VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

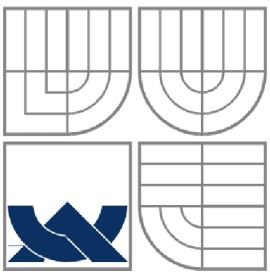
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SEO – OPTIMALIZACE PRO VYHLEDÁVČE

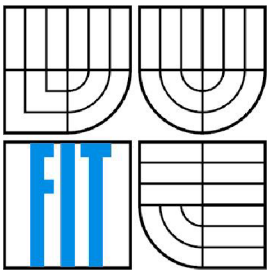
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. Jan Štefl
AUTHOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SEO – OPTIMALIZACE PRO VYHLEDÁVČE

SEO – SEARCH ENGINE OPTIMALIZATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Jan Štefl

VEDOUCÍ PRÁCE
SUPERVISOR

Prof. Ing. Tomáš Hruška, CSc.

BRNO 2007

Abstrakt

Viditelnost stránky na webu předpokládá její umisťování na předních pozicích ve výsledcích vyhledávání různých vyhledávačů, pro daná klíčová slova. Optimalizace webu pro vyhledávače je soubor pravidel, která by měla splňovat každá stránka. V této práci je popsáno vše podstatné, co se této problematiky týká, včetně metodologie usnadňující tvorbu správně napsaných stránek.

Klíčová slova

optimalizace pro vyhledávače, viditelnost na internetu, seo metodologie

Abstract

Visiblity of page assume that is situated at top positions in search engines results for particular keywords. Search engine optimization is collection of rules, that should pass every page. In this thesis is described all essences of this technique, including methodology, that helps create optimized pages.

Keywords

search engine optimization, visibility on the internet, seo methodology

Citace

Štefl Jan: SEO – Optimalizace pro vyhledávače. Brno, 2007, diplomová práce, FIT VUT v Brně.

SEO - optimalizace pro vyhledávače

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Prof. Ing. Tomáše Hrušky, CSc.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jméno Příjmení
Datum

Poděkování

Chtěl bych poděkovat mému vedoucímu Prof. Ing. Tomášovi Hruškovi, Csc za věcné připomínky při závěrečné kompletaci práce.

Obsah

Seznam tabulek

Seznam ilustrací

1 Úvod

SEO - *Search engine optimization* tedy optimalizace pro vyhledávače. Jedná se o proces zdokonalování kvality a objemu provozu mezi webovou stránkou a vyhledávačem (Google, Yahoo, Seznam, …) jako důsledek přirozeného vyhledávání klíčových slov.

Čím lépe je stránka vyhledávači ohodnocena, tím výše je umístěna v SERPs (viz slovníček pojmů) a má tedy i vyšší návštěvnost.

SEO též pomáhá stránky optimalizovat pro různé typy vyhledávání,od obrázků, přes lokální vyhledávání až po vyhledání specifické pro různá odvětví.

Cílem této práce je popsat vše zásadní, co s touto problematikou souvisí a navrhnout metodiku usnadňující vývoj optimalizovaných, resp. správně napsaných, stránek.

Budou probrány faktory ovlivňující viditelnost webu včetně diskuze jejich důležitosti. Bude zmíněna optimalizace z hlediska různých geografických oblastí a jazykových modifikací včetně přehledu dostupných nástrojů. Jedna z kapitol se bude věnovat i černým technikám, tzv. Black Hat SEO.

1.1 Historie

První optimalizace pro vyhledávače se začaly objevovat někdy v polovině devadesátých let. Jednalo se o tzv. katalogizovaný přístup, kde autor stránek obvykle poslal odkaz na své stránky přes nějaké vyhledávačem poskytnuté rozhraní. Proces zahrnoval nahrání stránky na server, kde byla uložena do systémových struktur. Následně pak indexér extrahoval různé informace, jako například která slova a kde jsou obsažena, prováděl jejich vahování a v neposlední řadě získával odkazy pro pozdější procházení.

Majitelé stránek si uvědomili, jak moc je výhodné být na předních pozicích vyhledávačů, což dalo prostor pro vznik bílých (white hat) a černých (black hat[1]) technik optimalizací stránek pro vyhledávače. Tak trochu ironií je, že první zmínka o „*search engine optimization*“ je spjata se spam zprávou zaslanou na usenet 26. července 1997 (viz [12]).

První verze vyhledávacích algoritmů byly založeny na analýze meta elementu obsahující klíčová slova resp. na tzv. ALIWEB souboru (*Archie Like Indexing for the WEB* viz [13]). Nicméně docházelo k častému zneužíváni těchto technik a nezřídka byla úmyslně podstrkována jiná (atraktivnější) klíčová slova, než relevantní. Stávalo se také, že zdrojová HTML stránka byla modifikována poskytovatelem webového prostoru ve snaze získat lepší pozici ve vyhledávačích. Avšak nesprávné a nekompletní údaje v meta elementech vedly k průběžnému znehodnocování SERPs. Bylo tedy zapotřebí přistoupit k jiným konceptům.

1 Black hat – černý klobouk. Vzpomínka na staré westerny, kde zlí kovbojové měli černé klobouky, na rozdíl od hodných s bílými

Dalším krokem byly algoritmy založené na komplexnějším ohodnocení stránky a na faktorech, autory webů hůře manipulovatelných. Ke slovu se dostává algoritmus vyvinutý dvojicí Jim Page a Sergey Brin (zakladatelé firmy Google), který dokáže vyčíslit důležitost stránky v kontextu internetu. Jeho produktem je číslo, tzv. PageRank, jenž je funkcí množství a kvality příchozích odkazů (*inbound links*).

V roce 1998 byla založena firma Google Inc., a její vyhledávač si záhy získal mnoho uživatelů, například i svou jednoduchostí. V potaz byly brány jak nepřímé (*off-page*), tak i přímé (*on-page*) faktory, což umožňovalo lepší analýzu relevantnosti obsahu. Jeden z nepřímých faktorů byl zmiňovaný *PageRank* a právě on se stal ve velké míře terčem zneužíváni.

Tvůrci webů po zjištění, že čím více odkazů, tím lépe, začali odkazy vyměňovat a obchodovat s nimi, a to v masivním měřítku. Vzniklo tak mnoho odkazových schémat – farem, jejichž jediným cílem bylo odkazovat na weby obsahující spam. To logicky opět vedlo ke snižování kvality SERPs.

Situace dospěla tak daleko, že se letos (rok 2007) firmy provozující vyhledávače rozhodly vůbec nezveřejňovat faktory ovliňující viditelnost ve výsledcích vyhledávání.

1.2 Základní principy

1.2.1 Jak fungují vyhledávače

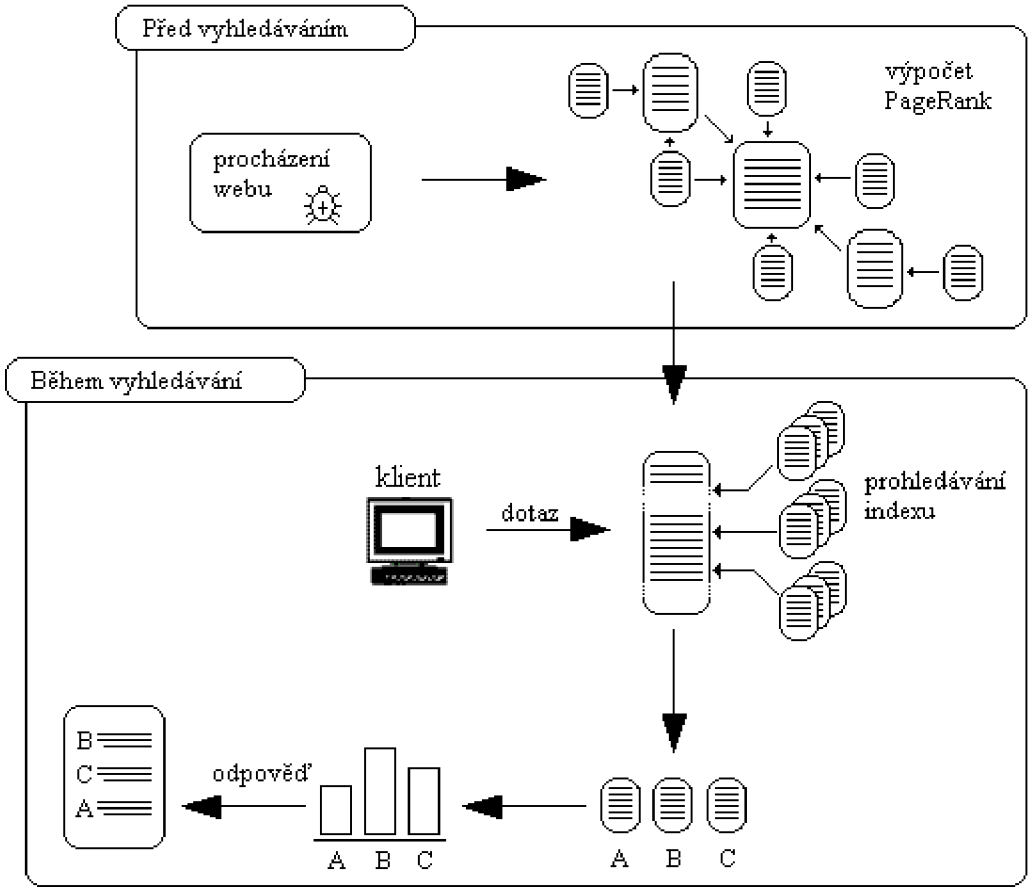
Společnost provozující vyhledávač musí disponovat nemalou skupinu počítačů, tzv. botem (v případě Google se jmenuje Googlebot), který neustále prochází obrovské objemy stránek. Procházení internetu je algoritmický proces, tedy je průběžně automaticky analyzováno, které servery se mají projít, jak často a kolik konkrétních stránek z daného serveru.

Proces procházení (*crawling,* subsystém procházející síť se nazývá *crawler* nebo také *spider*) začíná seznamem adres webových stránek. Bot v obdrženém obsahu detekuje odkazy a ukládá je na seznam adres, určených pro pozdější navštívení.

V okamžiku, kdy bot přijde na stránku, vytvoří si její kopii za účelem konstrukce indexu přítomných slov. Též si zaznamená informaci o jejich pozici.

Když uživatel položí dotaz, bot začne v indexu hledat odpovídající stránky, přičemž následně zobrazí nejrelevantnější nálezy. Relevance je udávána mnoha faktory (u Google jich je přes 200).

Základní princip vyhledávačů je zachycen na obrázku 1.1.



Obrázek 1: Základní princip vyhledávačů

1.2.2 PageRank

Je pravděpodobně nejznámější algoritmus pro ohodnocení důležitosti stránky (tedy ne celého webu) v kontextu internetu. Alternativními algoritmy jsou například HITS, TrustRank nebo S-rank (využívaný vyhledávačem seznam.cz). Jak bylo zmíněno výše, autory jsou Lary Page a Sergey Brin – spoluzakladatelé společnosti Google Inc.. PageRank je obchodní známka firmy Google Inc., algoritmus je patentován (U.S. Patent 6,285,999) univerzitou ve Stanfordu.

Popis algoritmu

PageRank vychází z demokratické povahy webu, kde každá stránka je identifikována odkazem. Ve své podstatě Google interpretuje odkaz ze stránky A na stránku B jako udělení hlasu pro stránku B stránkou A. Vedle počtu hlasů je zkoumáno i kdo hlasoval. Hlas udělený stránkou, která byla identifikována jako „důležitá“, má větší váhu hlasu, což pomáhá zvýšit i důležitost stránky pro kterou bylo hlasováno.

PageRank reprezentuje pravděpodobnost, že uživatel náhodně procházející dostupné odkazy na navštívených stránkách, dosáhne právě určité stránky.

Algoritmus může být aplikován na libovolnou množinu vzájemně se odkazujících entit. Numerická váha, přidělená libovolnému elementu *E* se též nazývá *PageRank E*, vyjadřujeme ho *PR(E)* a pohybuje se v intervalu 0-10.

Na začátku algoritmu mají všechny entity stejnou váhu; výpočet vyžaduje několik iterací.

Zjednodušená verze

Váha (PageRank) stránky *u* je vyjádřena vztahem:

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)} \tag{1.1}$$

Kde *v* je stránka odkazující na stránku *u*. *PR(v)* je celková váha stránky *v*. *L(v)* je celkový počet odkazů na stránce *v* a B_u je množina všech stránek odkazujících na stránku *u*.

Podrobnější informace viz [14].

2 Faktory ovlivňující viditelnost na webu

Tato kapitola pojednává o vlivech různých faktorů na umisťování stránky v SERPs. Členění odpovídá následujícím hlediskům:

- faktory svázané se stránkou
- faktory dané časem
- externí faktory

Každý faktor je pojmenován a popsán, s identifikací jeho důležitosti dle [2] („faktor nehodnocen" v tabulce oznamuje, že faktor nebyl v článku diskutován). Například Google rozpoznává asi 200 různých faktorů, proto je třeba předem avizovat, že se nejedná o kompletní seznam, nicméně vše podstatné je zmíněno.

Z důvodu neexistence standardu, jsou pravidla občas velmi vágní.

2.1 Faktory svázané se stránkou

Jsou přímo určeny stránkou jako takovou, tedy obsaženými texty a jejím zdrojovým kódem. V současné době vzniká snaha přejít na jiné faktory, důvodem je jejich snadná zmanipulovatelnost, nezřídka zneužitá pro spam. Nicméně to neznamená, že nejsou důležité. Dělíme je na:

- viditelné faktory
- neviditelné faktory

2.1.1 Viditelné faktory

Mají větší vliv než neviditelné, představují to, co uživatel vidí a jsou proto důvěryhodnější.

Titulek stránky

Jeden z nejzásadnějších faktorů. Jedná se o text umístěný uvnitř elementu `<title></title>`, který je umístěn v hlavičce stránky. Je zobrazován jak na titulkové liště prohlížeče resp. v záhlaví stránky, tak i ve výsledcích vyhledávačů. Z tohoto vyplývá, že každý titulek by měl být co nejvýstižnější, a pokud možno, bohatý na klíčová slova. Na druhou stranu titulek *přeplněný* klíčovými slovy pravděpodobně nedosáhne předpokládaného výsledku (*keyword stuffing*). Častou chybou je, že tvůrci webu ponechávají všem stránkám stejný titulek.

Bývá zvykem začínat titulek názvem společnosti, nicméně osobně jsem shledal za užitečné tuto informaci vkládat až nakonec, například: *Produkt* | *Kategorie* | *Obchod.cz*. Tedy, čím výstižnější informace o stránce, tím dříve ji v titulku zobrazit. Za povšimnutí také stojí určitá asymetrie titulku a URL stránky. K předešlému příkladu by vhodné URL mohlo vypadat následovně: *obchod.cz/kategorie/produkt*.

Pokud uživatelé vytvářejí příchozí odkazy (*inbound links*), nezřídka použijí titulek k popisu odkazu, což je velmi pozitivní skutečnost potvrzující, že titulek by měl být co nejvýstižnější. Poutavost titulku se kladně podílí na CTR (viz slovníček pojmů) odkazu.

Tabulka 1: Faktor - titulek stránky

Shrnutí	Celkový vliv
● výstižnost ● unikátnost pro každou stránku ● přiměřeně obsažena vhodná klíčová slova	● zásadní vliv

Nadpisy

Elementy definující nadpis jsou tvaru `<hX></hX>`, kde X je číslo od 1 do 6. Jejich cílem je usnadnění orientace v textu. V současné době nehrají takovou roli jako dříve (z důvodu jejich zneužívání), nicméně stále hrají svou roli v celkovém hodnocení stránky a rozhodně není vhodné je opomíjet.

V nadpisech by měla být umístěna klíčová slova vystihující text.

Někteří odborníci tvrdí, že H1 je druhá nejvýznamnější značka, nicméně na předních pozicích ve výsledcích vyhledávání, existují stránky i bez této značky, napříč tomu, že její nevyužití může působit jako prohřešek vůči organizaci textu a přístupnosti (viz slovníček pojmů) obecně.

Nadpisy vyšších úrovní hrají pravděpodobně o něco menší roli.

Nadpisy všech úrovní by se neměly doslovně shodovat s titulkem a popisem stránky, vypozorují-li vyhledávače příliš častou šablonovitost, hrozí stránce postihy.

Tabulka 2: Faktor - nadpisy

Shrnutí	Celkový vliv
● výstižnost ● klíčová slova ● používání těchto elementů vede k dobrému členění textu – měly by být proto využívány	● H1 – vysoký vliv ● H2 – H6 – střední vliv

Vztah obsahu stránky a klíčových slov (tématická analýza)

Vyhledávače dělají analýzu klíčových slov na základě textu. Je tedy přirozené, aby se v textu "rozumně" vyskytovala a to nejlépe v různých tvarech a v různých pozicích (z hlediska sousedních slov). Mezi osvědčené praktiky patří tzv. „důvtipné opakování", kdy se klíčová slova vyskytují jak na konci, tak na začátku věty.

Bohužel nikde není řečeno, kdy je stránka relevantní a kdy už překročila hranici a je považována za nežádoucí.

Tabulka 3: Faktor - vztah obsahu stránky a klíčových slov (tématická analýza)

Shrnutí	Celkový vliv
• přítomná relevantní klíčová slova • klíčová slova v různých tvarech a pořadích	• silný vliv • nadužívání cílových klíčových slov má silný negativní vliv

Odchozí odkazy (outbound links)

Odkazování na nežádoucí stránky vede ke snížení rankingu. Vyhledávače si tedy dokážou udělat obrázek o stránce na základě *sousedů stránky.*

Pokud odkazy vedou na stránku podobně tématicky laděnou, tím lépe.

Tabulka 4: Faktor - odchozí odkazy

Shrnutí	Celkový vliv
• neodkazovat na podezřelé stránky • sdružovat tématicky podobné weby	• silný vliv

Klíčová slova v URL a doménovém jméně

Tímto je myšleno například doména *klíčovéslovo.cz*. V současné době je tento faktor minoritní nicméně ne úplně opovrženíhodný.

Vhodný název domény má pozitivní vliv na CTR. Klíčová slova v URL se oddělují pomlčkou. V doménovém jméně se doporučuje maximálně jeden tento symbol, větší počet vede k nedůvěryhodnosti. Jinými slovy, lepší dobrá značka, než doména plná klíčových slov.

Tabulka 5: Faktor - klíčová slova v URL a doménovém jméně

Shrnutí	Celkový vliv
● maximálně jeden oddělovač -	● částečný vliv

Vnitřní struktura a provázanost webu

Vyhledávače pravděpodobně činí předpoklad, že stránka bez interních odkazů, resp. stránka těmito odkazy přehlcená, je méně významná.

Odkazy uvnitř webu mohou mít vliv na pozici jiných interních stránek, záleží na reputaci stránky ze které je odkazováno. Svou roli hraji i sémantická příbuznost mezi oběma propojovanými stránkami – shodné téma má zpravidla pozitivní efekt.

Každý web by měl obsahovat mapu webu, která bude dostupná z každé stránky.

Dále s tímto faktorem bývá spojováno úskalí pojmenované *death by pagination,* nebo-li smrt způsobená stránkováním. Tento problém vzniká v případech, kdy je, například nějaký článek, rozdělen na více stránek a jednotlivé části jsou rozděleny pouze navigací typu „další“ resp. „předchozí“. Pro vyhledávače je pak těžké se dostat až na poslední část článku, proto je vhodné použít navigaci typu *<předchozí 1 | 2 | 3 další>*, kde jsou z každé stránky dostupné všechny ostatní oddíly.

Tabulka 6: Faktor - vnitřní struktura a provázanost webu

Shrnutí	Celkový vliv
● „přiměřené“ množství interních odkazů ● propojení tématicky shodných stránek má pozitivní efekt ● vytvářet mapu webu ● vhodná navigace	● zásadní vliv

2.1.2 Neviditelné faktory

Mají o něco menší vliv než viditelné, důvodem je jejich snadná, návštěvníkem nepostřehnutelná, manipulovatelnost. Je třeba brát v potaz fakt, že za neviditelný faktor může být pokládána i ta část stránky, která je zneviditelněná pomocí stylovacího jazyka CSS. Stránka obsahující nadměrné množství neviditelného textu bývá pro vyhledávače podezřelá a může v krajním případě způsobit penalizaci celého webu.

Jak už předchozí odstavec naznačil, viditelnost je myšlena z pohledu návštěvníka stránky.

Meta popis (`meta description`)

Jedná se o element `<meta name="description" value="popis stránky"/>` umístěný v hlavičce stránky. Často bývá zobrazován ve výsledcích zobrazování, může proto mít vliv na CTR. Má minoritní podíl v celkovém hodnocení, nicméně je vhodné ho nezanedbat.

Popisek, stejně jako titulek by měl být unikátní, zajímavý a výstižný.

Tabulka 7: Faktor - meta popis

Shrnutí	Celkový vliv
● měl by čtenáře upoutat	● částečný vliv

Meta klíčová slova (`meta keywords`)

Jedná se o element `<meta name="keywords" value="slovo1, slovo2, ...">`. S ohledem na skutečnost, že je velmi snadno zmanipulovatelný, a na existenci kvalitnějších metod analýzy stránky, je tento element zanedbatelný.

V době psaní tohoto textu pouze vyhledávač YAHOO hledí na tento element, zůstává otázkou jakou mu přikládá důležitost (pravděpodobně jen pro detekci překlepů v textu [2]).

Tabulka 8: Faktor - meta klíčová slova

Shrnutí	Celkový vliv
● dnes už zbytečný parametr	● minimální vliv

Atributy `alt` a `title`

Některé elementy nabízejí použití atributu `title` pro poskytnutí podrobnějších informací obsažených uvnitř elementu. Atribut `alt` slouží k textové reprezentaci obrázku.

Využívaní těchto atributů má částečný vliv (indexace obrázků), nicméně jsou zcela nepostradatelné z hlediska přístupnosti.

Tabulka 9: Faktor - atributy `alt` a `title`

Shrnutí	Celkový vliv
• používané pro indexaci obrázků • klíčová slova	• Z hlediska SEO částečný vliv • Z hlediska přístupnosti a i zásadní vliv

Vliv užitých html elementů

Vyhledávače chápou blokové elementy jako prostředek pro seskupování příbuzných textů. Blokové elementy by proto měly být používány s rozvahou, viz příklad:

```
<div>Psí</div>
<div>jídlo</div> je pravděpodobně méně relevantní než:
<div>psí jídlo</div>.
```

Tabulka 10: Faktor - vliv užitých HTML elementů

Shrnutí	Celkový vliv
• blokový element by měl tvořit rámec informace	• faktor nehodnocen

2.2 Faktory dané časem

2.2.1 Stáří stránky a domény

Čím jsou stránky starší, tím je na ně více odkazů, což samo o sobě zvyšuje jejich „kvalitu“. Starší stránky jsou vyhledávači zpravidla vnímány jako autoritativnější, zatímco na ty mladší může být nahlíženo jako na relevantnější k danému tématu. Koncept odpovídá modelu lidského chování – zákazník spíš bude nakupovat v obchodech, které jsou časem prověřené.

Hodnotně aktualizované stránky získávají na důvěryhodnosti.

Stáří domény je důležitější než stáří dokumentu, tedy stejný dokument bude na starší doméně hodnocen lépe než na mladší.

Vyhledávače resetují *aging value* po vypršení doby platnosti domény.

Tabulka 11: Faktor - stáří stránky a domény

Shrnutí	Celkový vliv
• aktualizovat obsah stránek (indikuje to jejich živost a aktuálnost) • pozor při nákupu domény, jejíž platnost již vypršela	• silný vliv

2.2.2 Stáří příchozích odkazů

Předpokládá se, že asi 3-4 měsíce starý odkaz, začíná hrát roli (z hlediska Google).Čím starší, tím lepší.

Tabulka 12: Faktor - stáří příchozích odkazů

Shrnutí	Celkový vliv
• čím starší, tím lepší	• silný vliv

2.2.3 Délka doby registrace doménového jména

Domény používané k nekalým účelům, jsou zpravidla registrovány pouze na jeden rok, je nevhodné si doménu registrovat na takovouto dobu.

Tabulka 13: Faktor - délka doby registrace doménového jména

Shrnutí	Celkový vliv
• raději registrovat doménu na déle než rok	• faktor nehodnocen

2.3 Externí faktory

Jedná se o faktory, které nejsou přímo spojené se spravovanou stránkou. V této kapitole je diskutované *okolí* stránky a jeho vliv na stránku samotnou.

Je zde například zmíněno, zda může autor konkurenčního webu nějakým způsobem negativně ovlivnit celkové hodnocení naší stránky.

2.3.1 Kvantita, kvalita a tématická relevance příchozích odkazů

Kvalita odkazu je posuzována podle počtu a kvality odkazů vedoucích na odkazující stránku. Čím vyšší kvalita, tím lépe. Bohužel neexistuje přesná definice kvality. Každý vyhledávač ji interpretuje jinak, proto nezbývá, než se spolehnout na cit a intuici.

S kvalitou odkazu souvisí i sémantická relevance propojovaných stránek. Je vhodné propojovat stránky s příbuzným tématickým zaměřením.

Skupiny stejných odkazů (umístěné na různých webech) se schématickým textem jsou podezřelé a mohou být vyhledávači znehodnoceny. Příliš mnoho příchozích odkazů z irelevantních zdrojů může vést k postihu stránky.

Otázkou je, zda může konkurence znehodnotit naše stránky odkazováním na ně z irelevantních zdrojů. Podle Google,v drtivé většině případů, nemůže útočník dosáhnout poškození webu, resp. ho takto vyřadit z indexu. (viz [5])

Výjimku v této oblasti tvoří sociální weby (např. myspace.com, last.fm, apod.). Kvalita příchozích odkazů z uživatelských profilů je vyhledávači zpravidla redukována.

Dle [1] bývají odkazy na stránkách typu *links.php* znehodnocovány, resp. není jim přikládána taková váha, jaká by mohla.

Tabulka 14: Faktor - kvantita, kvalita a tématická relevance příchozích odkazů

Shrnutí	Celkový vliv
• záleží na kvalitě odkazujícího webu • čím více odkazů z kvalitních webů, tím lépe • propojovat sémanticky příbuzné stránky (weby) • šablonovité odkazování z irelevantních webů vede k postihům	• silný vliv

2.3.2 Kvalita a relevance odkazů na externí stránky či weby

Tento faktor je důležitý, zejména pro začínající weby. Je vhodné odkazovat na kvalitní, tématicky shodné stránky.

Tabulka 15: Faktor - kvalita a relevance odkazů na externí stránky či weby

Shrnutí	Celkový vliv
• odkazovat na kvalitní, tématicky shodné stránky	• Silný vliv

2.3.3 Čeření odkazů (link churn)

Pokud jsou na stránkách určité odkazy v čase proměnlivé (*link churn* – vír odkazů), jsou stránky podezřívány z nekalých činností. Nicméně pokud se nejedná o stránky určené ke spamování apod., nemusí se tímto faktorem autor znepokojovat.

2.3.4 Míra přírůstku kvality odkazů

Web je podezřelý pokud je nový a je na něj odkazováno z příliš mnoha (tisíce) odkazů - avšak situaci je trochu jiná pokud ji odkazují autoritativní weby (viz [6]).

2.3.5 Text odkazu a okolní text

Příchozí odkazy, které mají sémanticky relevantní text (*anchor text*), mají pozitivní vliv na celkové hodnocení stránky. Stejně tak je nezanedbatelný okolní text odkazu, ten může hrát stejnou roli (někteří dokonce předpokládají, že je stejně důležitý jako samotný text tvořící odkaz [2])

S tímto faktorem jsou zmiňovány tzv. Google bomby.(viz [7])

Tabulka 16: Faktor - text odkazu a okolní text

Shrnutí	Celkový vliv
• text odkazu by měl, co nejvíce vystihovat odkazované místo • okolní text by měl sémanticky odpovídat odkazu	• zásadní vliv

2.3.6 Vzájemné odkazy (*reciprocal links*)

Pokud jsou v příliš vysoké míře, tak jsou zpravidla devalvovány. Obecně počet jednosměrných odkazů by měl být větší, než těch vzájemných - to je přirozené a dobře hodnocené.

Tabulka 17: Faktor - vzájemné odkazy

Shrnutí	Celkový vliv
• využívat jich s mírou	• faktor nehodnocen

2.3.7 Počet odkazů na stránce

Odkaz na stránku s malým počtem odkazů je zpravidla horší než odkaz na stránku s mnoha odkazy.

Tabulka 18: Faktor - počet odkazů na stránce

Shrnutí	Celkový vliv
• raději se vyhýbat odkazování na stránky bez odkazů	• faktor nehodnocen

2.3.8 IP adresy vzájemně odkazovaných webů

Pokud jsou stránky propojeny odkazy, které přicházejí z podobných IP adres, stávají se podezřelými.

Tabulka 19: Faktor - IP adresy vzájemně odkazovaných webů

Shrnutí	Celkový vliv
• Pro provoz více webů, které potřebujeme z nějakého důvodu provázat odkazy je vhodné využít služeb poskytovatele internetového prostoru nabízejícího zákazníkovi více IP adres, resp. každou aplikaci provozovat jinde.	• faktor nehodnocen

2.3.9 TLD odkazující domény (.cz, .com, …)

Určité domény ("edu", "gov" nebo "mil") podléhají striktnímu omezení a nelze je libovolně registrovat.

Obecně se má za to, že domény *edu* resp. *gov* jsou nejméně podezřelé, proto mají spíše vyšší váhu.

Tabulka 20: Faktor - TLD odkazující domény (.cz, .com, …)

Shrnutí	Celkový vliv
● určitou roli na celkové hodnocení stránky může mít i její koncová doména.	● částečný vliv

2.3.10 Pozice odkazu na stránce

Pozicí je myšleno místo zobrazení odkazu z hlediska rozložení stánky. Nejvíce jsou ceněny odkazy umístěny asi uprostřed stránky. Nejméně dole, resp. externí odkazy dole mohou být označeny za spam.

Tabulka 21: Faktor - pozice odkazu na stránce

Shrnutí	Celkový vliv
● uprostřed stránky pravděpodobně větší priorita ● zápatí mají minimální prioritu	● Faktor v článku nehodnocen

2.3.11 Validita webu

Validitou se rozumí správnost kódu dokumentu či stránky vzhledem k doporučením standardizační organizace W3C.

Z hlediska SEO nevýznamné, to ovšem neplatí o přístupnosti.

Tabulka 22: Faktor - validita webu

Shrnutí	Celkový vliv
● každá použitelná stránka by měla být validní ● validita vypovídá o úrovni tvůrce kódu	● minimální vliv z pohledu SEO ● zásadní vliv z pohledu přístupnosti

2.4 Negativní faktory

Následuje seznam, více už nestrukturovaných negativních faktorů.

2.4.1 Server je často robotům nepřístupný

Uživatel jistě nebude mít zájem navštěvovat stránky, které jsou často nedostupné, vyhledávače to proto zohledňují. Dokonce berou v potaz i časovou odezvu serveru, v případě že je velmi pomalá dochází k penalizacím.

Dle [2] 48 hodinový výpadek může mít za následek vyřazení z indexu.

Tabulka 23: Faktor - server je často robotům nepřístupný

Shrnutí	Celkový vliv
● zajistit plynulý provoz serveru	● silný vliv

2.4.2 Duplicitní obsah

Vyhledávače se zpravidla snaží předcházet duplicitnímu obsahu. Někteří experti dokonce tvrdí, že dochází k penalizaci stránek s duplicitním obsahem. Jedná se o diskutovanou oblast, nicméně duplicitní obsah rozhodně nijak pozitivně nepřispívá k celkovému hodnocení ve všech majoritních vyhledávačích.

V souvislosti s duplicitním obsahem zavedl Google tzv. doplňkový index (*suplemental index*). Právě do něj ukládá tyto a podobné stránky.

Tabulka 24: Faktor - duplicitní obsah

Shrnutí	Celkový vliv
● obsah stránek by měl být pokud možno unikátní	● silný vliv

2.4.3 Odkazování na nekvalitní a spam weby

Vyhledávače identifikují povahu webu i na základě sousedů. Odkazuje-li web na nedůvěryhodné zdroje bývá zpravidla označen také za nedůvěryhodný. Určitou roli sehrává poměr kvalitních a nekvalitních odkazů.

Tabulka 25: Faktor - odkazování na nekvalitní a spam weby

Shrnutí	Celkový vliv
• *Řekni mi s kým se stýkáš a já ti řeknu kdo jsi.*	• silný vliv

2.4.4 Šablonovitý obsah elementů `title` a `meta` na velkém počtu stránek

Jeden z dalších faktorů, který může stránku deklasovat do pomocného indexu. Vyhledávače se snaží získat ze stránky co nejvíce relevantních informací. Pokud se bude na stránce vyskytovat duplicitní obsah na těchto zásadních místech, těžko dosáhne lepších pozicí ve výsledcích vyhledávání.

Tabulka 26: Faktor - šablonovitý obsah elementů `title` a `meta` na velkém počtu stránek

Shrnutí	Celkový vliv
• unikátní titulky a popisky stránek	• silný vliv

2.4.5 Účast v odkazovacích službách nebo prodej odkazů

Tvůrci vyhledávačů tvrdí, že využívají sofistikované systémy založené na pravděpodobnosti a analytických algoritmech pro odhalování placených odkazů. Nicméně ani tak se nedaří identifikovat všechny, proto koupení si odkazu může být výhodné, nicméně mělo by být vždy dopředu zváženo a nikdy nejít s davem.

Obecně se dá říci, že čím více kvalitních odkazů má web k dispozici, tím více snese těch špatných.

Tabulka 27: Faktor - účast v odkazovacích službách nebo prodej odkazů

Shrnutí	Celkový vliv
• občas může být výhodné koupit si odkaz	• pokud je placený odkaz odhalen má to negativní vliv

3 Odkazy – tvorba a správa

Tato kapitola popisuje jak vytvářet vhodné odkazy a jak s nimi pracovat v kontextu celé aplikace.

3.1 Statické a dynamické odkazy

V době vzniku internetu neexistovaly jiné odkazy než statické. Statický odkaz identifikuje server a na něm konkrétní dokument, například URL www.mojedomena.cz/dokument.txt zpřístupní obsah souboru dokument.txt umístěného v kořenovém adresáři aplikace.

S příchodem skriptovacích serverových technologií se začínají vyskytovat odkazy dynamické. Od statických se liší tím, že aplikaci předávají vstupní parametry. Jako příklad může posloužit URL imaginárního elektronického obchodu: obchod.cz/get_product.php?id=1. Přibyl zde otazník, oddělující statickou část od parametrů a dále pak jednotlivé parametry oddělené symbolem &.

Problém dynamických stránek z hlediska SEO souvisí s faktorem duplicitního obsahu (viz kapitola 5). Vyhledávače vycházející z předpokladu, že různé vstupní parametry budou generovat různý výstup a to bez ohledu na skutečnost, že se jedná stále o stejný zdrojový soubor (get_product.php). V případě, že bychom měli URL typu obchod.cz/get_product.php?id=1&optional_param=12, je velmi pravděpodobné, že budeme generovat stránku velmi podobnou té bez volitelného parametru, nicméně z hlediska vyhledávačů web obsahuje duplicitní obsah.

Počet parametrů by měl jen výjimečně přesahovat dva. Web se pak vyhledávačům špatně prochází a obecně takové adresy nejsou pro uživatele příliš lákavé (což má negativní vliv na CTR).

Dále je doporučeno vyhýbat se parametrům obsahující v názvů id, například předávání session id touto cestou je velmi nevhodné a vyhledávače se mohou rozhodnout ignorovat takovouto stránku.

Je několik způsobů jak převést dynamické url na statické.

3.2 Tvorba vhodných odkazů

Ideální URL (*nice URL*) neobsahují parametry a jsou bohatá na klíčová slova, například tedy redblack.cz/band/sad-harmony.

V současné době vyhledávače hledí zejména na text reprezentující odkaz (*anchor text*), než na text odkazu samotného, nicméně výstižné pojmenování povede ke zvýšení CTR. Nezřídka se stává, že odkaz na stránku bude vytvořen následujícím způsobem:

```
<a href="http://redblack.cz/band/sad-harmony">http://redblack.cz/band/
sad-harmony</a>
```

Obecně tvorba takovýchto odkazů klade větší požadavky na prostředky (obvykle spolupráce s databází, případně s mapovacím souborem).

Je potřeba si uvědomit, přechod na „hezká URL“ může být velký problém, obzvláště pokud s touto možností nebylo v původním návrhu počítáno.

3.3 Konzistence odkazů

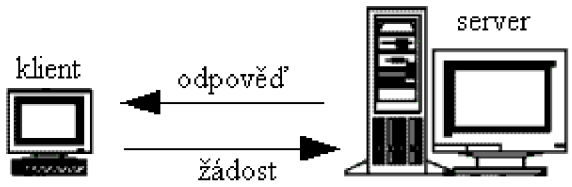
Bez ohledu, zda jsou URL statická nebo dynamická, je potřeba udržovat je v konzistentním stavu. Vyhledávače se ztotožňují s následujícími body:

- nerozlišují různé pořadí a počet parametrů
- rozlišují velikost písma
- `obchod.cz/kategorie/produkt` a `obchod.cz/kategorie/produkt/` jsou dva různé odkazy
- www.obchod.cz a obchod.cz jsou opět různé odkazy

Pokud nebude konzistence striktně dodržena, bude každý odkaz hodnocen vyhledávači zvlášť.

4 Relokace odkazů a HTTP stavové kódy

Internet je ve své podstatě koncipován jako architektura klient/server. Komunikace na aplikační vrstvě se provádí pomocí protokolu HTTP.



Obrázek 2: Architektura klient/server

Tedy klient pokládá žádosti a server odpovídá. Součástí každé odpovědi je informace jak dalece se podařilo žádost splnit, tzv. stavový kód (*status code*).

Následuje tabulka základních kódů, přičemž v praxi je vhodné využívat pouze stavy 301 (302), 404, 500. Ostatní je lepší, z důvodů různé interpretaci prohlížečů, opomenout. Kompletní výčet viz [8].

Tabulka 28: Stavové kódy HTTP protokolu

Stavový kód	Popis
300	Multiple Choices
301	Moved Permanently
302	Found
303	See Other (since HTTP/1.1)
304	Not Modified
305	Use Proxy (od HTTP/1.1)
307	Temporary Redirect (od HTTP/1.1)
400	Not Found
500	Internal Server Error

4.1 Přesměrovávání

Pokud je potřeba změnit URL stránky na jiné (například při přechodu na optimalizovaný web) je žádoucí, aby byla zachována hodnota odkazů (*link equity*).

Toho lze dosáhnout pomocí přesměrování ze starého URL na nové, tedy užitím stavového kódu 301 resp. 302, s nimiž je spjata informace, kde se požadovaná stránka nachází.

4.1.1 Stavový kód 301

V okamžiku, kdy robot (*crawler*) identifikuje přesměrování 301 deleguje informaci do jiného subsystému vyhledávače, který by měl tuto skutečnost reflektovat. Hodnota odkazu je v tomto případě zpravidla zachována (s možnou určitou časovou prodlevou).

```
HTTP/1.1 301 Moved Permanently
Date: Wed, 10 Dec 2007 09:50:39 GMT
Server: Apache/2.0.54 (Unix)
X-Powered-By: PHP/5.0.4
Location: http://www.example.com/new_page.php
Content-Length: 0
Connection: close
Content-Type: text/html; charset=ISO-8859-1
```

Příklad 4.1: Odpověď serveru obsahující stavový kód 301

4.1.2 Stavový kód 302

Indikuje, že staré URL není ještě zastaralé jen je obsah dočasně přesunut – v prohlížeči zpravidla nedochází k ukládání stránky do mezipaměti, pokud tak není řečeno explicitně (v hlavičce HTTP protokolu).

Problém tohoto kódu spočívá ve víceznačnosti, smysl totiž záleží na kontextu. V praxi se rozlišuje interní (v rámci domény) a externí (mimo doménu) přesměrování. Prohlížeče přesměrovávají v obou v případech, nicméně vyhledávače obvykle pouze reagují na interní přesměrování (Google a Yahoo!).

Externí přesměrováni (z hlediska stavového kódu 302) je problém. Pomoci něho může dojít ke krádeži obsahu z jiné stránky, jedná se o útok *302 Hijacking*. Tedy stránka na doméně A se tváří jako jiná stránka na doméně B, tím zvýší svůj PageRank. Dále využívá některou z technik zahalování (*cloaking*) pro přesměrování uživatele jinam.

Vyhledávače disponují stále se vyvíjejícími heuristikami pro odhalení tohoto útoku.

Zbývá tedy doporučit spíše se stavovému kódu 302 vyhnout a nahradit ho 301 nebo použít úplně jiný způsob přesměrování (např pomocí vkládání (*inkludování*) souborů pro interní přesměrování nebo načtení obsahu pomocí speciální knihovny – např. CURL v PHP, pro externí přesměrování (*php.net/curl*)).

4.2 Stavový kód 404

V okamžiku kdy je požadovaný zdroj na web serveru nedostupný je zaslán stavový kód 404. Podobně jako u stavového kódu 200 je umožněno s ním zasílat HTML obsah (například nabídnout návštěvníkovi možnost přesměrování na domovskou stránku, případně poskytnout relevantní adresy).

Bez ohledu, zda je použita generická nebo dodatečně vytvořená HTML stránka, vyhledávače budou vědět, že stránka nebyla nalezena, což bývá důvod k odstranění z indexu.

Vyhledávače nikdy nebudou indexovat stránku se stavovým kódem 404.

U statických stránek je mechanismus zajištěn implicitně – stačí smazat příslušný soubor a v okamžiku, když se ho někdo pokusí načíst, bude zaslán stavový kód 404 automaticky.

Při práci s dynamickými stránkami jsou na kód kladeny dodatečné požadavky – pokud je smazán z databáze například nějaký produkt, jsou zpravidla zrušeny i všechny interní odkazy na něj. Nicméně je potřeba ošetřit i externí odkazy.

Nejhorší případ je vrátit prázdnou stránku se stavovým kódem 200 (což se typicky stane pokud byl produkt smazán bez dodatečného ošetření), což vede k duplicitnímu obsahu. Řešením může být buď přesměrování na jiný produkt nebo vygenerování stavového kódu 404.

Častým problémem, vyskytujících se u některých poskytovatelů webového prostoru, bývá generování stránky s obsahem „Stránka nenalezena“ avšak se stavovým kódem 200, což potencionálně vede k nekonečnému množství stránek s duplicitním obsahem.

4.3 Stavový kód 500

Stavový kód 500 indikuje interní problémy na straně serveru. Například v okamžiku, když spadne databázový server, bude drtivá většina stránek vracet stavový kód 404 resp. bude prázdná.

V každém případě je potřeba zabránit situaci, kdy vyhledávač vidí pouze bílé stránky, resp. je informován o neexistenci obsahu (stavový kód 404).

V takovýchto kritických situacích je výhodnější, pokud nevidí vyhledávač vůbec nic – bude se pravděpodobně domnívat, že je problém ve spojení a navštíví stránky později. V každém případě by měl být problém co nejdříve vyřešen.

V kritických situacích je nejvhodnější (pokud funguje web server) zasílat stavový kód 500 a slušně informovat o vzniklé situaci.

4.4 Další typy přesměrování

Ačkoliv máme k dispozici více způsobů přesměrování, například `meta refresh` a přesměrování JavaScriptem, není obecně doporučeno držet se těchto technik.

V minulosti mnohokrát došlo k jejich zneužití spammery a proto je jejich používání vždy podezřelé.

U `meta refresh` se výrazně nedoporučuje zpoždění menší než deset vteřin.

```
<meta http-equiv="refresh" content="10;url=http://address.cz/">
```

Příklad 4.2: Přesměrování pomocí `meta refresh`

Přesměrování Javascriptem není doporučeno vůbec, protože pokud je objeveno, vždy směřuje k postihům.

5 Duplicitní obsah

O duplicitním obsahu mluvíme v okamžiku, když stránka obsahuje stejný nebo velmi podobný obsah jako stránka adresována jiným URL.

Byl diskutován jako jeden z faktorů negativně se promítajících na ohodnocení stránek vyhledávači. Zmínka o něm padla v souvislosti s dynamickým obsahem stránek resp. při práci se stavovými kódy. V následující kapitole bude tento jev zkoumám podrobně.

Častou otázkou bývá, kolik duplicitního obsahu znamená příliš. Neexistuje žádné striktní pravidlo, avšak je vhodné eliminovat duplicitní obsah jak jen to je možné.

Obecně můžeme duplicitní obsah rozdělit do dvou skupin:

- způsobený architekturou aplikace
- jako důsledek krádeže obsahu

5.1 Duplicitní obsah jako důsledek architektury aplikace

Příklady běžných konceptů vedoucích k duplicitnímu obsahu:

- alternativní verze stránky pro tisk
- drobečková navigace
- stránky s významně podobným obsahem
- kontribuční stránky (*affiliate pages*)
- stránky s duplicitními titulky resp. s totožným obsahem meta elementů
- problémy související s kanonizací

5.1.1 Prevence duplicitního obsahu v rámci běžných konceptů

Alternativní verze stránky pro tisk

V současné době se tento problém příliš nevyskytuje, jedná se spíše o pozůstatek z dob, kdy jazyk CSS nebylo možno použít pro různé periferie.

Pokud je požadavek aby stránka obsahoval verzi pro tisk, je v současné době vhodné využít stylovacího jazyka CSS. Pokud je nějaký důvod CSS nepoužít, je namístě zakázat indexování tisknutelné verze (viz později).

Navigační odkazy a drobečková navigace

Jasná a intuitivní navigace je samozřejmostí každého solidního webu. Nicméně někdy může být zdrojem duplicitního obsahu.

Drobečková navigace (*Breadcrumb navigation*) se obvykle vyskytuje ve tvaru *hlavní stránka > kategorie > produkt,* kde každému takovémuto navigačnímu řetězci odpovídá jedna URL adresa (tedy například *obchod.cz/kategorie/produkt*). Uživatel má možnost se vrátit do libovolné sekce, která je logicky výš než ta, ve které se nachází.

Z hlediska SEO je vše naprosto v pořádku, dokud produkt nespadá do více kategorií. Jedné stránce je potom přiřazeno více adres, čímž je vytvořen duplicitní obsah.

Jako řešení je možné vytvořit jednu primární kategorii a ostatní zakázat indexovat. Případně obsah vlastní stránky dynamicky obměňovat.

Primární sekce v kombinaci se zákazem indexace

Jak název napovídá, pro každý objekt je vybrána jedna ze sekcí související s objektem a je označena za primární (což znamená, pokud chceme drobečkovou navigaci generovat automaticky, bude potřeba tuto informaci někde ukládat – nejlépe v databázi).Ostatní URL, kde je objekt v jiných sekcích, se zakáží indexovat (`robots.txt`, viz dále).

Bohužel řešení má i své stinné stránky:

1. Nižší ranking než by se dalo očekávat, pro produkt v kategorii jiné než primární.
2. Pokud je externě odkazováno na produkt v neprimární sekci, nemá to žádný vliv na ranking.

   Navíc je odkazování na neindexovanou stránku diskutabilní.

Dynamická obměna stránky

Tuto metodu je nutné praktikovat uváženě a opatrně, při špatném použití by stránka mohla být penalizována.

Použití spočívá v generování různých popisků pro stejné produkty v různých kategoriích. Dále pak v možnosti nabízení jiných produktů relevantních k dané kategorii.

Autoři literatury [1] mají dobré zkušenosti s první metodou.

Stránky s významně podobným obsahem

V okamžiku, kdy se vyskytuje více podobných (resp. stejných) produktů na více stránkách je vždy vhodné zvážit, zda by nebylo vhodné vše zahrnout na jednu stránku (například stejná trička různé barvy).

Stránky s duplicitními titulky resp. s totožným obsahem meta elementů

Jedná se pravděpodobně o nejčastější SEO prohřešek. Někteří odborníci tvrdí, že je lepší tyto elementy nevyplňovat, než aby byly všude duplicitní.

V každém případě je vhodné, aby každá stránka měla svůj unikátní popisek a pokud možno i obsah meta elementů, stránky tak budou vyhledávači lépe hodnoceny.

Problémy související s kanonizací

Www.obchod.cz vs. obchod.cz

Je vhodné si vybrat jednu adresu a na tu druhou vždy provést přesměrování:

```
RewriteCond %{HTTP_HOST} ^obchod\.cz
RewriteRule ^(.*)$ http://www.obchod.cz/$1 [R=301,L]
```

Příklad 5.1: Přesměrování z adresy obchod.cz na adresu www.obchod.cz (konfigurace pomocí .htaccess)

Index.html vs. /

Je vhodné si vybrat variantu bez indexu a z ní přesměrovávat na /.

```
RewriteCond %{THE_REQUEST} ^GET\ .*/index\.(html)\ HTTP
RewriteRule ^(.*)index\.(php|html)$ /$1 [R=301,L]
```

Příklad 5.2: Přesměruje z /index.html na / (konfigurace pomocí .htaccess)

SessionID využívající URL

Představují závažný problém pro vyhledávače, protože při každé návštěvě je vyhledávači předáno jiné URL. Stránky se pak jeví, že mají stejný obsah pod různou adresou.

V praxi proto bývá vhodné, vypnout předávání sessionID v URL. Pokud to není možné, využít přesměrování.

Kontribuční stránky (*Affiliate pages*)

Kontribuční stránky nabízejí produkt z jiného webu, přičemž z každého, díky nim prodaného kusu, profitují.

Každá kontribuční stránka obsahuje v odkazu na produkt proměnou, která ji jednoznačně určuje, například:

```
http://obchod.cz/produkt.html?affid=123
```

Příklad 5.3: Identifikace kontribuční stránky

Tedy pro stejný obsah existují různé adresy, můžeme opět mluvit o duplicitním obsahu. Problém lze odstranit buď na základě analýzy HTTP hlaviček, vyloučením přidružených odkazů z indexace nebo přesměrovávání na základě parametru identifikujícího podílníka.

Analýza HTTP hlaviček

První a nepříliš životaschopnou možností je analýza HTTP hlavičky `Referer`, nesoucí informaci odkud návštěvník přišel, s následným uložením do cookies a podobně.

V okamžiku, kdy má uživatel nainstalovaný nějaký bezpečnostní software, může dojít k blokování této informace. Problémy též vyvstávají pokud majitel kontribučních stránek chce produkt nabízet i na jiných stránkách, které vlastní.

Vyloučení přidružených odkazů z indexace

Lze také použít soubor `robots.txt` resp. `meta` vyloučení (viz dále), pro ignorování přidružených odkazů vyhledávači. Je ale učiněn předpoklad, že všechny kontribuční odkazy jsou dodatečně spravovány na jednom místě.

Další možností je umístit do podadresáře skript vypořádávající se s kontribučními odkazy a do `robots.txt` umístit následující kód:

```
User-agent: *
Disallow: /aff/
```

Příklad 5.4: Vyloučení přidružených stránek z indexace pomocí robots.txt

Přesměrovávání na základě parametru identifikujícího podílníka

Asi nejlepší variantou je, detekovat zda příchozí URL obsahuje podílnický parametr, pokud ano, tak ho uschovat (cookies) a následně přesměrovat na adresu bez tohoto parametru (stavový kód 301).

5.2 Duplicitní obsah jako důsledek krádeže obsahu

Krádež obsahu představuje odlišný problém. Vyhledávače se obecně snaží předcházet duplicitnímu obsahu, proto jsou stavěny před problém identifikace, který text je originální – autoritativní. Bohužel, občas mohou vyhledávače určit špatně.

Pro vyhledávání duplicitního obsahu je možno použít službu „Copyscape“ na adrese http://www.copyscape.com.

5.3 Odstranění duplicitního obsahu

Pokud se duplicitní obsah vyskytuje, jako jedno z možných řešení (avšak může být velice komplikované) je změna architektury aplikace. Někdy však postačí jedna, případně kombinace následujících metod:

- `meta robots` element
- soubor `robots.txt`

5.3.1 Využití „meta robots“ elementu pro blokování duplicitního obsahu

Tuto metodu můžeme použít v případě, když pracujeme s aplikací jejíž zdrojový kód máme k dispozici nebo pokud aplikace obsahuje velké množství komplexních dynamických URL.

Realizace se provede vložením následujícího řádku do hlavičky HTML dokumentu:

```
<meta name="robots" content="noindex, nofollow" />
```

Příklad 5.1: Zabránění indexování indexu pomocí „meta robots“ elementu

Za povšimnutí stojí hodnota `nofollow` atributu `content`. Tím se zajistí, že žádný z odkazů přítomných na stránce, nebude procházen.

Atribut `name` může být nahrazen identifikací konkrétního robota, například `googlebot` (Google), `msnbot` (Yahoo!), `Slurp` (MSN Search), `Teoma` (Ask), ...

Stinnou stránkou této metody je potřeba načtení celé stránky, což teoreticky zpomaluje proces indexace. Takto lze spravovat pouze HTML soubory, k jejichž editaci máme oprávnění.

5.3.2 Soubor robots.txt

Soubor `robots.txt` bývá standardně umístěn v kořenovém adresáři aplikace. Není vhodné mít více takovýchto souborů, vyskytujících se kdekoliv jinde.

Kompletní reference včetně přesných jmen všech dostupných robotů je k dispozici v oficiální dokumentaci, viz [9].

Definice jednotlivých zdrojů, které se nemají indexovat, se provádí na základě pravidel. Pravidla umožňují označit celý adresář nebo všechny soubory začínající konkrétním znakem.

Občas se stane, že `robots.txt` je považován i za metodu zajišťující bezpečnost. Zde je na místě připomenout jedno ze základních pravidel bezpečnosti – „bezpečnost zajištěna neznalostí nefunguje“, tedy to, že citlivá data nejsou indexována vyhledávači, neznamená nic z hlediska jejich zabezpečení. Robots.txt má následující strukturu:

```
# komentář
 User-agent: *
 Disallow: /
```

Příklad 5.2: Struktura `robots.txt`

Na začátku každého souboru `robots.txt` by měl být definován vyhledávač (user-agent), hvězdička zahrnuje všechny. Následuje výčet zdrojů které nemají být indexovány. Jsou uvozeny klíčovým slovem `disallow`, přičemž symbol `/` definuje adresář. Komentář začíná symbolem `#`. Výše uvedený příklad zabrání indexaci všech souborů.

Obsah souboru by neměl být příliš rozsáhlý dle dostupných informací (viz [1]) by neměl například pro Google přesahovat 5000 znaků.

```
User-agent: *
Disallow: /adresar
Disallow: soubor.html

User-agent: googlebot
Disallow: /adresar
Disallow: soubor.html

Disallow: /adresar2/
Disallow: /*print-
```

Příklad 5.3: Složitější příklad `robots.txt`

V příkladu 5.3 je nejprve pro všechny vyhledávače zakázáno indexovat adresář `adresar` a všechny soubory, co začínají slovem `adresar` a `soubor.html`.

Následují pravidla pro vyhledávač Google, přičemž je požadavek, aby neindexoval to samé jako ostatní vyhledávače – což je potřeba znovu zapsat (aktuálně není jiné cesty). Dále pak definuje nová pravidla: První z nich zakazuje indexaci celého adresáře `adresar2` a poslední všechny soubory obsahující jako podřetězec slovo `print`.

Použitá hvězdička v pravidle je nestandardní, přičemž v současné době má podporu pouze u vyhledávačů Google, Yahoo! a MSN, proto je vhodné ji používat jen u pravidel přiřazených těmto vyhledávačům.

5.3.3 Co raději neindexovat

- stránku s nákupním košíkem
- administrátorskou přihlašovací stránku
- administrační stránky

6 Black Hat SEO

Jedná se o souhrn nekalých praktik vedoucích k neoprávněnému nabytí vlastnictví (přivlastnění obsahu jiných stránek) resp. deklasování obsahu stránek cizích. Mohou však být ve výjimečných případech i užitečné.

Znalost těchto technik je předpoklad k obraně před nimi.

6.1 Zahalování

Zahalování (*cloaking*) je technika zajišťující zasílání různého obsahu vyhledávačům a lidskému uživateli.

Tuto techniku je možno s výhodou použít i v případě, pokud je stránka v nečitelném formátu pro vyhledávače (například kompletně ve Flashi).Avšak jedná se spíše o krajní řešení.

6.2 Automatizované spam útoky

Obvyklým terčem útočníků bývají blogy, fóra apod. Jedním z jejich cílů je zanechat zde odkazy na jejich weby. Nezřídka tento proces bývá automatizován, proto vznikly různé obranné mechanismy založené na Turingově rozpoznávacím testu (CAPTCHA - *Completely Automated Public Turing test to tell Computers and Humans Apart*). Bohužel při jejich použití dojde k porušení přístupnosti – takováto stránka je nepřístupná pro nevidomé.

Útočníci však drží krok s dobou a zpravidla se jim tyto mechanismy daří prolomit (čím masově rozšířenější CAPTCHA, tím spíše je nebo bude prolomena).

Řešením je pravidelně a často aktualizovat, resp. začleňovat vlastní řešení, případně jejich kombinace.

Jedna z možných CAPTCHA realizací dle [1]. Při odesílání dat je pro uživatele vygenerován jednoduchý matematický příklad, který je převeden na slova. Je tedy například dotázán *kolik je dvacet minus pět?* Přičemž, s otázkou je k němu zaslána i zahašovaná odpověď. V okamžiku kdy jsou data na serveru přijímána, porovnají se příslušné haše.

6.3 Útok *HTML insertion*

Vstupní data ze všech nedůvěryhodných zdrojů by měla být z bezpečnostních důvodů – například jako prevence před útokem *SQL injection* – vždy předzpracována (*escape process*). Podobně tak i při pozdějším výpisu by měly být HTML znaky převedeny na ekvivalentní HTML entity.

Z hlediska Black Hat SEO nás zajímá především proces převedení na entity. Krom toho, že takto předcházíme XSS útoku (Cross-site scripting), zabráníme tak i vkládání (spam) odkazů.

Může se stát, že chceme návštěvníkovi umožnit, aby vložil odkaz na svoje stránky, ale nechceme, aby vyhledávače tuto skutečnost reflektovaly (což zpravidla odradí velkou část spamerů). Řešením je nestandardní hodnota `nofollow` atributu `rel` elementu `a` (zavedl Google).

```
<a rel="nofollow" href="http://spamer.cz">Nežádoucí odkaz</a>
```

Příklad 6.1: Odkaz zpravidla ignorovaný vyhledávači

6.4 Útok *301-redirect*

Pokud architektura aplikace využívá skript pro přesměrování na jinou stránku na základě parametru v URL, měla by být ošetřena situace, kdy je do parametru podstrčena nežádoucí (spam) adresa.

Mějme například adresu *mojeadresa.cz/redirect.php?adr=http://cilpresmerovani.cz*, která zajistí přesměrování na web uvedený v parametru `adr`,a to včetně zaslání stavového kódu 301. V tomto okamžiku může útočník rozmístit na internetu adresy typu *mojeadresa.cz/redirect.php?adr=http://spamer.cz*, čímž nekale profituje na našich stránkách.

Možná řešení:

- namísto stavového kódu 301 použít 302
- v `robots.txt` vyjmout `redirect.php` z indexace
- řešení s využitím databáze – přesměrování bude probíhat jen na důvěrné adresy

6.5 Obchodování s odkazy

Jak bylo řečeno dříve, zpětné odkazy mají velký podíl na viditelnost webu ve vyhledávačích. Proto není divu, že se s nimi začalo obchodovat.

Dle [1] pokud je detekována tato skutečnost (alespoň v Google kontextu), za nežádoucí bude označena zejména v případě, pokud se jedná o tématicky nerelevantní stránky.

7 Geo-cílení

7.1 Úvod

Geo-cílení (*Geo-targeting)* je technika zaměřená na optimalizaci webu pro určité geografické regiony – státy a podobně. Ve své podstatě není nepodobná zahalování (viz předchozí kapitola). Nicméně geo-cílení se snaží poskytnout různé obsahy jak uživateli, tak vyhledávači v závislosti na původu požadavku z geografického hlediska. Nutno podotknout, že je tento postup mnohem méně kontroverznější než diskutované zahalování a je respektován napříč všemi dostupnými vyhledávači.

7.2 Implementace

Geo-cílení, stejně jako zahalování jsou zpravidla implementovány s využitím metody *IP-delivery.* Na vstupu je síťová adresa návštěvníka, která slouží jako zdroj pro identifikaci geo-původu.

V případě geo-cílení jsou do databáze ukládány různé rozsahy IP adres a informace o jejich geografickém původu. Skript na základě příchozí adresy a údajích v databází vrátí relevantní obsah.

7.3 Tipy

Vyhledávač může použít aktuální fyzickou lokaci navštíveného web serveru pro identifikaci cíleného trhu, proto je obecně doporučeno přesměrovávat na server umístěný v daném regionu, než pouze poskytnout modifikovaný obsah.

S geo-cílením souvisí problematika cizojazyčných webů, viz kapitola 8.

8 Cizojazyčné weby

Není sporu o tom, že internet pomohl zglobalizovat celosvětový obchod. Nadneseně řečeno, kdokoliv odkudkoliv si může koupit cokoliv. Avšak je docela pravděpodobné, že uživatel z Čech si bude chtít koupit konkrétní věc na českém webu, Švéd na švédském webu a podobně.

Prvním z předpokladů úspěšné optimalizace je důkladné seznámení s jazykovými specifiky cílového trhu. Pokud to je možné, využít externích specialistů pro překlad.

8.1 Úpravy HTML kódu

U cizojazyčných webů je vhodné jednotlivé jazykové mutace identifikovat pomocí atributu `lang` u elementů `div` resp. `span`.; resp. v záhlaví stránky pro definici jazyka celé stránky.

Čeština je identifikována hodnotou `cs`, například australská angličtina `en-AU`. Kompletní přehled hodnot viz [15].

```
Identifikace '<span lang="cs">obsah</span>' jazykového kontextu obsahu.
```

Příklad 8.1. Identifikace jazykové mutace v text

```
<meta lang="en-AU">
```

Příklad 8.2. Identifikace jazykové mutace stránky (umístěno v hlavičce html dokumentu)

8.2 Lokace serveru a jméno domény

Vyhledávače občas používají aktuální geografickou pozici serveru pro identifikaci cíleného trhu.
Je proto dobré, aby server byl umístěn ve stejné lokaci jako je cílený region. Dále pak kód identifikující zemi (ccTLD - *country code top-level domains*) by měl splývat s cílovým trhem – nicméně toto se striktně netýká koncovek jako .com nebo .net a podobně.

8.3 Fyzická adresa společnosti v cizí oblasti

Obecně je známo, že vyhledávače pro identifikaci cílení analyzují zda je na stránkách přítomna fyzická adresa společnosti. Je proto vhodné ji vkládat do patičky každé stránky. A pokud to je možné, každá geo oblast by měla obsahovat svou konkrétní adresu.

8.4 Diakritika

Vedle češtiny existuje více jazyků obsahujících diakritiku. Některé vyhledávače, včetně Google, reflektují tuto skutečnost. V indexu jsou uchovávána slova s diakritikou i bez ní.

9 Metodika vytváření stránek optimálních z hlediska SEO

Tato kapitola obsahuje návrh metodiky usnadňující tvorbu optimalizovaných stránek. Framework jsem pojmenoval *Johnny*. Důvodem je snadná zapamatovatelnost a zároveň narážka na mé křestní jméno.

Framework bude určen pro tvorbu tzv. *CRUD* aplikací (*Create, Retrieve, Update, Delete* – společně s další operací *List* budou tvořit soubor základních abstraktních operací), tedy systémů, kde každá entita je spravovatelná. Framework rozlišuje několik typů entit, viz dále.

9.1 Návrh

Framework bude vytvářen v několik iteracích, přičemž první bude realizována v rámci této diplomové práce. Iterace jsou celkem tři:

- Iterace 1,
- iterace 2 a
- iterace α.

9.2 Iterace 1

Realizuje následujíccí výčet bodů:

- Správa entit,
- efektivní generování obsahu elementu `<title></title>`,
- podpora hezkých URL,
- generování relevantních HTTP stavových kódů pro příchozí požadavky a
- podpora více jazykových mutací.

9.2.1 Správa entit

Nakládání s entitou se bude lišit v závislosti na jejím typu. Entita může být:

- Standardní,
- adresovaná přes hezká URL,
- vícejazyčná a
- vícejazyčná s hezkými URL.

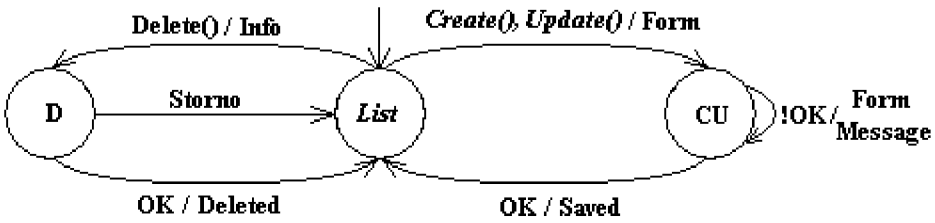
Tabulka uvádí jednotlivé příklady pro různé typy entit.

Tabulka 29: Typy entit a příklady použití

Typ entity	Příklad použití
standardní	entita *Koncert*, v jednojazyčné verzi webu nějaké kapely, obvykle není potřeba detail takovýchto entity
adresovaná přes hezká URL	entita *Novinka*, která pro kliknutí na titulek poskytne další informace
vícejazyčná	jakákoliv entita,s daty ve více jazykových mutacích
vícejazyčná s hezkými URL	kombinace předchozích dvou

Všechny typy entit musí umožňovat základní operace. V praxi budou všechny typy realizovány třídami, od nichž budou dědit konkrétní třídy (viz příklad jednoduchého modelu v příloze).

Následují obrázek obsahuje schéma reprezentující základní životní schéma aplikace:



Obrázek 3: Základní životní schéma aplikace

Jako výchozí operace je vždy použita *List* (vypíše seznam všech (resp. určitý počet) entit daného typu). Následně jak pak možné nad nimi provést další libovolnou základní operaci.

Operace *Create* a *Update* vytvoří relevantní formulář pro přidání resp. upravení entity a testují správnost zadaných dat. Pokud nejsou data v pořádku je formulář opět zobrazen a to včetně chybové hlášky.

Pokud je zavolána operace *Delete* dojde k vypsání potvrzovacího dialogu obsahující přehledové informace o právě odstraňovaných datech (ve všech jazykových mutacích) relevantních danému id (viz generování URL). Proces odstraňování lze v rámci dialogu zrušit – kliknutím na tlačítko storno.

Realizace databázové úrovně

V závislosti typu entitu mají databázové tabulky různě předdefinovanou strukturu. Framework reflektuje dva základní požadavky na entitu:

- Adresovatelnost přes URL (entita odvozena od `Johnny_Business_WithNiceUrl`) a
- možnost ukládat její obsah v různých jazycích (`Johnny_Business_International`).

Struktura tabulek pro entity typu `Johnny_Business_WithNiceUrl`

Je zapotřebí ukládat informaci jak bude vypadat klíčové URL. Dále, pokud je URL změněno, musí být provedeno vhodné přesměrování a vygenerovat příslušný stavový HTTP kód.

Tabulka 30: Databázová tabulka entit s pěkným URL

id	sloupec1	…	sloupecN	url	redirect	status_code

Nově vložená entita má hodnoty sloupců *url* a *redirect* nastaveny na NULL. Pohled poskytující přehled všech entit v dané tabulce nedodává řádky, kde je *redirect* různé od NULL. Naopak pohled ve kterém je konkrétní entita (resp. její detail) vyhledávána tyto řádky obsahuje (více viz kapitola o pohledech).

Hodnota sloupce *url* je vygenerována na aplikační úrovní filtrem `Johnny_Filter_NiceUrl`, a vychází s předem určené hodnoty v rámci business třídy (viz metoda `setUrlCol(urlCol)` rozhraní `Johnny_Business_WithNiceUrl_Interface` v UML diagramech v příloze).

Struktura tabulek pro entity typu `Johnny_Business_International`

Pokud entita obsahuje data v různých jazykových mutacích, potom jsou jazykově závislé položky ukládány do dodatečné tabulky svázané s původní tabulkou.

Tabulka 31: Jazykově nezávislá tabulka (entity)

id	sloupec1	sloupecN

Tabulka 32: Jazykově závislá tabulka (entity_international)

id	entity_id	sloupec1	sloupecN	lang

Sloupec *entity_id* obsahuje cizí klíč do první tabulky. Sloupec *lang* určuje jazyk záznamu v řádku.

Pohledy

Pro uniformní přístup z aplikační úrovně k datům jsou zavedeny databázové pohledy. Je několik pohledů na entitu a každý je využit v jiné situaci. Jméno každého typu pohledu je ve standardním tvaru.

Tabulka 33: Typy pohledů a jejich využití

Pohled	Využití
v_entita_{lang}	detailní náhledy, informace před odstraněním
v_entita_list_{lang}	seznamy
v_entita_raw_{lang}	data pro formuláře

Koncovka *lang* identifikuje jazykovou mutaci pohledu, pokud to je vyžadováno. Příklad je uveden v případové studii.

9.2.2 Efektivní generování obsahu elementu `<title></title>`

Entita se bude vždy nacházet v jednom ze stavů, který je daný jednou ze základních operací. Z toho vyplývá potřeba nejméně pěti různých vzorů titulků pro entitu.

Následující tabulka přiřazuje jednotlivým operacím titulkové vzory.

Tabulka 34: Titulkové vzory

Základní operace	Titulkový vzor	Příklad
Create	Přidej {entity} | {home}	Přidej článek | redblack.cz
Retrieve	{id} | {home}	Sad Harmony| redblack.cz
Update	Uprav {entity} | {home}	Uprav článek | redblack.cz
Delete	Smaž {entity} | {home}	Smaž článek | redblack.cz
List	{entity}$_P$ | {home}	články | redblack.cz

Symbol {entity} zastupuje textový popis obecné entity (například *kapela*), {id} identifikátor konkrétní entity (např. *Sad Harmony*) a {home} domovskou stránku celého webu (např. redblack.cz). Pokud je symbolu přiřazen dolní index P (plurar), jedná se potom o množné číslo, implicitně je použito jednotné.

9.2.3 Podpora hezkých URL

Hezké URL bude odvozeno od aktuálně prováděné základní operace. Následující tabulka poskytuje tvar URL pro danou základní operaci.

Tabulka 35: URL vzory

Základní operace	URL vzor	Příklad
Create	/{entity}/add	/news/add
Retrieve	/{entity}/detail/{id}	/news/detail/reporty-z-tour
Update	/{entity}/edit/id/{id}	/news/edit/id/2
Delete	/{entity}/delete/id/{id}	/news/delete/id/2
List	/{entity}	/cz/news

Za povšimnutí stojí hodnota {id} u operací *Update* a *Delete* – jedná se o primární klíčové hodnoty na úrovní databáze.

Správa entit tohoto typu s pěknými URL se vyznačuje následujícími specifiky:

- URL je zapotřebí ukládat zvlášť.
- Je zapotřebí definovat vlastnost, ze které se bude odvozovat URL.
- Pokud dojde ke změně hodnoty odvozovací vlastnosti, musí být změněno i URL.
- Při změně URL, automaticky generovat přesměrování z původního URL na nové (a zároveň generovat HTTP stavový kód 301).

9.2.4 Generování relevantních HTTP stavových kódů pro příchozí požadavky

Je nezbytné generovat odpovídající stavové HTTP kódy, pokud dojde k nestandardně splnitelnému požadavku. Následující tabulka obsahuje přehled použitých kódů.

Tabulka 36: Generované HTTP stavové kódy

Http kód	Kdy generován
404	pro všechna nevyhovující URL
301	přesměrovávání, (například pokud dojde ke změně hezkého URL konkrétní entity)

9.2.5 Podpora více jazykových mutací

Framework bude podporovat uchovávání konkrétní entity ve více jazykových mutacích. Entity vícejazyčného typu podléhají následujícím kritériím:

- Vlastnosti entity se dělí na jazykově závislé a jazykově nezávislé.
- Po vytvoření nové entity budou ve všech jazykových variantách stránky zobrazena jazykově nezávislá data.
- Jazykově závislá data budou budou ukládána podle globální aktuální jazykové varianty stránky.
- Pokud bude smazána záznam jedné jazykové varianty budou smazány i všechny ostatní záznamy s touto mutací spojené.

9.3 Iterace 2

Realizuje následující body:

- Geo-targeting,
- zahalování,
- automatické generování mapy webu,
- podpora pro drobečkovou navigaci,
- atd..

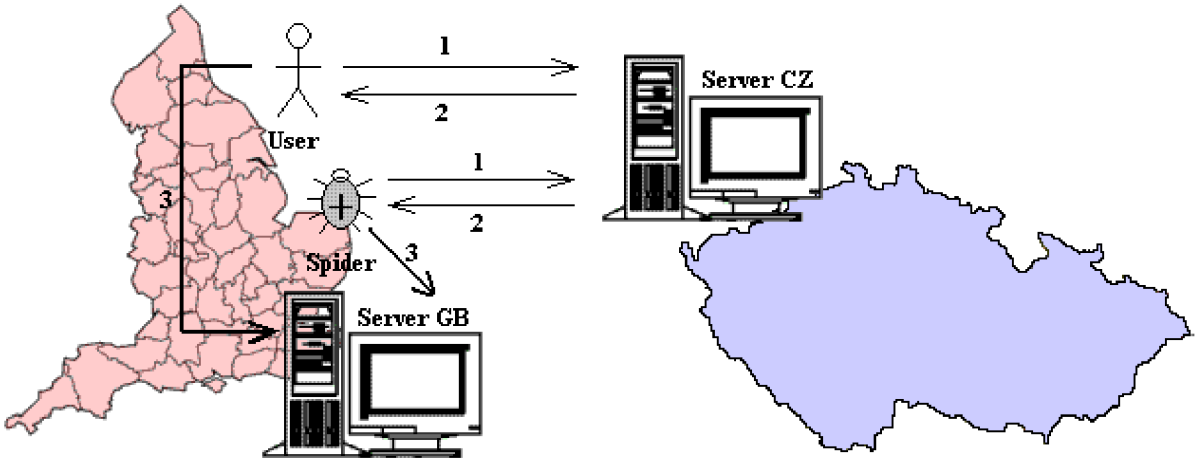
9.3.1 Geo-targeting

Realizace této problematiky využívá techniku *IP delivery*, kde je na základě síťové adresy identifikováno z jaké geografické oblasti požadavek přichází.

Podařilo se mi nalézt web poskytující službu, která identifikuje geografickou oblast požadavku tedy identifikace státu, města a souřadnic (momentálně nás zajímá 1.možnost) a zároveň je zdarma.

Výstup je v HTML nebo XML. Více na adrese www.hostip.info. Ceny komerčních řešení se pohybují okolo 50$.

Následující obrázek ilustruje koncept geo-targetingu. Je zde vidět že vyhledávač i obyčejný uživatel dostanou **stejnou** odpověď v závislosti na svojí poloze.



Obrázek 4: Geo-targeting

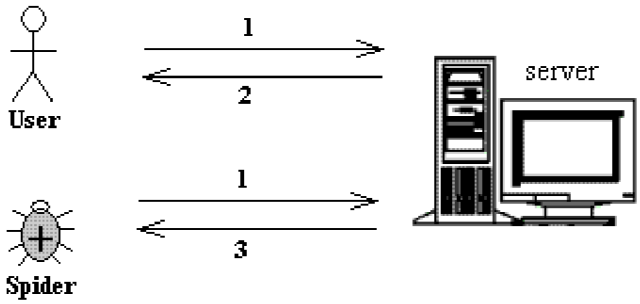
Cesta 1 je požadavek z Velké Británie na server v České republice. Dojde však k identifikaci zahraniční žádosti a je vráceno přesměrování (HTTP stavový kód 301) na server ve Velké Británii (cesta 2). Klient navštíví server ve své geografické oblasti (cesta 3)

9.3.2 Zahalování

Stejně jako zahalování, u kterého je taktéž využita technika *IP delivery*, avšak s tím rozdílem, že zde jsou sledovány konkrétní IP adresy a ne jejich rozsahy.

Existuje možnost sledovat uživatelského agenta požadavku (HTTP hlavička *User-Agent*) a na jeho základě vyhledávač identifikovat. Nejedná se však o spolehlivou a doporučeníhodnou metodu, protože může dojít k modifikaci této hlavičky a tedy ke snadnému odhalení zahalování (to může být nepříjemné například v případě, že tuto skutečnost odhalí konkurence. Jak bylo uvedeno dříve, jedná se o velmi kontroverzní techniku, a proto je nejlepší, aby byla dobře skrytá).

Seznam síťových adres vyhledávačů je k dispozici například na adrese iplists.com. Následující obrázek ilustruje koncept zahalování:



Obrázek 5: Zahalování

Uživateli a vyhledávači jsou zaslány na stejnou žádost různé odpovědi.

9.3.3 Automatické generování mapy webu

Pro lepší pochopení struktury webu, by měla být na každém takovém netriviálním přítomna jeho mapa. Framework bude nabízet automatické generování mapy webu na základě struktury menu.

9.3.4 Podpora pro drobečkovou navigaci

Framework bude implementovat koncept nastíněný v kapitole 5, konkrétně variantu: *Primární sekce v kombinaci se zákazem indexace.*

9.4 Iterace α

Iterace koexistující s předcházejícími. Jejím cílem je vybudovat programové nástroje (platformu) automatizující tvorbu webových aplikací využívajících Johnny Framework. Následuje výčet rysů, přičemž v rámci této práce je realizován první bod.

- Na základě abstraktního popisu aplikace (v UML – konkrétně diagramu tříd) generovat kompletní busines třídy (návrhový vzor *Model-View-Controller*). Nástroj se bude jmenovat `ClassDiagram_CodeGenerator_Php`.
- Na základě abstraktního popisu aplikace (diagramu případu použití) generovat kontroléry pro jednotlivé entity.
- Na základě abstraktního popisu aplikace (diagramu tříd) generovat databázové tabulky a jejich ORM třídy.

- Na základě abstraktního popisu aplikace (diagramu případu použití) generovat *Role*, *Zdroje* a *ACL*.

Jedná se o základ webové platformy diskutované v [18].

9.4.1 ClassDiagram_CodeGenerator_Php

Výchozím bodem tvorby aplikace bude její model. Postup převodu na kód se skládá z následující kroků:

- Vytvoření modelu aplikace pomocí vhodného modelovacího jazyka,
- převod modelu do strojově přívětivé podoby,
- vytvoření interního modelu a
- generování kódu v jazyku cílové platformy.

9.4.2 Vytvoření modelu aplikace pomocí vhodného modelovacího jazyka

Reálný projekt obsahuje mnoho entit, proto jsem shledal za vhodné použít grafického modelovacího jazyka, který zprostředkovává vyšší perspektivu na daný problém. Konkrétně jsem vybral jazyk UML. Využity jsou modelovací prostředky diagramu případu použití a diagramu tříd. Osobně mám relativně uspokojivé zkušenosti s modelovacím nástrojem *ArgoUML* (argouml.tigris.org).

9.4.3 Převod modelu do strojově přívětivé podoby

Pro textovou reprezentaci modelu jsem zvolil jazyk XMI (*XML Metadata Interchange*). Jedná se o popis UML modelu v jazyce XML. Výše uvedený software *ArgoUML* podporuje export do tohoto formátu.

9.4.4 Vytvoření interního modelu

Interní model reprezentující diagram případu použití je tvořen následující strukturou:

```
$ucDiagrModel[entityId] = array( roleId => operationId )
```

Kód 9.1: Interní reprezentace diagramu případu použití

Na základě této části jsou nad jednotlivými entitami tvořeny kontroléry (viz návrhový vzor Model-View-Controller), seznam oprávnění (*ACL*) a vyplývá z něj také URL logika (např.: /entita/operace/id/hodnota).

Interní model reprezentující diagram tříd je tvořen následující strukturou:

```
$classDiagramModel[packageName][className] = array (
      "modifiers" => array( … ),
      "doc" => array( … ),
      "parent" => array( … ),
      "relationWith" => array( … ),
      "interfaces" => array( … ),
      "attribs" => array( … ),
      "methods" => array( … )
)
```

Kód 9.2: Interní reprezentace diagramu tříd.

9.4.5 Generování kódu v jazyku cílové platformy

Aktuální cílová platforma je PHP s využitím *Zend Frameworku*. Členění vygenerovaných souborů odpovídá *konvenční modulární adresářové struktuře*, viz [21].Vlastní kód je v souladu se *Zend kódovými standardy*, viz [20].

10 Závěr

Cílem této práce bylo přinést základní informace o SEO problematice. Byly zmíněny všechny zásadní faktory ovlivňující viditelnost stránek na internetu, práce se dotkla i oblastí jako geografické resp. jazykové optimalizace.

Na základě popsaných činitelů byl navrhnut a implementován framework usnadňující tvorbu optimalizovaných resp. správně napsaných stránek. Důraz je kladen na unikátnost titulků každé stránky, správu záznamů s podporou přesměrovávání a generování příslušných HTTP stavových kódů.

Framework byl prezentován na případové studiu stránek Brněnského vydavatelství Redblack, dostupných na adrese www.redblack.cz. (testovací verze je na adrese redblack.jan-stefl.eu, viz přílohy)

Současné verze však obsahuje několik problémů, jejichž oprava bude předmětem dalšího vývoje. Patří sem nerelevantní reakce na chybně zadaná data do formuláře. Pokud je zadán špatný vstup, uživatel o tom není informován, musí se dodatečně přesvědčit, zda záznam byl skutečně přidán. Problém se také vyskytne, pokud je změněno URL adresující entitu (třídy odvozené od Johnny_Business_WithNiceUrl, Johnny_Business_InternationalWithNiceUrl) a následně pak přejmenováno tak, aby bylo generováno URL v prvotní podobě.

Další rozšíření budou zaměřena na automatické generování mapy webu, drobečkovou navigaci, podporu zahalování nebo integraci konceptu CAPTCHA.

Součástí práce bylo i vyhodnocení postavení případové studie v SERPs (viz přílohy) různých vyhledávačů. Z něho vyplývá, že reakční doba obnovení mezipamětí může být několik dnů (možná i týdnů). Dále byl učiněn předpoklad, že vše podstatné by mělo být dostupné z domácí stránky webu.

Souběžně s touto prací vznikl projekt, který si klade za cíl transformovat UML model (resp. jeho XMI podobu) na webovou aplikaci, podrobnější informace viz [18]. Za zmínku stojí, že využívá Johnny frameworku.

Literatura

[1] Jaimie Sirovich, Cristian Darie. *Professional Search Engine Optimization with PHP*. Indianapolis, Wiley publishing 2007.

[2] Rand Fishkin, CEO & Jeff Pollardm. *Google Search Engine Ranking Factors V2*, 2. dubna 2007. Dokument dostupný na URL http://www.seomoz.org/article/search-ranking-factors (říjen 2007).

[3] Společnost Google. *Vytěžte ze svého obsahu co nejvíce.* Doprovodná brožura k přednášce „Secret of search“ od Douglase Merilla 2007.

[4] Rand Fishkin, CEO & Jeff Pollardm. *Google Search Engine Ranking Factors V2*, 2. dubna 2007. Dokument dostupný na URL http://www.seomoz.org/article/search-ranking-factors (říjen 2007).

[5] *Webmaster Guidelines*. Dokument dostupný na URL http://www.google.com/support/webmasters/bin/answer.py?answer=35769 (prosinec 2007)

[6] Matt Cutts. *More SEO Answers on Video*. Dostupné na URL http://www.mattcutts.com/blog/more-seo-answers-on-video/ (prosinec 2007)

[7] *Google bomb*. Dokument dostupný na URL http://en.wikipedia.org/wiki/Google_bomb (prosinec 2007).

[8] *List of HTTP status codes.* Dokument dostupný na URL http://en.wikipedia.org/wiki/List_of_HTTP_status_codes (prosinec 2007).

[9] *About /robots.txt*. http://www.robotstxt.org/robotstxt.html (prosinec 2007).

[10] Matt Cutts, Maile Ohye, *Information about buying and selling links that pass PageRank*, 1. prosinec 2007. Dokument dostupný na URL http://googlewebmastercentral.blogspot.com/2007/12/information-about-buying-and-selling.html (prosinec 2007).

[11] *Search engine optimization.* http://en.wikipedia.org/wiki/Search_engine_optimisation (prosinec 2007).

[12] *The Truth About Internet Marketing*. Dostupné na URL http://groups.google.com/group/alt.current-events.net-abuse.spam/browse_thread/thread/6fee2777dc17b8ab/3858bff94e56aff3?lnk=st&q=%22search+engine+optimization%22&rnum=1 (prosinec 2007).

[13] Aliweb - Dostupné na URL http://en.wikipedia.org/wiki/Aliweb (prosinec 2007).

[14] PageRank - Dostupné na URL http://en.wikipedia.org/wiki/PageRank (prosinec 2007).

[15] *Using Language Identifiers (RFC 3066)* – dostupné na URL http://www.i18nguy.com/unicode/language-identifiers.html (prosinec 2007).

[16] *Programmer's Reference Guide* – Dostupné na URL http://framework.zend.com/manual/en (prosinec 2007).

[17] *Model-view-controller* - Dostupné na URL http://en.wikipedia.org/wiki/Model-view-controller (prosinec 2007).

[18] Jan Štefl. *Johnny – web platform for creating internet applications. Proceedings of the 14th Conference, STUDENT EEICT 2008, Volume 2,s. 185.* Vysoké učení technické v Brně, 2008.

[19] Sebastian Bergmann, *PHPUnit Pocket Guide,* poslední modifikace květen 2008. Dokument dostupný na URL http://www.phpunit.de/pocket_guide/3.2/en/ (květen 2008).

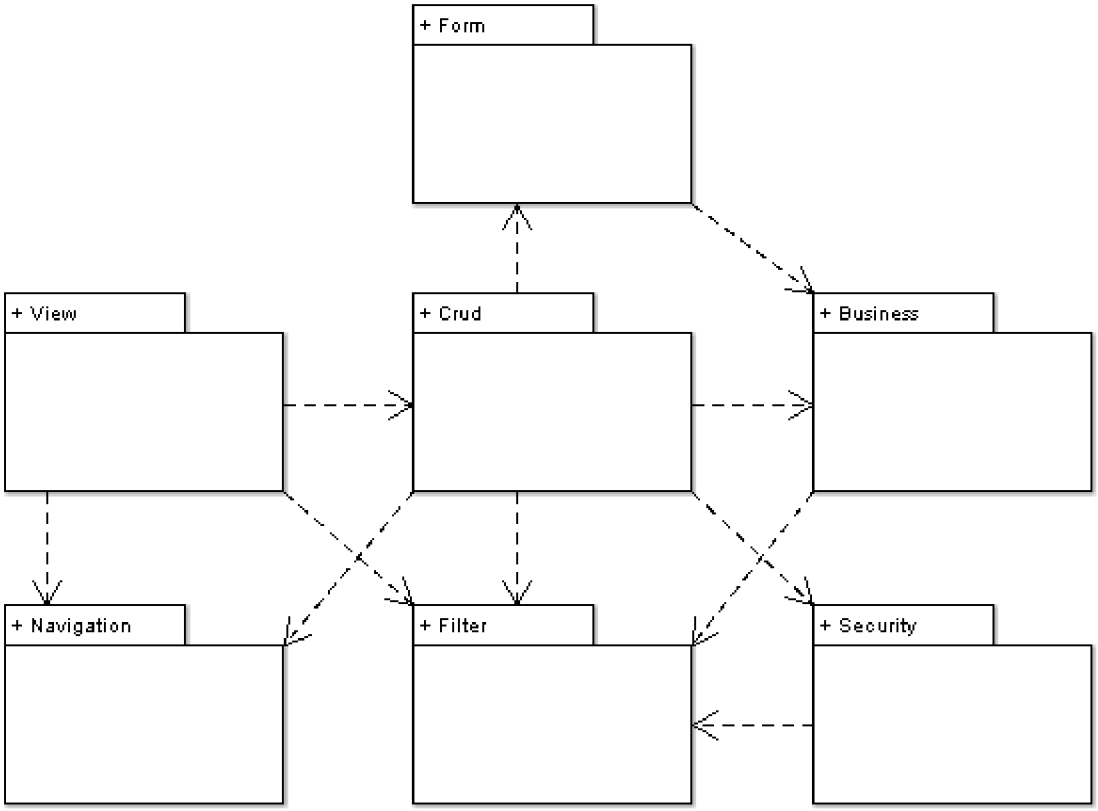
[20] Zend Technologies. Zend Framework PHP Coding Standard. Dokument dostupný na URL http://framework.zend.com/manual/en/coding-standard.html (květen 2008).

[21] Zend Technologies. Using a Conventional Modular Directory Structure. Dokument dostupný na URL http://framework.zend.com/manual/en/zend.controller.modular.html. (květen 2008).

Seznam příloh

A) Johnny Framework

Struktura Frameworku je rozčleněna do jednotlivých koncepčních balíčků. Každý balíček zastřešuje realizaci určitých oblastí. Balíčky jsou na sebe vzájemně závislé, viz následující obrázek:



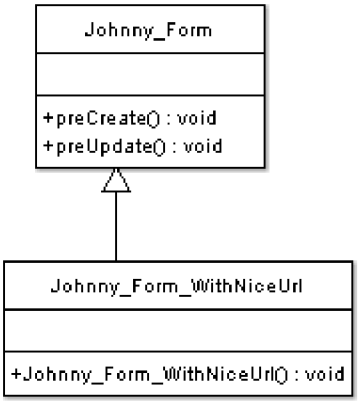
Obrázek 6: Diagram balíčků

Vlastní realizace bude provedena v jazyku PHP s využitím Zend frameworku (viz [16]). Závislosti na něm jsem z důvodu přehlednosti nekreslil, nicméně všude, kde je potřeba, jsou zmíněny. Dále je uveden popis všech balíčků se zachycením jejich obsahu v UML. Následuje popis případové studie využívající Johnny framework.

A.1 Popis balíčků

A.1.1 Form

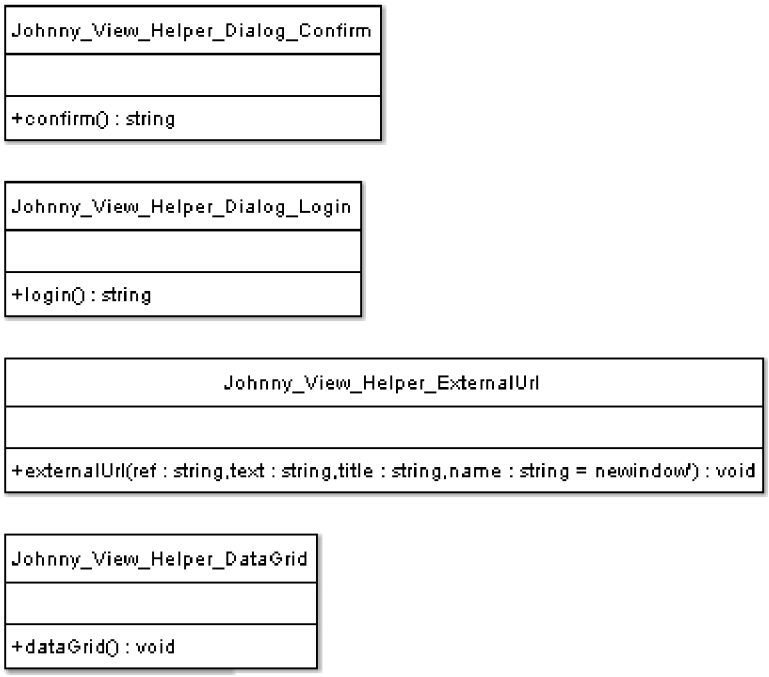
Abstrakce HTML formulářů. Dědí od třídy `Zend_Form`.



Obrázek 7: Balíček Form

A.1.2 View

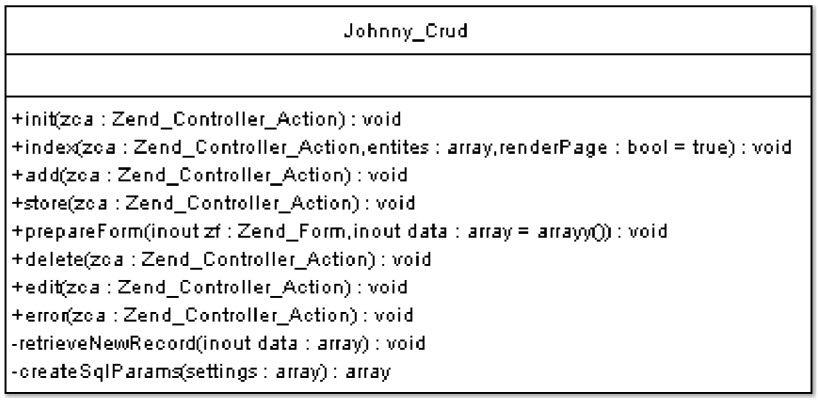
Jedna z částí návrhového vzoru Model-View-Controller (viz [17]).Vychází z implementace třídy `Zend_View`.



Obrázek 8: Balíček View

A.1.3 Crud

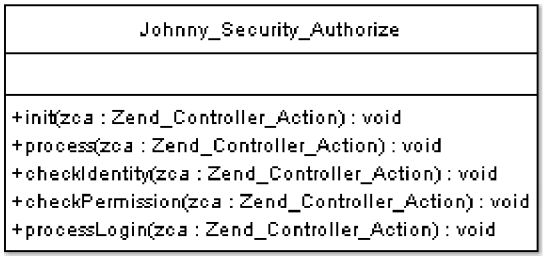
CRUD tedy „*Create*“ „*Retrieve*“ „*Update*“ „*Delete*“. Abstrahuje práci části Controller (Model-View-Controller (viz [17])). Jedná se o nižší vrstvu (blíže k databázi) reagující na požadavky uživatele.

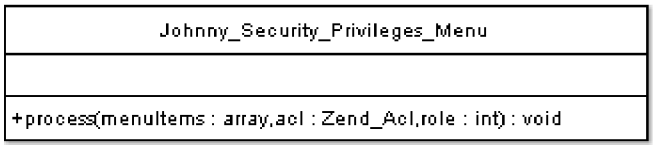


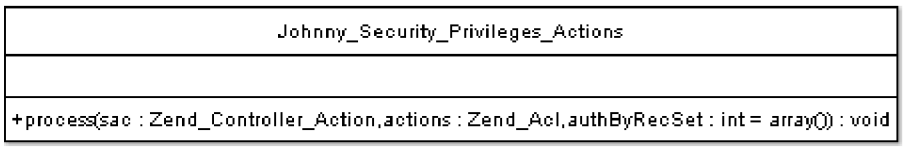
Obrázek 9: Balíček Crud

A.1.4 Security

Podpora pro autentizaci a autorizaci. Využívá Zend Frameworku.

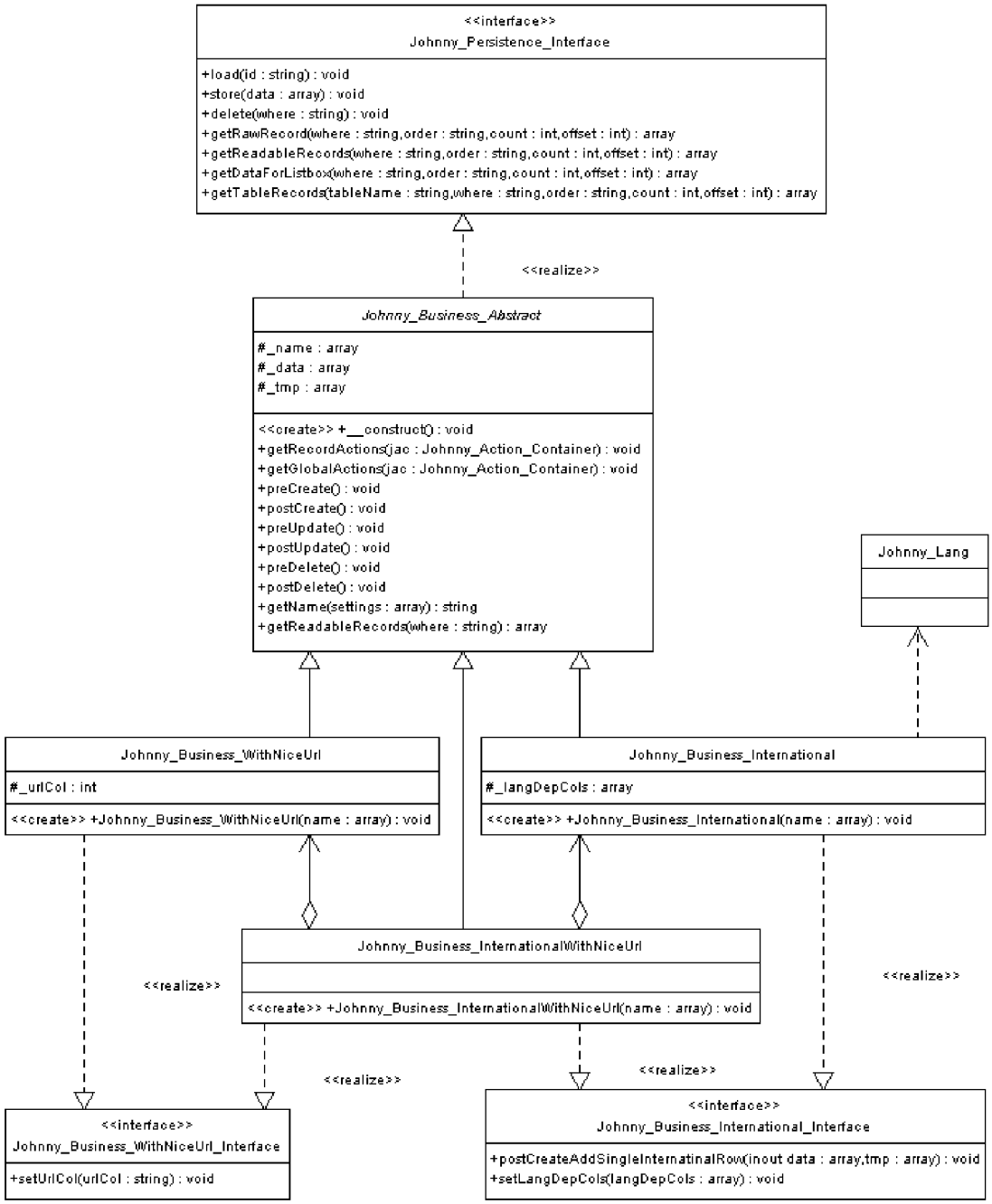






Obrázek 10: Balíček Security

A.1.5 Business



Obrázek 11: Balíček Business

V návrhovém vzoru Model-View-Controller (viz [17]) se jedná o část Model. Třídy jsou zodpovědné za příjem jednotlivých entit, přičemž nakládání s nimi se liší podle toho, zda je entita adresována

přes textový řetězec v URL (standardně přes id v URL) resp. zda je entita uchovávána v různých jazykových mutacích.

Johnny_Persistence_Interface

Rozhraní poskytující metody pro práci na perzistentním úložišti (databázi).

Johnny_Business_Abstract

Definuje převážně abstraktní metody pro správu business entit.

Johnny_Business_International

Implementuje správu entit v různých jazykových mutacích. Obsahuje vnitřní člen nesoucí informaci o mezinárodních položkách entity.

Johnny_Business_WithNiceUrl

Určuje zacházení s entitami, které jsou adresovány přes „hezká URL“ (např.: /cz/band/detail/sad-harmony).

Johnny_Business_InternationalWithNiceUrl

Jedná se o sjednocení tříd `Johnny_Business_International` a `Johnny_Business_WithNiceUrl`.

Johnny_Business_International_Interface a Johnny_Business_WithNiceUrl_Interface

Rozhraní jsou zavedena z důvodu snadnější a bezpečnější rozšiřitelnosti tříd „WithNiceUrl“ a „International“ – pokud bude zapotřebí přidat novou metodu, je vhodné modifikovat nejprve rozhraní a následné provést příslušné modifikace v obou třídách. Pokud bychom někde zapomněli metodu přidat, budeme interpretem upozorněni.

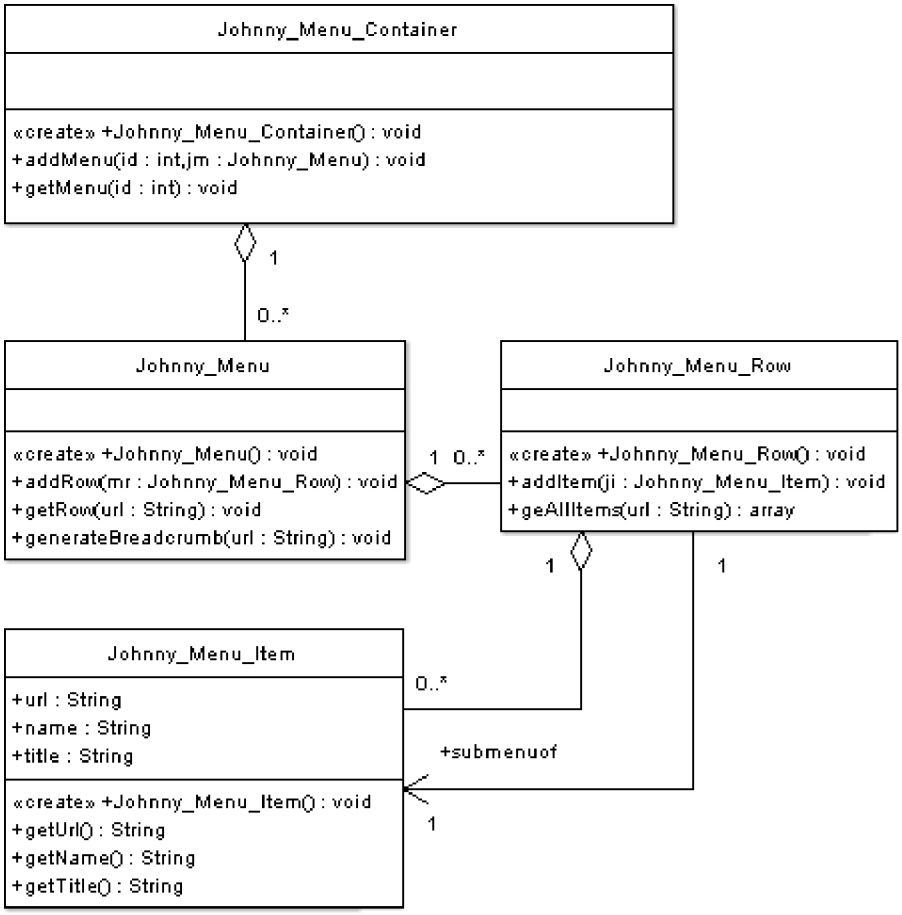
A.1.6 Filter

Obsahuje nástroje pro různé druhy filtrací a textových transformací. Jsou zde například filtry pro tvorbu hezkých URL, transformací jména entity na jméno databázové tabulky nebo na jméno business třídy apod.

Všechny filtry jsou odvozeny od třídy `Zend_Filter`.

Navigation

Balíček zodpovědný za veškerou navigaci v aplikaci.



Obrázek 12: Balíček Navigation

A.2 Správa entit v souladu se SEO

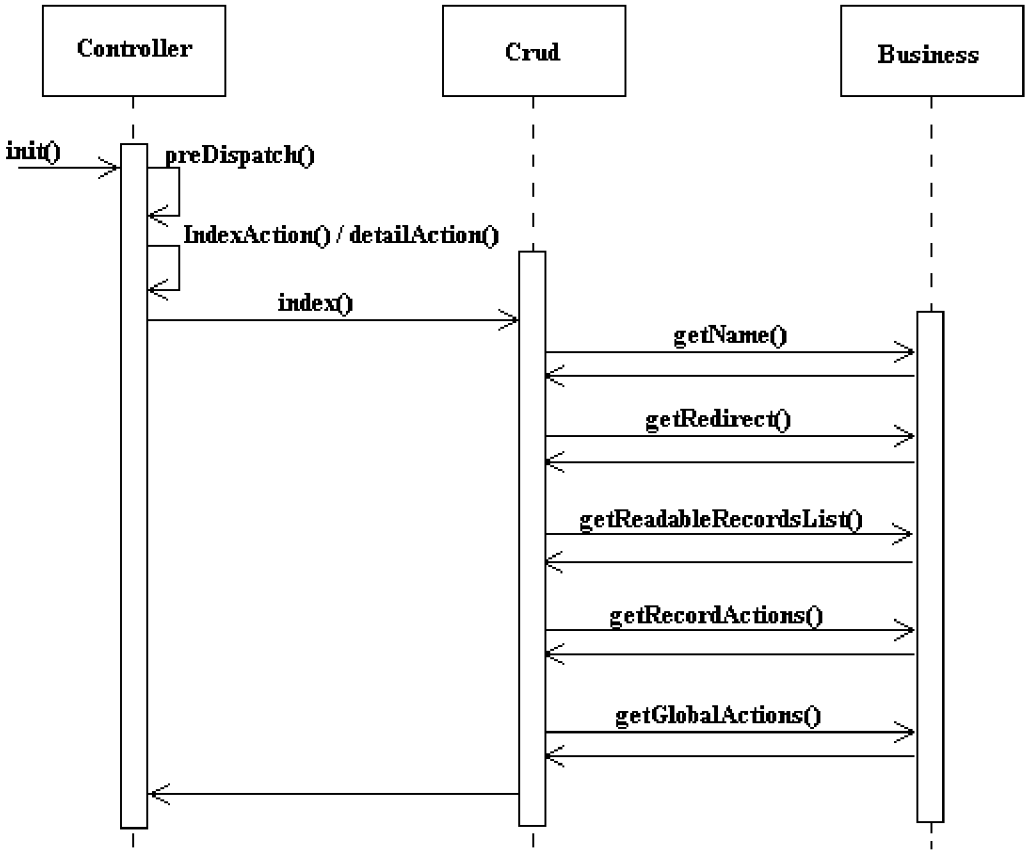
Jak se bude na programové úrovni entita chovat se odvíjí od toho, jakým způsobem ji chceme klíčovat, případně zda ji chceme uchovávat ve více jazykových mutacích.. Následující kapitoly popíší jak je tato problematika řešena v rámci diskutovaného frameworku.

Tato kapitola dopovídá realizaci návrhu popsaného v kapitole 9.1.1 Správa entit.

A.2.1 Zobrazování

Titulek bude generován ve tvaru *titulek entity | URL domácí stránky*.

Následující sekvenční diagram ilustruje posloupnost volání jednotlivých akcí, které vedou ke zobrazení seznamu entit (`indexAction()`), resp. detailu (`detailAction()`) entity.

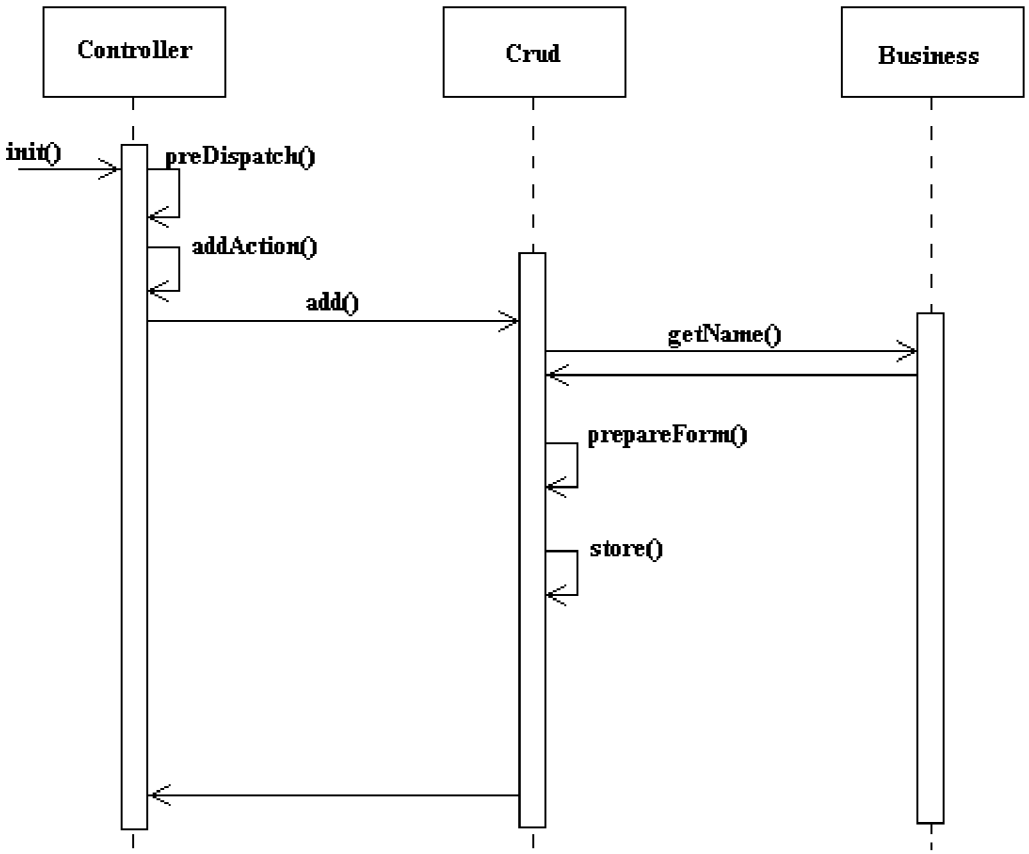


Obrázek 13: Sekvenční diagram zobrazování entit

Metoda `getRedirect()` identifikuje, zda nebylo navštíveno URL přejmenované entity, a zda tedy není potřeba provést přesměrování. Údaje pro titulek a nadpis jednotlivých entit je získán metodou `getName`(), která má parametr určující číslo (jednotné, množné) případně jazyk. Metody `getReadableRecordsList()`, resp. `getReadableRecord()`, získávají entitní data z perzistentního úložiště. Metody `getRecordActions()`, resp. `getGlobalActions()`, dodávají informace o možných operacích (akcích) nad entitami.

A.2.2 Přidávání

Pro přidání entity je potřeba vygenerovat formulář. Titulek těchto stránek má tvar *entita* | *přidej* | *URL domácí stránky*. Sled volání metod je následující:



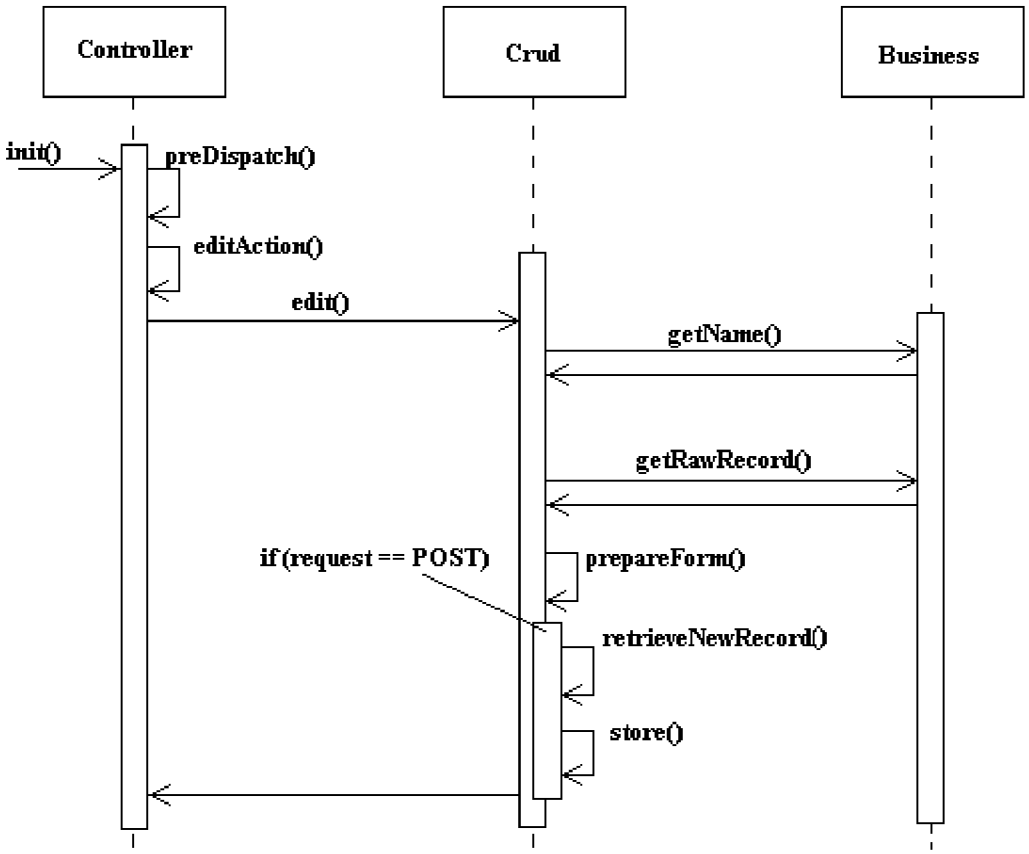
Obrázek 14: Sekvenční diagram přidávání nové entity

Metoda `prepareForm()` inicializuje formulář. Metoda `store()` ověří správnost dat a na základě existence id přijímané entity ji vytvoří (id neexistuje) nebo modifikuje (id existuje).

A.2.3 Modifikování

Je zapotřebí zobrazit formulář s požadovanými daty. Dále pak je třeba sledovat skutečnost, zda byl upraven údaj definující URL entity.

Titulek má tvar *jméno entity* | *uprav* | *URL domácí stránky*.

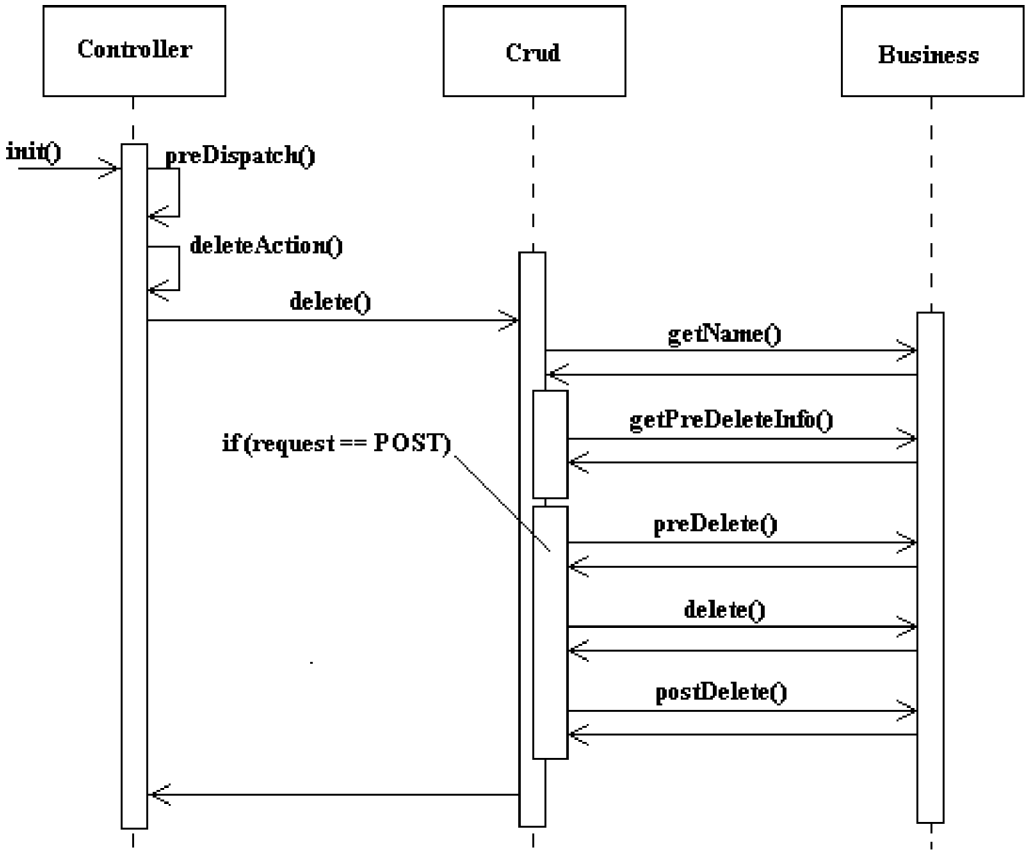


Obrázek 15: Sekvenční diagram modifikace entity

Metoda `getRawRecord()` vrací hodnoty pro korektní zobrazení formuláře (zejména hodnoty pro listboxy apod.).

A.2.4 Odstraňování

Data jsou smazána po odsouhlasení potvrzujícího dialogu. Titulek má tvar *jméno entity* | *smaž* | *URL domácí stránky*.



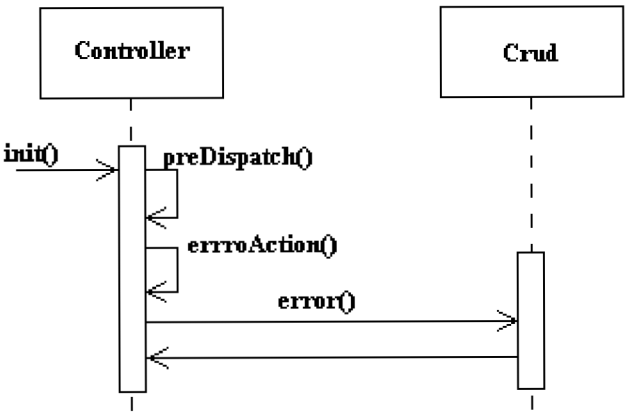
Obrázek 16: Sekvenční diagram odstraňování existující entity

Metoda `getPreDeleteInfo()` zobrazí souhrnné informace o právě odstraňované entitě.

A.3 Nerelevantní požadavky

Pokud je požadováno neexistující URL, je zapotřebí oznámit uživateli tuto skutečnost a vrátit odpovídající HTTP stavový kód (301).

Pokud Zend Framework detekuje, že je požadován neexistující kontrolér, resp. neexistující akce, je vykonána následující sekvence příkazů.



Obrázek 9.12: Sekvenční diagram nerelevantního požadavku

A.4 Ukázky kódů

Na následujících řádcích je stěžejní ukázky kódů pro základní typy entit.

A.4.1 Entity typu `Johnny_Business_WithNiceUrl`

Přidávání

Sekvenční diagram procesu přidávání obecné entity obsahuje metodu `store()`. Ta před vlastním přidáním záznamu zavolá metodu `preCrate()` a po jejím vykonání metodu `postCrate()` (v případě entit typu `Johnny_Business_WithNiceUrl` obsahuje prázdné tělo).

```
public function preCreate()
{
    Zend_Loader::loadClass('Johnny_Filter_NiceUrl');
    $filter = new Johnny_Filter_NiceUrl();
    $this->_data['url'] = $filter->filter($this->_data[$this->getUrlCol()]);

    $this->_tmp['inForUrl'] = $this->_data['inForUrl'];
    unset($this->_data['inForUrl']);
}
```

Kód A.1: Metoda `preCreate()` třídy `Johnny_Business_WithNiceUrl`

Atribut `_data` obsahuje data přidávané entity. Důležitý je řádek 3, kde dochází k vygenerování URL, který identifikuje přidávanou entitu. Obsah proměnné `_data` je později uložen do databáze.

Modifikování

Modifikace se od přidávání liší tím, že jsou volány metody `preUpdate()` a `postUpdate()`.

```
public function preUpdate()
{
    $this->preCreate();
    Zend_Loader::loadClass('Johnny_Filter_NiceUrl');
    $filter = new Johnny_Filter_NiceUrl();

    $oldUrl = $filter->filter($this->_tmp['inForUrl']);
    $newUrl = $filter->filter($this->_data[$this->getUrlCol()]);

    $this->_tmp['createNewRec'] = (strcasecmp($oldUrl, $newUrl) != 0);
    if ( $this->_tmp['createNewRec'] ) {
        $this->_tmp['id'] = @$this->_data['id'];
        unset($this->_data['id']);
    }

}
```

Kód A.2: Metoda `preUpdate()` třídy `Johnny_Business_WithNiceUrl`

Je zde identifikováno zda nedošlo ke změně hodnoty sloupce, ze kterého je odvozeno URL. Tato skutečnost je zaznamenána do proměnné `_tmp['createNewRec']`. Pokud došlo ke změně, jsou proměnné nastaveny tak, aby v metodě `store`() byla příchozí data vložena do nového řádku.

```
public function postUpdate()
{
    if ( $this->_tmp['createNewRec'] ) {
        $entity_id = $this->_data['entity_id'];
        $this->_data['id'] = $this->_tmp['id'];
        $this->_data['redirect'] = $this->_data['url'];
        $this->_data['status_code'] = '301';

        foreach ($this->_data as $key => $item) {
            if ($key != 'id' and $key != 'redirect' and $key != 'status_code' ) {
                unset($this->_data[$key]);
            }
        }

        $this->store();
        $this->_data['entity_id'] = $entity_id;
    }
}
```

Kód A.3: Metoda `postUpdate()` třídy `Johnny_Business_WithNiceUrl`

V případě, že došlo ke změně hodnoty sloupce pro odvození URL je v této fázi původní záznam upraven tak, aby hodnota *redirect* obsahovala URL entity vložené v metodě `store()` a stavový kód byl nastaven na 301.

A.4.2 Entity typu `Johnny_Business_International`

Přidávání

I zde jsou zavolány metody `preCreate()` a `postCreate()`. První zajistí oddělení jazykově závislých dat a jejich přesunutí do pomocné proměnné. Druhá načte jazykově závislá data a provede jejich uložení včetně vygenerování „prázdných“ řádků pro snadnější pozdější modifikaci dat v jiném jazyce (z důvodu přehlednosti není zdrojový kód metody `postCreate()` uveden).

```
public function preCreate()
{
    $depData = array();

    foreach ( $this->_langDepCols as $column ) {
        @$this->_tmp[$column] = $this->_data[$column];
        unset($this->_data[$column]);
    }

}
```

Kód A.4: Metoda preCreate() třídy Johnny_Business_International

Modifikování

Metoda `preUpdate()` volá kód metody `preCreate()`. Metoda `postUpdate()` modifikuje příslušná jazykově závislá data. Princip je podobný jako při vytváření.

B) ClassDiagram_CodeGenerator_Php

V rámci diplomové práce byl zrealizován nástroj pro převod UML diagramu (resp. jeho XMI podoby) na odpovídající business třídy. V technické zprávě byl uveden stručný popis, na následujících řádcích je k dispozici návod k použití.

Následuje jeho popis včetně názorné ukázky použití.

B.1 Úvod

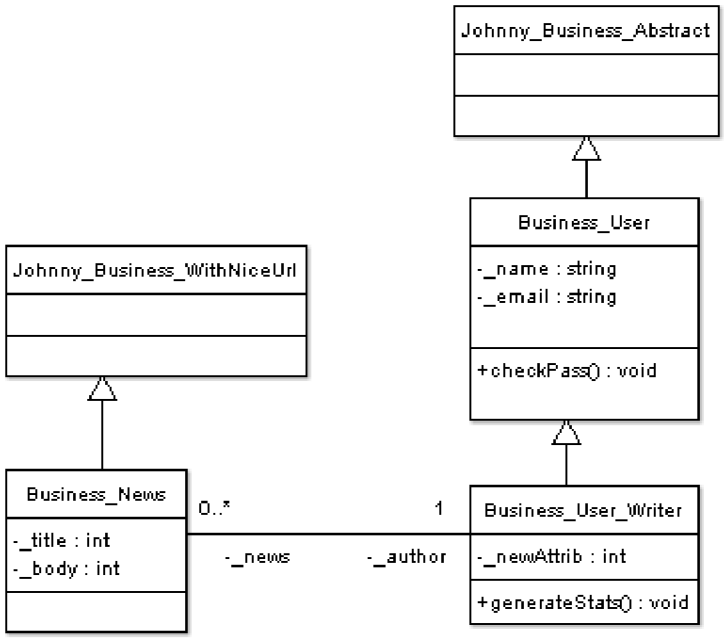
Vlastní proces tvorby se skládá ze třech hlavních kroků:

1. Vytvoření modelu nové aplikace.
2. Vygenerování kódu.
3. Začlenění vygenerovaného kódu do projektu.

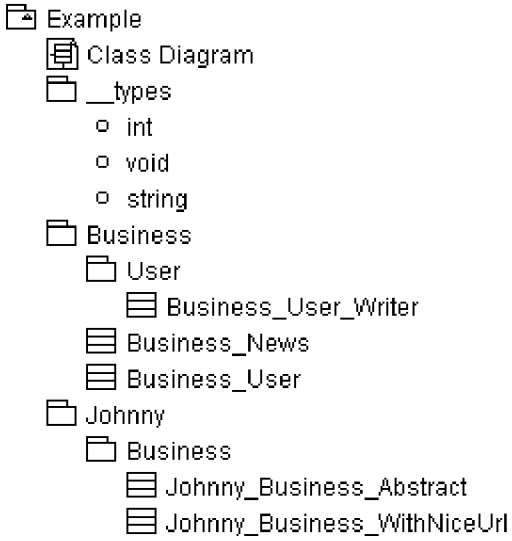
Adresářová struktura aplikace, kterou budeme vytvářet odpovídá *konvenční modulární adresářové struktuře*, viz [21]. V našem případě je stěžejní konvence organizování jednotlivých tříd. Tedy například třída `Business_Concert` bude uložena na v souboru Business/Concert.php.

B.2 Vytvoření modelu nové aplikace

Nejprve je zapotřebí vytvořit model. Pro tento krok poslouží software ArgoUML (argouml.tigris.org). Následující obrázky prezentuje jednoduchý model tříd.



Obrázek 17: Příklad diagramu tříd



Obrázek 18: Příklad orgranizace tříd

Na prvním diagramu je podstatné dědění od Johnny_Business tříd – čímž je vlastním třídám dodána dříve popsaná funkcionalita. Druhý obrázek sleduje hierarchickou organizaci tříd z hlediska balíčků.

B.2.1 Vygenerování kódu

S ohledem na skutečnost, že popisovaný nástroj je pouze první ze všech komponent cílového souboru nástrojů, může vlastní tvorba působit těžkopádně, ale nutno upozornit, že se nejedná o konečnou podobu. Proces generování kódu vyžaduje nainstalovanou podporu jednotkových testů – PHPUnit (phpunit.de).

Generování kódu má následující postup:

1. Vygenerování souboru XMI (V ArgoUML: File/Export XMI).
2. Vytvořit nový adresář v Johnny\TEST\Business\ClassDiagram\CodeGenerator\Php\{adresář} a zkopírovat do něj soubor *.XMI.
3. Ve stejném adresáři vytvořit soubor {jméno}Test.php, vložit do něj následující kód a upravit zvýrazněná místa:

```
<?php
require_once dirname(dirname(dirname(dirname(dirname(dirname(__FILE__))))))
                . DIRECTORY_SEPARATOR . 'TestHelper.php';

require_once 'Business/ClassDiagram/Model/Creator.php';
require_once 'Business/ClassDiagram/CodeGenerator/Php.php';
require_once 'Johnny/FileSystem/Directory.php';
require_once 'Johnny/FileSystem/File.php';


class ExampleTest
extends PHPUnit_Framework_TestCase
{

    public function testCreate()
    {
        $modelCreator = new Business_ClassDiagram_Model_Creator(
            'model.xmi'
        );
        $model = $modelCreator->create();

        $codeGeneratorPhp = new Business_ClassDiagram_CodeGenerator_Php(
            $model,
            'output'
        );
        $codeGeneratorPhp->generate();
    }

}
```

Obrázek 19: Příklad třídy generující cílový kód

4. V příkazovém řádku se nastavit do pracovního adresáře a spustit příkaz `phpunit ExampleTest`. Do zadaného adresáře bude vygenerován namodelovaný kód. Zde je

prezentován pouze soubor news.php, ostatní jsou k dispozici v adresáři PŘÍLOHY\SRC\Example\output.

```php
<?php

require_once "Johnny/Business/WithNiceUrl.php";
require_once "Business/User/Writer.php";

/**
 */
class Business_News extends Johnny_Business_WithNiceUrl
{

    /**
     * @type Business_User_Writer
     */
    private  $_author;


    /**
     * Getter of the _author.
     *
     * @return  Business_User_Writer
     */

    public function getAuthor()
    {
        return $this->_author;
    }


    /**
     * Getter of the _author.
     *
     * @param   Business_User_Writer
     * @return  void
     */

    public function setAuthor()
    {
        $this->_author = $_author;
    }


}
```

Obrázek 20: Vygenerovaný soubor Business/news.php

B.2.2 Začlenění vygenerovaného kódu do projektu

Adresář output/Business zkopírovat do adressáře{projekt}\application\models\Business.

C) Případová studie

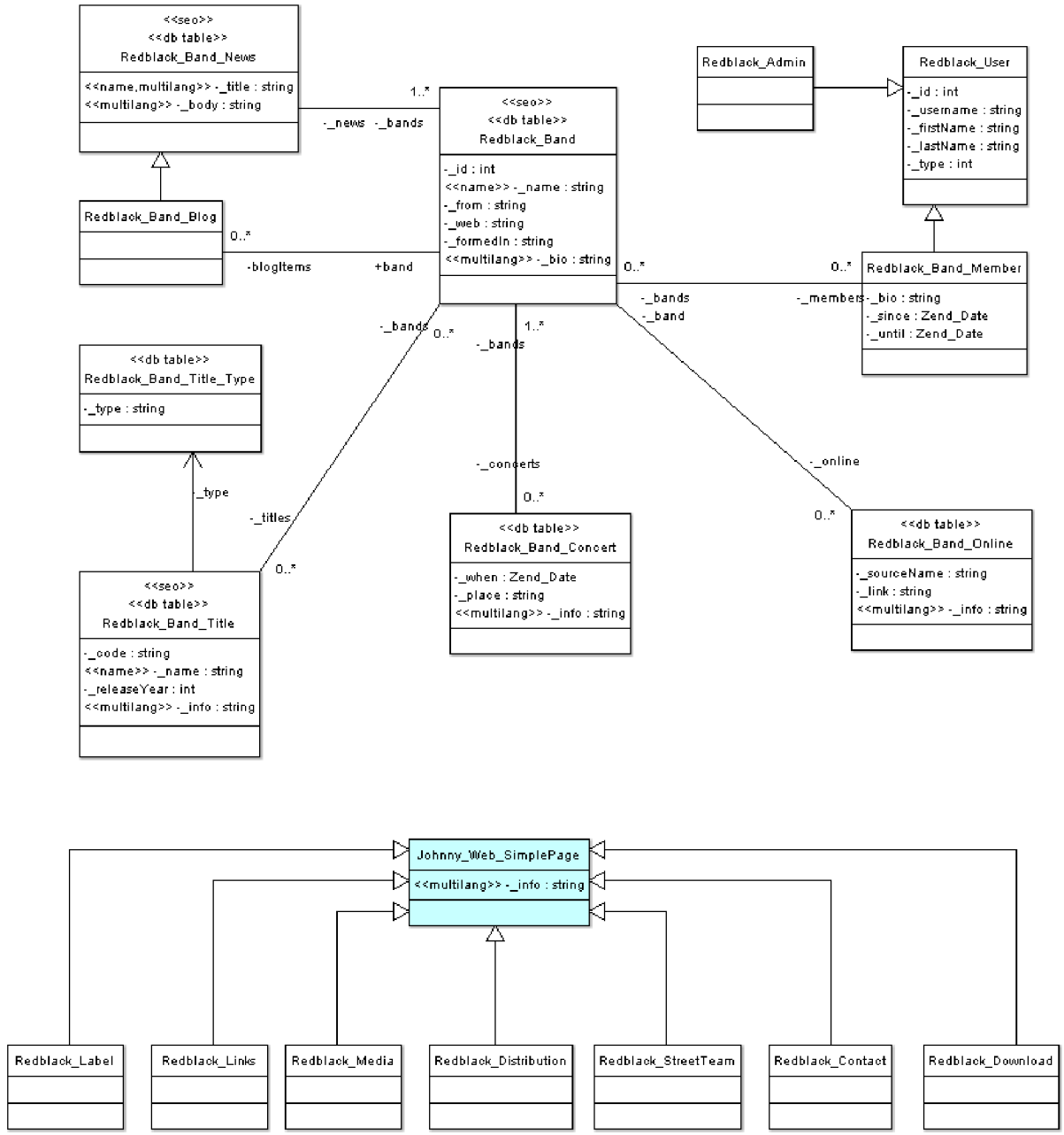
Jako případovou studii jsem zvolil internetové stránky Brněnského hudební vydavatelství Redblack. Stránky musí nabízet správu následujících entit:

- Novinky,
- kapely,
- hudebníci,
- tituly,
- spřátelená vydavatelství,
- kontakt, media, distribuce …

C.1 Návrh

Následující diagram zachycuje všechny modelované entity a vyjadřuje vztahy mezi nimi.

Za povšimnutí stojí stereotyp `<<seo>>`, jím označené třídy budou minimálně dědit od třídy `Johnny_Business_WithNiceUrl`. Pokud je nějakému atributu takové třídy přidělen stereotyp `<<multilang>>` bude odvozena od `Johnny_Business_InternationalWithNiceUrl`.



Obrázek 21: Diagram tříd hudebního vydavatelství

C.2 Ukázky z kódu

Tato kapitola se zaměří na jednotlivé části návrhového vzoru „Model-View-Controller“, resp. na jejich realizaci v rámci aplikace a dále pak na realizaci databázové vrstvy. Vždy bude uveden jeden příklad za všechny.

C.2.1 Model

Následující kód je realizací modelu titulu. Titulem se rozumí hudební nosič.

```
class Business_Redblack_Title extends Johnny_Business_InternationalWithNiceUrl
{

    public function __construct()
    {
        $this->_name = array(Johnny_Lang::CZ =>
                                array(Johnny_Lang::SINGULAR => 'titul',
                                      Johnny_Lang::PLURAR => 'tituly'),

                             Johnny_Lang::EN =>
                                array(Johnny_Lang::SINGULAR => 'title',
                                      Johnny_Lang::PLURAR => 'titles'));

        parent::__construct($this->_name);

        $this->setUrlCol('name');
        $this->setLangDepCols(array('info', 'lang'));
    }

}
```

Kód C.1: Model - Titul

Nejprve je inicializována proměnná `_name`. Nese jméno entity v různých mluvnických číslech a jazykových mutacích. Následně je zavolán rodičovský konstruktor, kterému jsou předány informace o jméně. Dále pak je nastaven sloupec databázové tabulky z něhož bude generováno klíčové URL. Poslední řádek identifikuje jazykově závislé sloupce.

C.2.2 Controller

Každá kapela může pořádat koncert a na jednom koncertu může vystupovat více kapel (tedy vztah N:M). Situace je v „Controleru“ zachycena následovně:

```
class ConcertController extends Business_Redblack_Controller_Action
{

    public function init()
    {
        Johnny_Crud::init($this);
        $this->_business = new Business_Redblack_Concert();
        $this->_form = new Business_Redblack_Concert_Form();
        $this->_nm = array( 'name'   => 'bands',
                            'inst'   => new Business_Redblack_Concert_Of_Band(),
                            'l'      => 'concert_id',
                            'N'      => 'band_id'    );
    }


}
```

Kód C.2: Controller - Koncert

Je provedena inicializace objektu `Johnny_Crud`, business třídy a jejího formuláře a následně vyjádření vztahu N:M mezi koncertem a kapelou, přesněji je zde řečeno, že bude ke každému koncertu dodán výpis všech kapel.

C.2.3 View

Zobrazení titulu.

```
Zend_Loader::loadClass('Johnny_Filter_NiceUrl');
$filter = new Johnny_Filter_NiceUrl();

if (isset($this->tableData[Business_Redblack_Const::TITLE])) {
    foreach ($this->tableData[Business_Redblack_Const::TITLE]['data'] as $title) {

        // body
        echo '<img class="imgcdtitle" src="/public/images/band/'
             . $filter->filter($title['band_name']) . '/title/'
                . $filter->filter($title['name']) . '.jpg" alt="'
                . $title['band_name'] . '" />'
             . '<h3>' . Business_Redblack_Lang::getPhrase(
                'name', Zend_Registry::get('lang')) . ': </h3>'
             . '<p>' . $title['name'] . '</p>'
             . '<h3>' . Business_Redblack_Lang::getPhrase(
                'band', Zend_Registry::get('lang')) . ': </h3>'
             . '<p>' . $title['band_name'] . '</p>'
             . '<h3>' . Business_Redblack_Lang::getPhrase(
                'release', Zend_Registry::get('lang')) . ': </h3>'
             . '<p>' . $title['release_year'] . '</p>'
             . '<h3>' . Johnny_Lang::getPhrase(
                'type', Zend_Registry::get('lang')) . ': </h3>'
             . '<p>' . $title['type'] . '</p>'
             . '<h3>'. Johnny_Lang::getPhrase(
                'code', Zend_Registry::get('lang')) .': </h3>'
             . '<p>' . $title['code'] . '</p>'
             . '<h3> Info: </h3>'
             . '<p>' . $title['info'] . '</p>';

        // actions
        echo Johnny_Action_View::getRecordActions($this,
                                                  $this->subject,
                                                  $title['id']);
    }

}
```

Kód C.3: View - Titul

Postupně jsou vypsány všechny tituly, které byly načteny v částech Model a Controller. V závěru jsou vypsány relevantní akce nad entitou – pokud byly nějaké danému uživateli (resp. roli) přiděleny.

C.2.4 Databázové tabulky a pohledy

Pro demonstraci jsem vybral entitu titul. Informace o ní jsou uchovávány ve více jazycích a podporuje hezká URL.

Základní tabulka obsahuje jazykově nezávislé atributy a informace o přesměrovávání.

```
CREATE TABLE IF NOT EXISTS `title` (
  `id` smallint(5) unsigned NOT NULL auto_increment,
  `band_id` tinyint(3) unsigned NOT NULL,
  `code` char(10) collate utf8_czech_ci default NULL,
  `name` varchar(128) collate utf8_czech_ci NOT NULL,
  `release_year` smallint(5) unsigned NOT NULL,
  `type_id` tinyint(3) unsigned NOT NULL default '1',
  `url` varchar(128) collate utf8_czech_ci default NULL,
  `status_code` varchar(64) collate utf8_czech_ci default NULL,
  `redirect` varchar(512) collate utf8_czech_ci default NULL,
  `who_last_modified` smallint(5) unsigned NOT NULL,
  `ts` timestamp NOT NULL default CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP,
  PRIMARY KEY  (`id`),
  KEY `band_id` (`band_id`),
  KEY `type_id` (`type_id`),
  KEY `who_last_modified` (`who_last_modified`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

ALTER TABLE `title`
  ADD CONSTRAINT `title_ibfk_1` FOREIGN KEY (`band_id`)
    REFERENCES `band` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `title_ibfk_2` FOREIGN KEY (`type_id`)
    REFERENCES `title_type` (`id`) ON DELETE NO ACTION ON UPDATE CASCADE,
  ADD CONSTRAINT `title_ibfk_3` FOREIGN KEY (`who_last_modified`)
    REFERENCES `user` (`id`) ON DELETE NO ACTION ON UPDATE CASCADE;
```

Kód C.4: Databázová tabulka „title“

Tabulka `title_international` obsahuje jazykově závislá data. Odkazuje se na základní tabulku.

```
CREATE TABLE IF NOT EXISTS `title_international` (
  `id` smallint(5) unsigned NOT NULL auto_increment,
  `title_id` smallint(5) unsigned NOT NULL,
  `info` text collate utf8_czech_ci NOT NULL,
  `lang` enum('cz','en') collate utf8_czech_ci NOT NULL,
  `who_last_modified` smallint(5) unsigned NOT NULL default '1',
  `ts` timestamp NOT NULL default CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP,
  PRIMARY KEY  (`id`),
  KEY `title_id` (`title_id`),
  KEY `who_last_modified` (`who_last_modified`),
  KEY `ts` (`ts`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

ALTER TABLE `title_international`
  ADD CONSTRAINT `title_international_ibfk_1` FOREIGN KEY (`title_id`)
    REFERENCES `title` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `title_international_ibfk_2` FOREIGN KEY (`who_last_modified`)
    REFERENCES `user` (`id`) ON DELETE NO ACTION ON UPDATE CASCADE;
```

Kód C.5: Databázová tabulka „title_international“

Pohled `v_title_cz` je použit při detailním náhledu na entitu.

```
CREATE OR REPLACE VIEW `v_title_cz` AS
SELECT
  t.id,
  t.name,
  t.release_year,
  t.code,
  ti.info,
  b.name band_name,
  t.url,
  tti.type
FROM
  title t,
  title_international ti,
  band b,
  title_type_international tti
WHERE
  t.band_id = b.id
  AND t.id = ti.title_id
  AND t.type_id = tti.title_type_id
  AND tti.lang='cz'
  AND ti.lang='cz'
  AND t.redirect is null
ORDER BY
  t.release_year DESC;
```

Kód C.6: Pohled pro zobrazení detailu titulu

Pohled `v_title_raw_cz` je používán pro načtení formulářových dat.

```
CREATE OR REPLACE VIEW `v_title_raw_cz` AS
SELECT
  t.id,
  t.name,
  t.release_year,
  t.code,
  ti.info,
  t.band_id,
  t.type_id,
  ti.lang,
  t.name inForUrl
FROM
  title t,
  title_international ti,
  title_type tt,
  title_type_international tti
WHERE
  t.id = ti.title_id
  AND t.type_id = tt.id
  AND tt.id = tti.title_type_id
  AND tti.lang='cz'
  AND ti.lang='cz'
ORDER BY
  t.release_year DESC;
```

Kód C.7: Pohled pro načtení formulářových dat

Pohled `v_title_list_cz` se používá při zobrazování seznamu entit titul.

```
CREATE OR REPLACE VIEW `v_title_list_cz` AS
SELECT
  t.id,
  t.name,
  b.name band_name,
  t.release_year,
  t.url,
  t.band_id
FROM
  title t,
  band b,
  title_type tt
WHERE
  t.band_id = b.id
  AND t.type_id = tt.id
  AND t.redirect is null
ORDER BY
  t.release_year DESC;
```

Kód C.8: Pohled pro zobrazení seznamu titulů

C.2.5 Soubor robots.txt

Pomocí souboru robots.txt, je vyhledávačům zakázána indexace stránky pro přihlášení a všech stánek, které čekají na vložení vhodného obsahu.

```
User-agent: *
     Disallow: /cz/auth
     Disallow: /en/auth
     Disallow: /cz/link
     Disallow: /en/link
     Disallow: /cz/download
     Disallow: /en/download
     Disallow: /cz/media
     Disallow: /en/media
     Disallow: /en/streetteam
```

Kód C.9: Případová studie - obsah souboru robots.txt

C.2.6 Soubor .htaccess

Je zde definováno přesměrování z URL s prefixem www na variantu bez něho. Dále je definováno přesměrování všech příchozích požadavků do bootstrapu (viz framework.zend.com/manual/en, kapitola 7. Zend_Controller).

```
RewriteEngine On

RewriteCond %{HTTP_HOST} ^www.new.redblack\.cz
RewriteRule ^(.*)$ http://new.redblack.cz/$1 [R=301,L]

#RewriteRule !\.(js|ico|gif|jpg|png|css)$ index.php

RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule .* /index.php
```

Kód C.10: Případová studie - obsah souboru .htaccess

C.2.7 Soubor redirect.php

Zde prezentovaný web je náhradou za již existující. Aby bylo zachováno *hodnota* odkazů bylo zapotřebí provést příslušná přesměrování. To zajišťuje kód souboru redirect.php, který je vložen hned na začátku bootstrapu (index.php).

Následuje ukázka fragmentu kódu.

```
$lang = explode('/', $_SERVER['REQUEST_URI']);
$lang = $lang[1];

if ( $_SERVER['REQUEST_URI'] == '/' . $lang . '/index.html' ) {
    Header( 'HTTP/1.1 301 Moved Permanently' );
    Header( 'Location: /' );
    exit;
} elseif ( $_SERVER['REQUEST_URI'] == '/' . $lang . '/releases.php' ) {
    Header( 'HTTP/1.1 301 Moved Permanently' );
    Header( 'Location: /' . $lang . '/title' );
    exit;
} elseif ( $_SERVER['REQUEST_URI'] == '/' . $lang . '/tourdates.php' ) {
    Header( 'HTTP/1.1 301 Moved Permanently' );
    Header( 'Location: /' . $lang . '/concert' );
    exit;
}
```

Kód C.11: Případová studie - redirect.php

C.2.8 Testovací verze stránek

Adresa webu	redblack.jan-stefl.eu
Přihlašovací stránka	redblack.jan-stefl.eu/cz/auth
uživatelské jméno	tester
Heslo	pokus

Po přihlášení přibudou na stránkách nové ikony. Někdy je potřeba zobrazit detail (například u novinky kliknutím na obrázek pro zobrazení operací *Uprav* resp. *Odstraň*)

	Přidá nové údaje
	Upraví aktuální údaje
	Odstraní vybrané údaje

Zda byla entita úspěšně přidána, je potřeba zpětně zkontrolovat.

C.3 Zhodnocení výsledků ve vyhledávačích

Měření provedeno po třech dnech umístění stránek do provozu.

- - nenalezeno
- N(X) neaktuální data, X určuje pozici
- P(X) nalezena data spjatá s vydavatelstvím, X určuje pozici (zpravidla stránky obchodu)

Tabulka 37: Zhodnocení výsledků ve vyhledávačích

Výraz	Google	MS Live	Yahoo!	Seznam.cz	Jyxo.cz	Komentář
redblack	1	N(1)	N(2)	N(4)	P(1)	Jyxo – stránky obchodu
redblack novinky	1	P(1)	P(1)	P(8)	P(1)	Všechna P jsou stránky obchodu
redblack kapely	N(3)	N(1)	N(3)	N(4)	-	
redblack sad harmony	N(2)	N(2)	N(2)	N(1)	N(1)	
redblack memoria koncert	N(9)	N(3)	-	-	-	
kapela Apatheia	N(7)	-	-	-	-	
silent stream of godless elegy	N(7)		N(7)	N(2)	-	

Vyhledávané fráze jsou seřazeny sestupně podle pravděpodobnosti kvalitního umístění v SERPs (odhad).

Nutno předem říci, že by bylo velmi vhodné, aby bylo provedeno více takovýchto porovnání s odstupem určitého časového úseku. Bohužel to se mi už nepodařilo uskutečnit.

Jak je z tabulky patrné vyhledávače v časovém intervalu tří dnů nestačili obnovit údaje v mezipaměti. Výjimkou je pouze vyhledávač Google, který je u prvních svou frází zcela relevantní. Všude, kde je v tabulce uvedeno N(X), se dá předpokládat, že brzy dojde k zaktualizování těchto údajů, protože v drtivé většině případů je u nalezených odkazů přítomno správné přesměrování (včetně generování HTTP stavového kódu 301).

Z výsledků je dále patrné, že čím hlouběji je údaj zanořen ve struktuře webu, tím hůře je umístěn ve výsledcích, příkladem jsou kapely. Možným vylepšením by mohlo být vytvoření nového panelu kde bude obsažen seznam všech kapel, informující například o žánrovém zařazení. Obdobné

vylepšení by bylo vhodné například pro koncerty, tedy na úvodní stránce kupříkladu zobrazovat koncerty na měsíc dopředu.

Obecně lze závěrem říci, že vše podstatné by mělo být dostupné na úvodní stránce a z ní odkazovat na podrobnosti ukryté v hlubší úrovni webu.

D) Přehled užitečných nástrojů

Název : **Google webmaster tools**

Adresa : google.com/webmasters/tools

Popis : Základní rysy:

- sleduje frekvenci návštěv stránek vyhledávačem
- identifikuje validitu robots.txt
- identifikuje atraktivní obsah
- zjišťuje, které vyhledávané fráze směřují na sledovaný web
- poskytuje různé informace o indexaci
- umožňuje nahrát mapu webu pro lepší procházení stránek vyhledávačem
- …

Nepostradatelná pomůcka pro práci v oblasti SEO.

Název : **Search status (extenze pro Firefox)**

Adresa : www.quirk.biz/searchstatus

Popis : Zobrazuje odhadované hodnoty prohlížené stránky pro:

- PageRank
- Alexa rank
- Compete.com rank

Dále poskytuje:

- informaci o příchozích odkazech
- grafy návštěvnosti apod.

Název : **Google trends**

Adresa : google.com/trends

Popis : Zobrazuje statistické informace o klíčových slovech v kontextu světového dění.

Název : **10^3bees**

Adresa : 103bees.com

Popis : Online služba zaměřená na analýzu přirozeného provozu vyhledávačů. Obsahuje rozsáhlé aktuální statistické informace o sledovaném webu.

Název : **LiveHTTPHeaders (extenze pro Firefox)**

Adresa : livehttpheaders.mozdev.org

Popis : Pomocník pro analýzu http hlaviček.

Název : **Copyscape**

Adresa : www.copyscape.com

Popis : Vyhledávání ukradeného obsahu.

Název : **Pozice ve vyhledávačích – SEO optimalizace**

Adresa : http://www.pozice-ve-vyhledavacich.cz/

Popis : Přehled pozice stránky ve vyhledávačích k zadané klíčové frázi – velmi přeh.ledné

Název : **Dnsstuff**

Adresa : www.dnsstuff.com

Popis : *DNSreport* - komplexní informace o konkrétní doméně
WHOIS Lookup - hledá kontaktní informace na vlastníka dané domény
Reverse DNS lookup - identifikuje hostname asociované se specifickou IP adresou

E) Slovníček pojmů

CTR - Click-through rate. Česky se vžilo označení proklik. Základní definice říká, že to je podíl počtu kliknutí na odkaz a počtu jeho zobrazení. Udává se v procentech. Například pokud na 100krát zobrazený odkaz bylo kliknuto jedním uživatelem, bude CTR 1%. Tato hodnota se v průměru pohybuje hluboko pod jedním procentem.

SERPs - S*earch engine results pages.* Skupina stránek vrácená vyhledávačem, odpovídající k zadané frázi.

přístupnost - Pod pojmem přístupnost chápeme takový stav, kdy daná věc neklade svým uživatelům při používání žádné překážky. Přístupnou budovu mohou tedy např. používat vozíčkáři a přístupný web zase např. slabozrací. Přístupnost je tedy bezbariérovost.