve zkratce „lepší“ než systémy analyzované v první části práce. CMS systémy samotné a i požadavky na ně vynikají tak velkou variabilitou, že jejich kvalitní vzájemné srovnání by svým rozsahem výrazně překročilo hranice této práce. Představené vlastní řešení má sloužit jen jako příklad toho, že lze i s omezeným rozpočtem vyvinout kvalitní základ CMS systému, který netrpí nedostatky, které mohou zrovna nám na populárních volně dostupných CMS systémech vadit.

3. Závěr

V první části práce byly představeny tři populární volně dostupné CMS systémy Drupal, CMS Made Simple a TYPOlight. Na systémy bylo nahlíženo ze třech pohledů: z pohledu hodnotícího stav komunity, podpory a dostupnosti doplňků, z pohledu administrátora a z pohledu vývojáře. Byly prezentovány různé závažné nedostatky, kterými tyto systémy navzdory své popularitě trpí.

Součástí první části práce je zároveň i pojednání o možné podstatě některých problémů, na které lze u těchto systémů často narazit.

Pro subjekty (firmy i jednotlivce) zabývající se webdesignem (dále jen pro „wedesignery“) se jako vhodná alternativa k volně dostupným CMS systémům nabízí vývoj vlastního řešení, zejména pak řešení založeného na některém z moderních PHP frameworků. Tohoto východiska webdesigneři v praxi často využívají – mimo jiné právě proto, že jejich vlastní řešení pak zpravidla netrpí nedostatky, které konkrétně jim na volně dostupných systémech vadí. Tím je pro ně vývoj vlastního základu CMS systému jedním z možných řešení, jak se s nedostatky volně dostupných systémů vypořádat.

V úvodu druhé části práce byly uvedeny další všeobecné výhody vlastních řešení založených na moderních frameworcích a následně byla navržena koncepce CMS systému v Nette Frameworku. Bylo demonstrováno, že návrh a vývoj dobře rozšiřitelného systému nemusí být nijak nákladný a může být efektivní – klíčový je otevřený návrh a co největší využití možností frameworku.

Přílohou práce je ukázka kompletní implementace systému podle navrženého konceptu. Její součástí je i několik modulů na míru, které ukazují, jak snadno lze takový systém rozšiřovat a jak snadno v něm lze realizovat i velmi specifické požadavky. Přiložená ukázka se svou implementační náročností 100 hodin zároveň dokazuje, že vývoj vlastního řešení nemusí být tak nákladný, jak by se mohlo na první pohled zdát.

Hlavní přínos práce

Hlavním přínosem práce k dané problematice (bráno z hlediska originality) je v první řadě návrh a ukázka realizace vlastního řešení CMS systému v prostředí relativně mladého frameworku Nette. Autorovi práce není známo, že by kdy někde byla koncepce či přímo ukázka řešení CMS systému v prostředí frameworku Nette publikována.

Uvedený koncept spolu s ukázkou implementace může mít značný význam pro každého, kdo řeší vývoj vlastního CMS systému: koncept mu může být inspirací, implementace pak přímým zdrojem zdrojových kódů. Práce jako celek pak může být přínosná pro každého webdesignera, který se aktuálně rozhoduje, jaký systém typu CMS by optimálně řešil běžné požadavky jeho klientů.

Literatura

[1] KAPRÁL, A. *Implementace vlastního redakčního systému (CMS) do již existující firemní webové prezentace. Projekt s prototypovým řešením*. Bakalářská práce. VŠE Praha, 2008.

[2] DARIE, C. *AJAX a PHP: tvoříme interaktivní webové aplikace profesionálně*. Brno: Zoner Press, 2006. ISBN 80-86815-47-1.

[3] KOSEK, J. *XML a PHP5*. Praha: Grada Publishing, 2005

[4] ZGRAJA, M. *Aktuální trendy ve vývoji webových aplikací*. Bakalářská práce. VŠE Praha, 2007.

[5] CÍSAŘ, D. *Bezpečnost webových aplikací (PHP)*. Bakalářská práce. VŠE Praha, 2007.

[6] *PHP triky - Verze PHP v ČR – únor 2010*. [cit. 7. 5. 2010]. Dostupný z WWW: http://php.vrana.cz/verze-php-v-cr-unor-2010.php

[7] *PHP: History of PHP*. [cit. 7. 5. 2010]. Dostupný z WWW: http://www.php.net/manual/en/history.php.php

[8] *HTML5*. [cit. 8. 5. 2010]. Dostupný z WWW: http://dev.w3.org/html5/spec/Overview.html

[9] *adaptive path » ajax: a new approach to web applications*. [cit. 7. 8. 2010]. Dostupný z WWW: http://www.adaptivepath.com/ideas/essays/archives/000385.php

[10] *About Native XMLHTTP*. [cit. 7. 8. 2010]. Dostupný z WWW: http://msdn.microsoft.com/en-us/library/ms537505(VS.85).aspx

[11] *Drupal 7 Alpha Preview*. [cit. 7. 8. 2010]. Dostupný z WWW: http://www.youtube.com/watch?v=DaofScfCF1w

[12] *History | drupal.org*. [cit. 16. 8. 2010]. Dostupný z WWW: http://drupal.org/about/history

[13] *Community Initiatives | drupal.org*. [cit. 17. 8. 2010]. Dostupný z WWW: http://drupal.org/community-initiatives

[14] *Drupal Usability Research Report*. [cit. 17. 8. 2010]. Dostupný z WWW: http://groups.drupal.org/files/DrupalUsabilityResearchReport.pdf

[15] *Usability, usability, and usability | Dries Buytaert*. [cit. 1. 12. 2010]. Dostupný z WWW: http://buytaert.net/usability-usability-and-usability

[16] *University of Baltimore Drupal Usability Study*. [cit. 1. 12. 2010]. Dostupný z WWW: http://video.google.com/videoplay?docid=-3942663057311698508#

[17] *Drupal Deployment*. [cit. 2. 12. 2010]. Dostupný z WWW: http://www.slideshare.net/eaton/drupal-deployment-presentation

[18] *Deployment | drupal.org*. [cit. 2. 12. 2010]. Dostupný z WWW: http://drupal.org/project/deploy

[19] *CMS Made Simple - About Us*. [cit. 2. 12. 2010]. Dostupný z WWW: http://www.cmsmadesimple.org/about-link/about-us/

[20] *CMS Made Simple Documentation*. [cit. 2. 12. 2010]. Dostupný z WWW: http://wiki.cmsmadesimple.org/index.php/Main_Page

[21] *Inversion of Control Containers and the Dependency Injection pattern*. [cit. 4. 12. 2010]. Dostupný z WWW: http://martinfowler.com/articles/injection.html#InversionOfControl

[22] *TYPOlight becomes Contao - Contao Open Source CMS (fka TYPOlight)*. [cit. 4. 12. 2010]. Dostupný z WWW: http://www.contao.org/news/items/typolight-becomes-contao.html

[23] *PHP Frameworks*. [cit. 6. 12. 2010]. Dostupný z WWW: http://www.phpframeworks.com

[24] *Escapování - definitivní příručka » phpFashion*. [cit. 6. 12. 2010]. Dostupný z WWW: http://phpfashion.com/escapovani-definitivni-prirucka

[25] *Co je Cross-Site Request Forgery a jak se mu bránit - Zdroják*. [cit. 6. 12. 2010]. Dostupný z WWW: http://zdrojak.root.cz/clanky/co-je-cross-site-request-forgery-a-jak-se-branit/

[26] *Trygve/MVC*. [cit. 6. 12. 2010]. Dostupný z WWW: http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html

[27] *Nette Framework | Nette Framework*. [cit. 6. 12. 2010]. Dostupný z WWW: http://nette.org

[28] *About the Foundation | Nette Foundation*. [cit. 6. 12. 2010]. Dostupný z WWW: http://nettefoundation.com

[29] *Velký test PHP frameworků: Zend, Nette, PHP a RoR - Root.cz*. [cit. 6. 12. 2010]. Dostupný z WWW: http://www.root.cz/clanky/velky-test-php-frameworku-zend-nette-php-a-ror/

[30] *MySQL :: Managing Hierarchical Data in MySQL*. [cit. 6. 12. 2010]. Dostupný z WWW: http://dev.mysql.com/tech-resources/articles/hierarchical-data.html

A. DVD s implementací CMS systému podle návrhu

Přílohou práce je DVD s kompletní implementací vlastního řešení CMS systému ve frameworku Nette podle návrhu popsaného ve druhé kapitole práce.

V kořenovém adresáři DVD je k dispozici soubor `readme.txt`, který obsahuje kompletní návod ke zprovoznění aplikace a k jejímu instantnímu naplnění předpřipraveným ukázkovým obsahem.

V případě zájmu je celá aplikace zprovozněna i naplněna vzorovým obsahem on-line na adrese http://klient.mravenec.net/bp/document_root/. Administrační rozhraní se nachází na adrese http://klient.mravenec.net/bp/document_root/cs/admin, kde uživatelské jméno je *admin* a heslo je *prilohaPrace486*.