Adaptívny proxy server

Technická dokumentácia

Tím 13 – Old School Brothers

Tímový projekt 2010/2011
Fakulta informatiky a informačných technológií
Slovenská technická univerzita

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Tím 13 – Old School Brothers
email: team013@googlegroups.com

Obsah

Obsah 1
Šprint 1 – Eniac 2
Inicializácia pomocou evercookie 3
Prenos evercookie 4
Prezeranie logov 5
Nastavenie evercookie do response 6
Logovanie času 8
Prístup do logov 9
Odstraňovanie Evercookie 10
Šprint 2 – MITS Altair 8800 11
Offline tasky v offline adresari 14
Folder pre migrácie 15
Adresár pre konfigurácie pluginov 16
Adresár pre statický obsah 17
Adresár pre libky 18
Adresár pre zdrojové súbory 19
Spustenie tasku po deploymente 20
Šprint 3 – Fairchild Illiac-IV 21
Messageboard nad stránkou 22
SearchEngineService a získavanie výsledkov vyhľadávania z Google 25
SearchEngineService a menenie výsledkov vyhľadávania z Google 26
Šprint 4 – Comodore 27
Prezeranie metadát k aktuálnej stránke 28
SearchEngineService a získavanie vvýsledkov vyhľadávania z Yahoo a Bing 30
SearchEngineService a menenie výsledkov vyhľadávania z Google 31
Proxy používa Metall 33
Modul pre vytváranie DOM webovej stránky 34
Šprint 5 – ZX Spectrum+ 37
Refactoring, prerobenie messageboard + keywords 38
Celkový pohľad na riešenie po zimnom semestri 40

Adaptívny proxy server 40
Naša práca 41
Technológie 43
Prototyp 43
Šprint 6 – PMD85 44
Paralelne logovanie prístupov do couchDB 45
Možnosť nahlásenia nefunkčnej stránky 46
Mergovanie variables.xml z viacerých bundlov 48
Šprint 7, 8 – Didaktik 1, 2 49
Asynchrónne načítavanie JavaScript súborov 50
Úprava proxy webu na CouchDB 52
Hromadnejšie mazanie z logov cez označenie checkboxami 54
Šprint 9 – Didaktik 3 55
Identifikácia cieľa vyhľadávania na webe 56
Realtime social navigation 58

Šprint 1 – Eniac

Požiadavky

- Proxy bez browser-patcher

Analýza

Na to, aby adaptívny proxy server mohol plniť dostatočne zaujímavé úlohy, je potrebné, aby vedel jednoznačne identifikovať klienta. Používa sa na to vygenerovaný reťazec čísel, ktorý je jedinečný pre každého používateľa. Tento reťazec je však potrebné uložiť na strane klienta tak, aby sa táto informácia posielala do requestov.

V minulosti túto úlohu zabezpečoval user-agent, čo je premenná, ktorá popisuje prostredie klienta. Na to, aby sa do user-agenta uložil aj identifikátor používateľa však bolo potrebné robiť užívateľsky nepohodlné nastavenia, prípadne si inštalovať program, ktorý tieto nastavenia urobil za nás.

Ukázala sa teda potreba ukladať identifikátor používateľa iným spôsobom. Jedným z možných riešení je aj využitie technológie evercookie, ktorá uloží ľubovoľnú premennú takým spôsobom, že je pomerne ťažké ju odtiaľ vymazať.

Úlohou tohto šprintu je teda zmeniť spôsob ukladania a čítania identifikátora používateľa pomocou technológie evercookie.

Používateľské príbehy

Požiadavka bola rozdelená na tieto používateľské príbehy (*user-stories*):

- Inicializácia pomocou evercookie
- Prenos evercookie
- Prezeranie logov
- Nastavenie evercookie do response
- Logovanie času
- Prístup do logov
- Odstraňovanie evercookie

Jednotlivé príbehy sú opísané v nasledujúcich podkapitolách.

Inicializácia pomocou evercookie

Analýza

Táto úloha sa zaoberá procesom inicializácie nového používateľa – teda vygenerovanie a uloženie identifikátora pre klienta, ktorý ešte žiaden identifikátor nemá. Túto skutočnosť bude potrebné implementovať pomocou technológie evercookie.

Webová stránka proxy servera robí kontrolu, či už užívateľ má pridelený identifikátor. Ak zistí, že nemá, ponúkne mu návod na nastavenie user-agenta alebo stiahnutie Browser Patchera. Úlohou je teda zabezpečiť, aby sa identifikátor kontroloval pomocou novej technológie, a taktiež aby stránka už neponúkala návod na nastavenie user-agenta.

Návrh

Na mieste, kde sa doteraz kontroloval identifikátor z user-agenta dorobiť kontrolu aj na evercookie. Kontrolu na user-agenta je potrebné nechať kôli zachovaniu spätnej kompatibility.

Na mieste, kde bolo ponúkané stiahnutie Browser Patchera je potrebné dorobiť linku, ktorá nastaví evercookie. Ak mal používateľ nastavený identifikátor pomocou user-agenta, táto hodnota zostáva uložená v session a tá istá hodnota sa zapíša aj do evercookie. Tím sa zabezpečí spätná kompatibilita.

Taktiež je potrebné vymazať časti stránky, ktoré obsahujú návod na nastavenie user-agenta. Nakoniec je potrebné zmeniť texty na stránke tak, aby zodpovedali novej technológii.

Implementácia a testovanie

Úloha bola úspešne implementovaná do existujúcej stránky proxy servra v skriptovacom jazyku Ruby on Rails. Samotná technológia evercookie je naprogramovaná v javascripte, bolo potrebné importovanie knižnice, ktoré sa však už vykonalo v rámci inej úlohy.

Implementácia bola pretestovaná na lokálnom serveri, neskôr nasadená aj na našu produkčnú verziu (staging).

Prenos evercookie

Šprint: Eniac; *zodpovedná osoba:* Pavol Sokol

Analýza

Pri prechode identifikácie používateľa z user-agent hlavičky prehliadača na evercookie vznikla požiadavka preklopiť existujúcu funkcionalitu. Prenos evercookie slúži na uloženie identifikátora používateľa z jedného prehliadača a jeho načítanie na prehliadače ďalších zariadení.

Návrh

Zadanie bude riešené v dvoch samostatných úlohách. Prvá úloha rieši uloženie evercookies, v ktorej sa do databázovej tabuľky stored_apuids uloží spolu s evercookie aj užívateľom zadaný reťazec, podľa ktorého sa neskôr vyhľadá správna cookie, čo je zasa druhá úloha. Pôvodná implementácia pre nastavenie identifikátora vygenerovala aplikáciu browser patcher pre upravenie user-agent hlavičky prehliadača. V novom návrhu bude do počítača uložená evercookie. Taktiež je potrebné upraviť texty a tlačidlá v používateľskom rozhraní.

Implementácia a testovanie

Keďže identifikátor pre užívateľský identifikátor ostáva vnútri systému rovnaký, pre úlohu uloženia evercookie je potrebné iba zmeniť texty a tlačidla na stránke. Pre získanie a uloženie evercookie je potrebné definovať novú funkciu retrieve_uid_ever_cookie, ktorá po požiadavke od klienta získa v hlavičke POST identifikátor. Ten porovná s existujúcimi záznamami v databáze a vyhľadá zhodu. V prípade, ak zhodu nenájde vypíše chybu, inak vygeneruje html a javascript kód, ktorý vráti a nastaví evercookie identifikátor u klienta. Tiež je potrebné ošetriť prázdny vstup od užívateľa.

Prezeranie logov

Šprint: Eniac; z*odpovedná osoba:* Pavol Sokol

Analýza

Pri prechode identifikácie používateľa z user-agent hlavičky prehliadača na evercookie vznikla požiadavka preklopiť existujúcu funkcionalitu. Prezeranie logov slúži používateľom pre prípadnú spätnú kontrolu navštívených stránok. Dáta sú čítané z databázy a renderované používateľom. Pri prechode na evercookie ostávajú identifikátory aj úložiská zhodné, čo z prvotného návrhu nebolo zrejmé, takže na splnenie tejto požiadavky nie je potrebné systém prerábať.

Nastavenie evercookie do response

Šprint: Eniac; z*odpovedná osoba:* Ivan Pleško

Po zrušení identifikácie user-agentom nastáva problém identifikovania používateľa. Do stránky treba vkladať javascript, ktorý nastaví u klienta premennú s jeho uid. Ak má používateľ user-agenta, treba mu uid nastaiť aj do evercookie. Zabezpečí sa tak spätná kompatibilita.

Analýza

UID je údaj, ktorý má používateľ uložený vo svojom počítači pomocou evercookie nastavenú pre doménu proxy-webu. Môžu nastať 2 situácie:

1. Používateľ má nastavenú cookie pre proxy-web, a teda stačí vygenerovať javascript, ktorý nastaví premennú
2. Používateľ nemá nastavenú cookie, ale dá sa obnoviť pomocou evercookie. Ak je tento stav zistený, treba používateľovu cookie obnoviť z evercookie.

Návrh

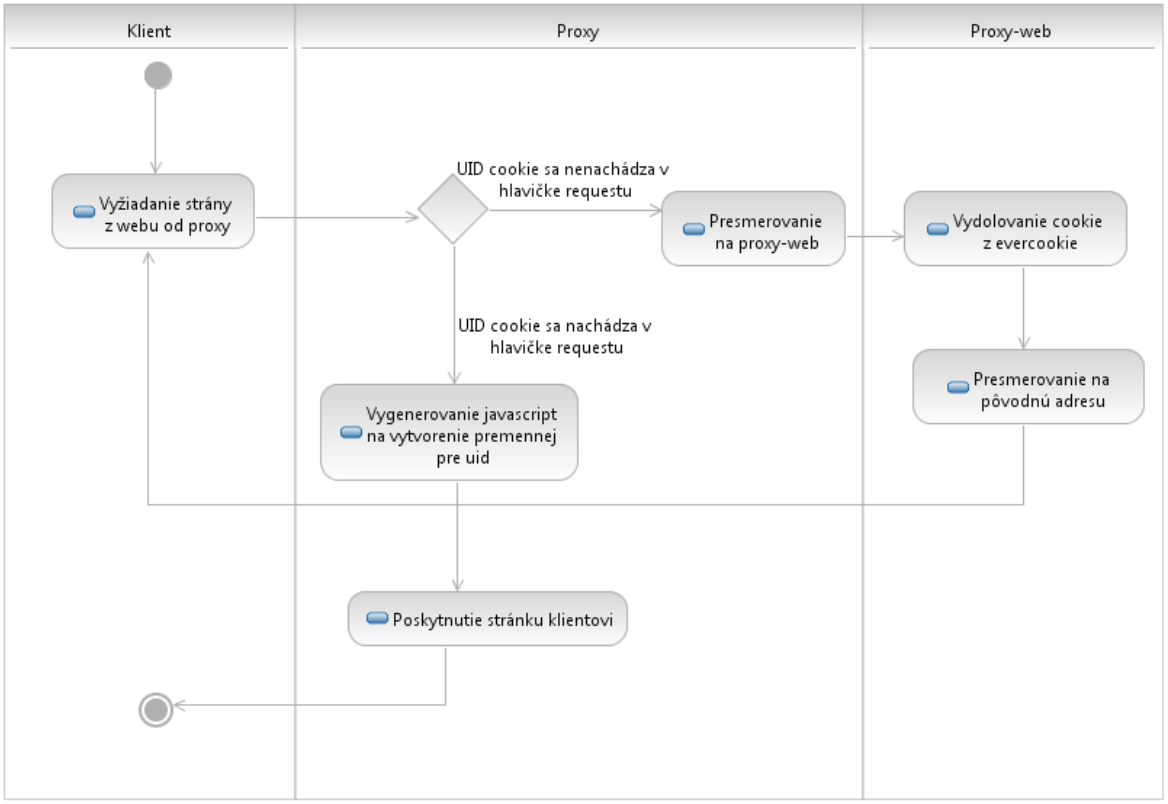
Riešenie bude pozostávať z doch častí:

1. vytvorenie pluginu do proxy, ktorý bude generovať vhodný javascript,
2. vytvorenie sekcie stránky na proxy-webe, ktorá sa postará o obnovenie cookie z evercookie

Do stránky sa bude vkladať kód na vloženie javascript súbora, ktorý bude vyzerať, že je umiestnený na proxy-webe. Browser k nemu teda pripojí hlavičky pre proxy-web. Ako ukazuje obrázok, proxy tento request odchytí, zistí, či sa v hlavičkách nachádza uid a ak áno, vygeneruje príkaz na nastavenie javascript premennej. Ak nie, vygeneruje príkaz na presmerovanie na podstránku proxy-webu, ktorá vykoná obnovenie cookie z evercookie a na záver presmeruje používateľa naspäť na web, ktorý chcel pôvodne navštíviť.

Implementácia

Funkcionalita na strane proxy je obsluhovaná triedou UIDFromCookiPlugin, ktorá dedí od triedy JavaScriptInjectingProcessingPlugin. Na strane proxy-webu je funkcionalita obnovenia cookie z evercookie implementovaná v conrolleri set_cookie_controller.rb a view cookie.html.rb.


Klient
Proxy
Proxy-web
Vyžiadanie stránky z webu od proxy
UID cookie sa nenachádza v hlavičke requestu
Presmerovanie na proxy-web
Vydolovanie cookie z evercookie
UID cookie sa nachádza v hlavičke requestu
Presmerovanie na pôvodnú adresu
Vygenerovanie javascript na vytvorenie premennej pre uid
Poskytnutie stránku klientovi

Logovanie času

Šprint: Eniac; *zodpovedná osoba:* Juraj Spusta

Analýza

Účelom tohto príbehu bolo znovunastavenie služby logovanie užívateľskej aktivity do databázy Proxy servera. Táto funkcionalita bola dočasne obmedzená výmenou pôvodných cookies na evercookies.

Počas analýzy bolo potrebné si naštudovať jednotlivé časti projektu a oboznámiť sa s fungovaním evercookies.

Implementácia a testovanie

Pri riešení úlohy bolo nutné nastaviť príslušné konfiguračné súbory a zabezpečiť posielanie potrebných dát na Proxy-web, aby sa následne zalogovali výsledky vrátené z JavaScriptu umiestneného na aktívnej stránke.

Prístup do logov

Šprint: Eniac; *zodpovedná osoba:* Ján Hudek

Analýza

V dôsledku cross-domain politiky nie je známe UID užívateľa pri odosielaní požiadavky prehliadačom. Je nutné zabezpečiť odoslanie ďalšej požiadavky obsahujúcej UID a následného zalogovania na strane proxy servera.

Návrh

Tento problém sa dá vyriešiť vložením javascriptovej funkcie do zdrojového kódu stránky, ktorý následne odošle ďalšiu požiadavku obsahujúcu UID užívateľa a checksum stránky. Tieto informácie sa budú odosielať nastavením premenných v metóde POST. Následne sa na strane proxy zaloguje prístup na stránku.

Riešenie bude rozdelené na dve časti:

- Vloženie javascriptu ktorý odošle naspäť potrebné informácie
- Odchytenie požiadavky obsahujúcej UID a následné zalogovanie

Implementácia a testovanie

Funkcionalita je zahrnutá v plugine UserAccessLoggingProcessingplugin dediacej od triedy RequestAndResponseProcessingPluginAdapter. Metóda processResponse sa zavolá pri spracovaní odpovede od servera vracajúceho vygenerovanú stránku. Do tejto stránky sa vloží javascript, ktorý následne odošle ďalšiu požiadavku na tento server na adresu userRecognitionRequest.html.

Táto požiadavka je zachytená na proxy metódou processRequest. Keďže požiadavka už obsahuje UID a checksum stránky obsiahnuté v premenných metódy POST, vloží sa do tabuľky access_logs záznam. Po zachytení tejto požiadavky sa ďalej neposúva ale vracia sa prehliadaču priamo odpoveď 200 OK. Toto je zabezpečené návratovou hodnotou RequestProcessingActions.FINAL_RESPONSE.

Odstraňovanie Evercookie

Šprint: Eniac; z*odpovedná osoba:* Michal Mrázik

Analýza

Proxy server si anonymne rozdeľuje používateľov na základe vygenerovaného čísla. Toto číslo je uložené v evercookie. Evercookie je knižnica, ktorá ako je z názvu jasné, ukladá informácie na strane užívateľa viacerými spôsobmi tak aby sa v prípade jej vymazania dokázali obnoviť.

Je potrbné zistiť, či sa evercookie nachádza v posielaných reqestoch. Ak áno, tak ju treba odstrániť. Druhou úlohou je odstraňovanie evercookie z počítača používateľa na základe jeho požiadavky tak, aby už neobsahoval informáciu o používateľovom id.

Návrh

1. Sledovať počas behu servera, či sa evercookie vkladá do requestov, ktoré sa posielajú ďalej. Ak sa nachádzajú, tak ich z requestov odstrániť.
2. Vymazať evercookie z počítača používateľa a to takým spôsobom, že pôvodná evercookie obsahújuca id používateľa sa nastaví na prázdny reťazec. Tu je potrebné spraviť zmeny na strane proxy webu. Konkrétne sú to tieto súbory: application_controller.rb, user_agent.en.html.erb a user_agent.sk.html.erb.

Implementácia a testovanie

1. Ku prvej časti úlohy sa zistilo, že sa evercookie nevkladá do requestov a tak nie je potrebné z nich nič vymazávať.
2. Ako prvé sa zmenili súboy, user_agent.en.html.erb a user_agent.sk.html.erb. Tu sa príkazom

   <a href="#" onclick="setEverCookie('='); <% @uid=nil %>return false" class="button handle">Remove evercookie from my computer

   Vytvoril link na stránke proxy servera, ktorý nastavuje hodnotu cookie na nil.

   Dalej bolo potrebné spraviť zmenu aj v application_controler.rb, konkrétne vo funkcii get_uid_from_cookie. Tu sa pôvodne vracala získaná hodnota z evercookie, ale po pridaní vymazávania môže touto hodnotou byť aj nil. Z tohto dôvodu testujeme a ak je hodnota cookieval nil, tak ju nahradíme prázdnym reťazcom.

   ```
   def get_uid_from_cookie
   - return request.cookies["__peweproxy_uid"]
   + cookieVal = request.cookies["__peweproxy_uid"]
   + return cookieVal== "" ? nil : cookieVal
   ```

Šprint 2 – MITS Altair 8800

Požiadavky

V tomto šprinte bolo potrebné implementovať tieto požiadavky:

- Servovanie skriptov a súborov priamo z proxy
- Štruktúrovanie zdrojákov pluginu

Návrh

Je potrebné navrhnúť novú adresárovú štruktúru – akýsi template, podľa ktorého sa budú vytvárať všetky nové bundle pluginov.

Adresárová štruktúra bude vyzerať nasledovne:

```
adaptive-proxy-core/
   Rakefile
   adaptive-proxy/
   jkey-extractor/
   plugins/
      plugin_name/
         offline/
         external_libs/
         migrations/
         plugins/
         scr/
         Rakefile
```

Používateľské príbehy

- Servovanie skriptov z proxy
- Spracovanie requestov z javascriptu priamo v proxy
- Offline tásky v offline adresári
- Folder pre migrácie
- Adresár pre konfigurácie pluginov
- Adresár pre statický obsah

- Adresár pre knižnice
- Adresár pre zdrojové kódy
- Spustenie tasku po deploymente

Úlohy

Bola definovaná aj jedna úloha mimo používateľských príbehov.

- Úprava skriptov pre deployment staging a production

Jednotlivé príbehy sú opísané v nasledujúcich podkapitolách.

Servovanie skriptov rovno z proxy

Šprint: MITS Altair 8800; z*odpovedná osoba:* Juraj Spusta

Analýza

Pôvodne sú všetky Javascripty umiestnené na Proxy-webe a v prípade ich pridávania do stránok smerujú linky na ne na Proxy-web. V rámci presúvania obsahu Webu do Proxy –servera bolo potrebné nájsť dané skripty a premiestniť ich.

Implementácia a testovanie

Pri implementácií úlohy boli presunuté nájdené JavaScripty. Zároveň boli pozmenené konfiguračné súbory jednotlivých pluginov, ktoré tieto skripty využívali.

Spracovanie requestov z javascriptov priamo v proxy

Šprint: MITS Altair 8800; z*odpovedná osoba:* Juraj Spusta

Analýza

Ďalším krokom pri preklápaní obsahu Proxy-webu na Proxy-server bolo vymädzenie funkcionality, ktorá sa by sa mohla vykonávať priamo na serveri a nie je potrebné, aby sa požiadavky preposielali na Web. Boli identifikované dva pluginy a to logovanie aktivity a extrakcia kľúčových slov.

Implementácia a testovanie

V rámci riešenia úlohy bol kód vykonávajúci logovanie aktivity (jazyk Ruby, projekt Web) presunutý na Proxy-server (jazyk Java, projekt Plugins). Pri implementovaní riešenia bol využitý pôvodný plugin pre preposielanie dát na Web.

Offline tasky v offline adresari

Šprint: MITS Altair 8800; *zodpovedná osoba:* Ivan Pleško

Analýza

Každý bundle pluginov bude ponúkať možnosť vytvoriť offline tasky - úlohy spúšťané CRONom. Môže sa jednať buď o Java aplikáciu alebo script. Rakefile skompiluje Javu (ak treba), skopíruje ju na vhodné miesto a nastaví CRON.

Návrh

Rakefile ponúkne 3 tasky:

- offline:build
  - prezrie všetky podadresáre adresára `offline` a skúsi ich skompilovať ako Javu
  - zo skompilovaných súborov vytvorí .jar súbory a umiestni ich dočasne do adresára `jar`, ktorý je podadresárom adresára `offline`
- offline:copy
  - presunie súbory z jar do adresára `offline`, ktorý je podadresárom hlavného proxy adresára
- offline:schedule
  - gem `whenever` nastaví CRON podľa súboru `schedule.rb`.

Implementácia

Adresár `offline` obsahuje súbor `Rakefile`, ktorý má v namespace `offline` 3 vyššie spomenuté metódy. Default task spustí všetky 3 tasky v poradí, v akom sú vymenované vyššie.

Folder pre migrácie

Šprint: MITS Altair 8800; *zodpovedná osoba:* Michal Mrázik

Analýza

Každý bundle pluginov bude ponúkať možnosť pre vytvorenie railsovych migracií databázováho modelu. Pri deploymente sa railsové migrácie samé pospúšťajú.

Návrh

V adresári migrations každého bundla pluginov sa budú nachádzať .rb skripty pre migráciu databázy, konfiguračný database.yml súbor a Rakefile súbor, ktorý pospúšťa všetky skripty.

Toto všetko je už vytvorené pre základný bundle pluginov ktorý sa nazýva adaptive-proxy-plugins-core. Je potrebné to len skopírovať, zmeniť cesty .rb súborov v Rakefile a otestovať v zmenenom adresári.

Keďže táto migrácia vytvára databázu proxy s množstvom nepotrebných tabuľiek, treba niektoré .rb skripty vymazať alebo upraviť.

Implementácia a testovanie

Po skopírovaní a zmeneí ciest ku .rb súborom nevznikol žiadny problém a databáza zmigrovala bez problémov.

Po upravení a odstránení niektorých .rb súborov ostávajú v databáze len potrebné tabuľky.

Adresár pre konfigurácie pluginov

Šprint: MITS Altair 8800; *zodpovedná osoba:* Ján Hudek

Analýza

Po vytvorení novej štruktúry pluginov vznikla pri nasadzovaní proxy potreba jedného adresára obsahujúceho všetky konfiguračné súbory pluginov. Tie by mali byť logicky rozdelené do adresárov podľa toho k akému plugin boundle prislúchajú.

Návrh

Každý vytvorený plugin bude obsahovať vlastný Rakefile. Po jeho zavolaní sa do adresára zahrnutého v nasadenom proxy, vytvorí nový adresár s menom plugin boundlu. Do neho sa následne skopírujú všetky konfiguraćné xml súbory.

Implementácia a testovanie

Potrebnú funkcionalitu obsahuje metóda copyConfigFiles. Jej úlohou je vytvorenie adresára plugins v nasadzovanom proxy, ak neexistuje, vytvore adresára s menom plugin boundle a v ďalšom kroku volaním FileUtils.cp_r, rekurzívne skopírovať všetky konfiguračné súbory.

Adresár pre statický obsah

Šprint: MITS Altair 8800; *zodpovedná osoba:* Ján Hudek

Analýza

Po vytvorení novej štruktúry pluginov vznikla pri nasadzovaní proxy potreba jedného adresára obsahujúceho všetok statický obsah pluginov. Ten by mal byť logicky rozdelený do adresárov podľa toho k akému plugin boundle prislúcha.

Návrh

Každý vytvorený plugin bude obsahovať vlastný Rakefile. Po jeho zavolaní sa do adresára zahrnutého v nasadenom proxy, vytvorí nový adresár s menom plugin boundlu. Do neho sa následne skopíruje všetok statický obsah.

Implementácia a testovanie

Potrebnú funkcionalitu obsahuje metóda copyStaticContent. Jej úlohou je vytvorenie adresára static v nasadzovanom proxy, ak neexistuje, vytvore adresára s menom plugin boundle a v ďalšom kroku volaním FileUtils.cp_r, rekurzívne skopírovať všetok statický obsah.

Adresár pre libky

Šprint: MITS Altair 8800; z*odpovedná osoba:* Michal Mrázik

Analýza

Každý bundle pluginov bude adresár external_libs do ktorého si tvorca pluginu vloží externé knižnice (.jar). Pri deploymente sa sa tieto knižnice skopírujú do jednotného adresára pre všetky pluginy.

Návrh

Vytvoriť rakefile, ktorý na základe zadaného miesta skopíruje všetky externé knižnice pluginov do jednotného adresára. Pri tejto úlohe si pomôcť knižnicou fileutils ,ktorá sa dá nainštalovať ako gem.

Implementácia a testovanie

Vznikol rakefile s nasledovnou funkciou copy, pričom target je adresár do ktorého sa bude kopírovať.

```
require 'fileutils'
require 'rubygems'
require 'rake'

desc "copy"
task :copy do
  target = "../../libs/"
  FileList['**/*.jar'].each do |source|
    file_name = File.basename(source)
    target_dir = target + File.new(source).path.sub(file_name, '')
    makedirs target_dir
    cp_r source, target_dir + file_name, :verbose => true
  end
end

task :default => ["copy"]
```

Adresár pre zdrojové súbory

Šprint: MITS Altair 8800; z*odpovedná osoba:* Michal Valluš

Analýza

Vo vytváranej adresárovej štruktúre pre plugin bude adresár src, ktorý bude obsahovať všetky zdrojové kódy. V lokálnom Rakefile pre plugin vzniknú tasky, ktoré budú zabezpečovať zbuildovanie zdrojových kódov a následné uloženie do jar súboru.

V hlavnom Rakefile je potrebné zbuildovať najprv jkey-extractor, ktorého jar súbor je potrebné skopírovať do adresára external_libs v štruktúre súčasných pluginov. Následne je potrebné zbuildovať Proxy a všetky bundle pluginov, ktoré sa nachádzajú v adresári plugins.

Návrh

V lokálno Rakefile budú vytvorené tásky :build a :jar. Tieto budú zahrnuté do default tásku, takže pri spustení skriptu sa zdrojové súbory skompilujú. Výsledný jar súbor sa bude volať rovnako ako adresár, v ktorom bol skript spustený.

Implementácia a testovanie

Navrhované tásky boli implementované v jazyku Ruby. Pri testovaní boli problémy so spustením kompilácie proxy servera, nakoľko príkazový riadok bol príliž dlhý a v prostredí Windows skript hlásil chybu. Tento task preto bolo potrebné otestovať pod Linuxom, kde spomínané problémy nenastali.

Spustenie tasku po deploymente

Šprint: Mits Altair 8800; z*odpovedná osoba:* Pavol Sokol

Analýza

V súčasnom systéme sú zásuvné moduly vytvárané do jadra modulov, čo v prípade nasadenia pre širokú komunitu vývojárov nie je vhodné. Je potrebné zabezpečiť, aby ďalšie vytvárané moduly, boli od seba oddelené. Navrhnuté riešenie musí obsluhovať aj jednoduché nasadenie tohto systému. V tíme bola vytvorená šablóna pre moduly. Táto šablóna a existujúce zásuvné moduly musia spĺňať rozhranie pre vytvorenie úloh vykonaných po nasadení.

Návrh

V tíme bol dohodnutý formát hlavného a ďalších súborov vykonávajúcich nasadenie systému. V rámci tohto konkrétneho zadania je deklarovaná špecifiká úloha, ktorú si môže nadefinovať vývojár zásuvného modulu a bude vykonaná po nasadení systému. Volanie všetkých úloh prebieha z jedného miesta – súboru.

Implementácia a testovanie

Implementácia znamená vytvorenie úlohy after_deploy v rake súbore modulov a zabezpečenie jeho volania po nasadení vykonanom utilitou capistrano s rozšírením multistage, aby sa zabezpečilo jednoduché nasadenie na „ostrý“ aj testovací server.

Šprint 3 – Fairchild Illiac-IV

Do 3. šprintu boli zahrnuté nasledovné požiadavky:

- Plugin pre messageboard nad stránkou
- Adaptive proxy web v Rails 3
- Set evercookie ako cookie pre každú doménu
- API pre tvorcov pluginov nad vyhľadávačmi

Tieto požiadavky boli následne rozdelené do používateľských príbehov (user-story):

- Messageboard nad stránkou
- Proxy-web v Rails 3
- Evercookie pre všetky domény
- SearchEngineService a získavanie výsledkov vyhľadávania z Google
- SearchEngineService a menenie výsledkov vyhľadávania z Google
- Plugin pre vytváranie DOM webovej stránky

Dokumentácie k jednotlivým user-story sa nachádzajú v nasledujúcich podkapitolách. Niektoré user-story neboli v tomto šprinte riadne dokončené, preto sa tu ani nenachádza dokumentácia k nim.

Messageboard nad stránkou

Šprint: Fairchild Illiac-IV; *zodpovedná osoba:* Ivan Pleško

Špecifikácia

Proxy server ako prostredník medzi klientom a serverom obohatí stránku o diskusiu k stránke. Diskusia bude viazaná na URL. Používateľ bude prispievať pod menom, ktoré si zvolí. Príspevky budú radené chronologicky. Je pravdepodobné, že príspevkov bude mnoho, preto je požadovaná implementácia listovania medzi stránkami. Ak používateľ otvorí okno s diskusiou, bude otvorené aj na inej url. Ak ho zavrie, ostane skryté aj na iných url. Správy sa budú načítavať každých 8 sekúnd.

Návrh

Riešenie spočíva v úprave HTML kódu stránky na strane servera. Upravený zdrojový kód obsahujúci rozhranie na prezeranie a pridávanie príspevkov, úpravu používateľského mena a listovanie sa odošle k používateľovi.

Zobrazovanie iných strán diskusie, pridávanie príspevkov aj editácia používateľského mena bude realizovaná ako asynchrónne požiadavky na server (AJAX).

Myšlienku komunikácie zobrazuje obrázok 1.

Implementácia

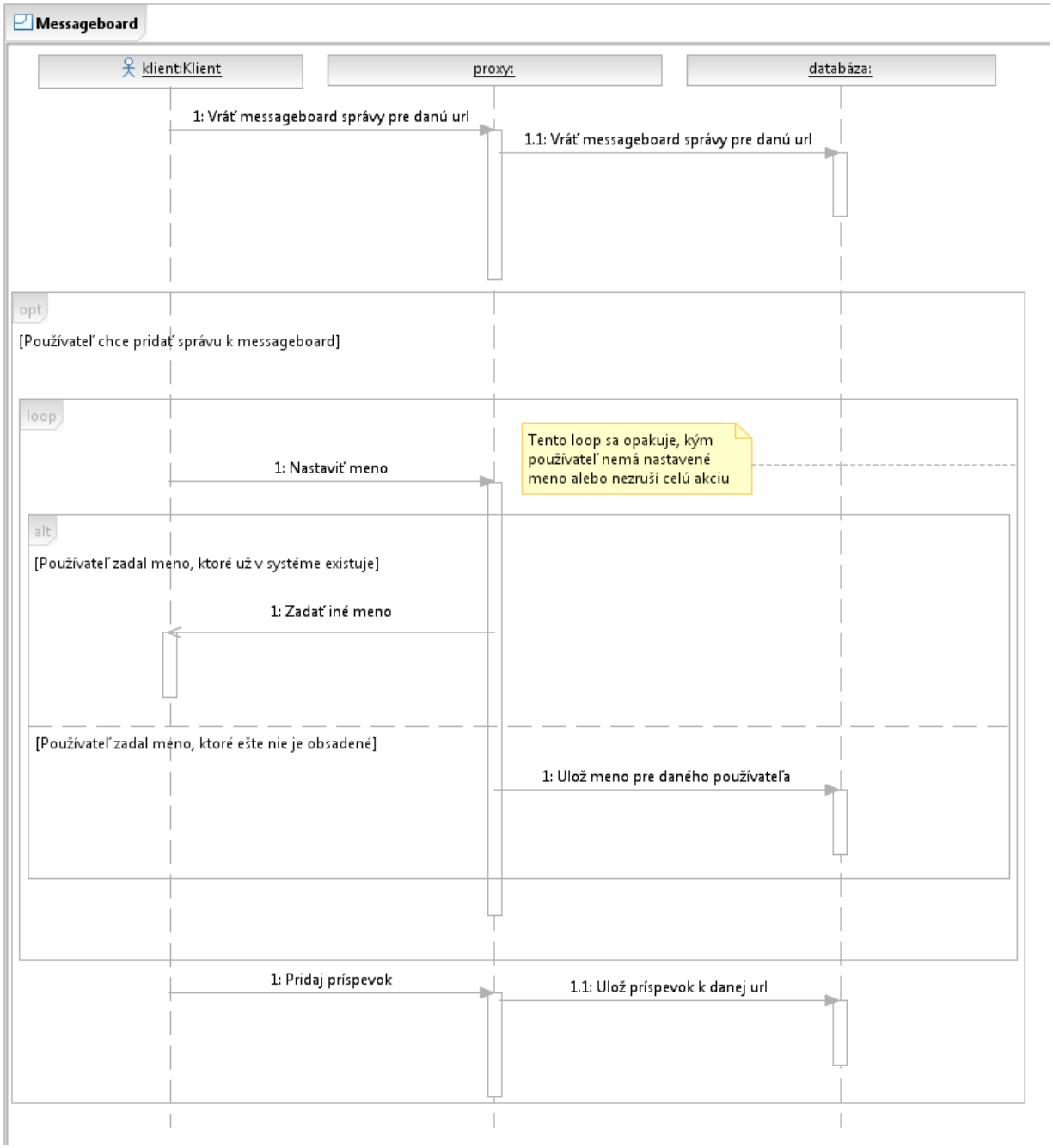
Funkcionalitu zabezpečuje na strane servera MessageboardComunicationProcessingPlugin a na strane klienta súbor peweproxy_messageboard.js (adresár static bundla pluginov messageboard).

Dopyt na server sa vykonáva odoslaním POST requestu na http://{DOMÉNA}/adaptive-proxy/messageboard_call.html. Toto volanie sa odchytí na proxy serveri a v odpovedi sa vrátia požadované dáta, prípadne informácia o úspešnosti akcie. Volanie metódy sa odovzdáva parametrom action.

Server poskytuje nasledovné metódy:

- setMessageboardNick – zmena používateľského mena
  - parametre: nick, uid
  - odpoveď: OK pri úspechu, NICK_EXISTS ak používateľské meno existuje, FAIL ak sa zmena mena nepodarila
- getMessages – vráti správy
  - parametre: from – od ktorého príspevku začať, count – koľko príspevkov zobraziť, decorateLinks (1/0) – obaliť linky do príslušných <a> tagov?
  - odpoveď: JSON: {messages:[{text,nick,time,},...],total}

- setShown – nastaví zobrazenie okna pre daného používateľa
  - parametre: shown (1/0), uid
  - odpoveď: OK/FAIL
- addMessage
  - parametre: text, uid
  - odpoveď: OK/FAIL
- getMessageCount
  - parametre: žiadne
  - odpoveď: počet správ (číslo)
- getUserPreferences
  - parametre: uid
  - odpoveď: JSON: {messageboard_nick, language, visibility}



Obr. 1: Sekvenčný diagram opisujúci prácu messageboardu.

SearchEngineService a získavanie výsledkov vyhľadávania z Google

Šprint: Fairchild Illiac IV; z*odpovedná osoba:* Michal Mrázik

Analýza

Je potrebné aby tvorcovia pluginov mali možnosť vyžiadať si SearchEngineService, ktorý im zatiaľ nad google.com poskytne nasledovnú funkcionalitu: vráti Array s výsledkami + query string, ktorý k týmto výsledkom viedol s tým, že k výsledku budú mať nasledovné údaje:

- URL
- nadpis
- perex
- poradie

Návrh

1. V package sk.fiit.rabbit.adaptiveproxy.plugins.service definitions vytvoriť interface, ktorý bude implementovať ProxyService.
2. Ďalej v package sk.fiit.rabbit.adaptiveproxy.plugins.sevices vivorit package search. Tu sa budú vytvárať triedy, ktoré vykonávajú danú funkcionalitu. Tieto triedy sa delia na Providery a moduly. Modul na základe service vykoná požadované operácie. V nažom prípade je to ArrayList s v ktorom sa bude nachádzať informácia o danom vyhľadávaní na google.

Implementácia a testovanie

1. Vytvorenie abstraktnej triedy SearchResultService, ktorá v implementuje Proxy Service a obsahuje tieto ľ metódy: public **ArrayList<SearchResultObject>** getData() a metóodu **public String getQueryString.**
2. Ďalej sa vytvoria triedy **GoogleSeachResultModule**, ktorá slúži na spúšťanie jednotlivých providerov a **GoogleSearchResult Provider**, ktorý získava požadované informácie od služby dom.

Implemenácia bola testovaná sériou náhodných vyhľadávaní.

SearchEngineService a menenie výsledkov vyhľadávania z Google

Šprint: Fairchild Illiac-IV; *zodpovedná osoba:* Michal Valluš

Analýza

V rámci analýzy bolo potrebné v prvom rade analyzovať mechanizmus pluginov a služieb, ktorý je implementovaný v adaptívnom proxy serveri. Nakoľko existuje viacero implementácií rôznych iných pluginov, táto analýza bola pomerne efektívna.

Ďalej bola vykonaná analýza štruktúry HTML kódu, ktorý vracia vyhľadávač Google. Je potrebné zakázať v prehliadači Javascripty, pretože inak vyhľadávač nevráti čisté html, ale údaje posiela pomocou Javascriptu, čo je následne problém parsovať.

Výsledky z vyhľadávača Google sú umiestnené v štruktúre spôsobom, ktorý zodpovedá nasledujúcemu vyjadreniu v jazyku XPath:

```
/html/body/div[@id='cnt']/div[@id='nr_container']/div[@id='center_col'
]/div[@id='res']/div[@id='ires']/ol
```

Návrh

Je potrebné vytvoriť službu, ktorá bude schopná poskytnúť požadovanú funkcionalitu.

Do štruktúry adaptívneho proxy servera bol navrhnutý modul, ktorý bude rozhodovať o tom, aký provider na prácu s výsledkami vyhľadávania poskytne pluginu. Vzhľadom na to, že je predpoklad implementovať v budúcnosti takúto funkcionalitu aj pre ďalšie vyhľadávače, tento prístup sa javí ako celkom efektívny aj z hľadiska rozšíriteľnosti kódu.

Implementovaná služba bude poskytovať štyri základné operácie:

- put()
- delete()
- swap()
- move()

Implementácia a testovanie

Podľa návrhu bola implementovaná služba aj jej provider. Class diagram riešenia je zobrazený spolu so službami pre ďalšie dva vyhľadávače v nasledujúcom šprinte.

Pri testovaní sa vyskytli problémy s prezentačnou vrstvou, konkrétne s parsovaním DOM do HTML.

Šprint 4 – Comodore

Do 4. šprintu boli zahrnuté nasledovné požiadavky:

- Prezeranie metadát k aktuálnej stránke
- API pre tvorcov pluginov nad vyhľadávačmi
- Proxy používa Metall
- Nahradenie checksum za ID requestu
- Load-balancing

Tieto požiadavky boli následne rozdelené do používateľských príbehov (user-story):

- Prezeranie metadát k aktuálnej stránke
- Pridávanie metadát k aktuálne zobrazenej stránke
- Mazanie metadát k stránke
- SearchEngineService pre získanie výsledkov z Yahoo a Bing
- SearchEngineService pre zmenu výsledkov z Yahoo a Bing
- Proxy používa Metall
- Nahradenie checksum za request ID
- Load-balancer pre proxy

V tomto šprinte sa riešili aj niektoré nedokončené user-story z predchádzajúceho šprintu. Takisto, nie všetky user-story z tohto šprintu boli dokončené, preto sa tu nenachádzajú ani dokumentácie k nim.

Prezeranie metadát k aktuálnej stránke

Šprint: Commodore; z*odpovedná osoba:* Ján Hudek

Analýza

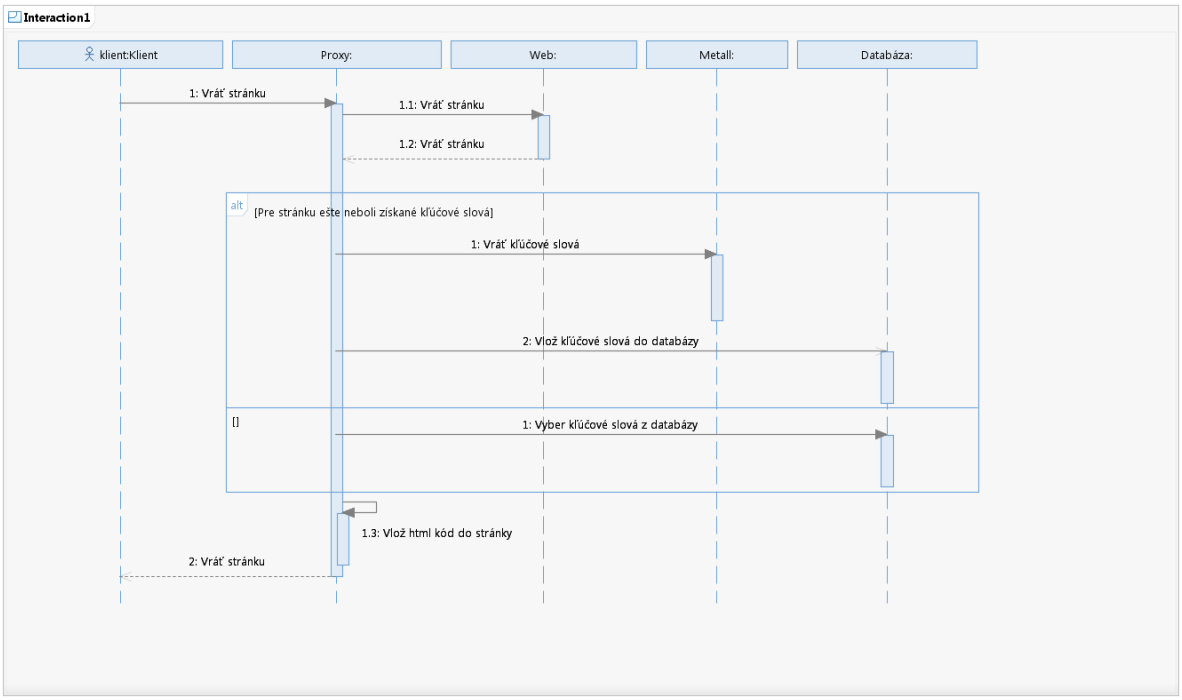
Pre každú navštívenú stránku adaptívny proxy server generuje kľúčové slová. Tieto slová sú v podobe metadát k stránke ukladané do databázy. Cieľom je vytvoriť rozhranie medzi užívateľom a serverom, ktoré by umožňovalo používateľovi editovať, pridávať alebo mazať kľúčové slová.

Toto rozhranie by bolo zobrazené priamo u klienta pri navštívenej stránke. Obsahovalo by zoznam kľúčových slov k danej stránke a ovládacie prvky určené k editácii. Rozhranie by sa nemalo zobrazovať na každej stránke ale pri voľbe používateľa.

Návrh

Riešenie spočíva v úprave HTML kódu stránku na strane servera. Upravený zdrojový kód obsahujúci rozhranie na zobrazovanie a editáciu kľúčových slov sa odošle k používateľovi (priebeh komunikácie je zobrazený v sekvenčnom diagrame na obrázku č.1). Rozhranie by okrem zobrazovania kľúčových slov malo obsahovať nasledovnú funkcionalitu:

- pridanie kľúčového slova
- zmazanie kľúčového slova
- úprava kľúčového slova



Obr. 1: Zobrazenie rozhrania prezerania a editácie kľúčových slov

Implementácia a testovanie

Funkcionalitu zabezpečuje KeyWordsProcessingPlugin. Jeho funkcionalita sa dá rozdeliť na dve časti:

- vloženie HTML, CSS a JavaScript kódu rozhrania do kódu stránky
- odpovedanie na požiadavky od používateľa

Dopyt na server sa vykonáva volaním súboru adaptive-proxy/key_words_call.html. Toto volanie sa odchytí na proxy serveri a v odpovedi sa vrátia požadované dáta, prípadne informácia o úspešnosti akcie. Volanie metódy sa odovzdáva parametrom action.

Server poskytuje nasledovné metódy:

- getKeyWords
- editKeyWord
- removeKeyWord
- addKeyWord

SearchEngineService a získavanie vvýsledkov vyhľadávania z Yahoo a Bing

Šprint: Commodore; z*odpovedná osoba:* Michal Mrázik

Analýza

Je potrebné aby tvorcovia pluginov mali možnosť vyžiadať si SearchEngineService, ktorý im poskytne naslednovnú funkcionalitu aj nad ďalšími vyhľadávačmi konkrétne yahoo a bing: vráti Array s výsledkami + query string, ktorý k týmto výsledkom viedol s tým, že k výsledku budú mať nasledovné údaje:

- URL
- nadpis
- perex
- poradie

Návrh

1. V package sk.fiit.rabbit.adaptiveproxy.plugins.service definitions už je vytvorený interface, ktorý bude implementuje ProxyService.
2. Ďalej v package sk.fiit.rabbit.adaptiveproxy.plugins.sevices vivorit package search. Tu sa budú vytvárať triedy, ktoré vykonávajú danú funkcionalitu. Tieto triedy sa delia na Providery a moduly. Modul na základe service vykoná požadované operácie. V nažom prípade je to ArrayList s v ktorom sa bude nachádzať informácia o danom vyhľadávaní na na yahoo alebo bing vyhľadávači.

Implementácia a testovanie

1. Vytvorenie abstraktnej triedy SearchResultService, ktorá v implementuje Proxy Service a obsahuje tieto ľ metódy: public **ArrayList<SearchResultObject>** getData() a metóodu **public String getQueryString.**
2. Ďalej sa vytvoria triedy **YahooSeachResultModule a BingSeachResultModule**, ktoré slúži na spúšťanie jednotlivých providerov a **YahooSearchResult Provider a BingSearchResult Provider**, ktoré získavajú požadované informácie od služby dom.

Implemenácie boli testovaná sériou náhodných vyhľadávaní.

SearchEngineService a menenie výsledkov vyhľadávania z Google

Analýza

V tomto prípade bola už štruktúra prostredia adaptívneho proxy servera pomerne jasná. Bolo potrebné už iba analyzovať štruktúru HTML kódu, ktorý vracajú jednotlivé vyhľadávače Yahoo a Bing. Výsledky sa teda nachádzajú v takejto štruktúre (XPath notácia):

Yahoo:

```
/html/body/div[@id='doc']/div[@role='document']/div[@id='results']/div
[@id='cols']/div[@id='left']/div[@id='main']/div[@id='web']/ol
```

Bing:

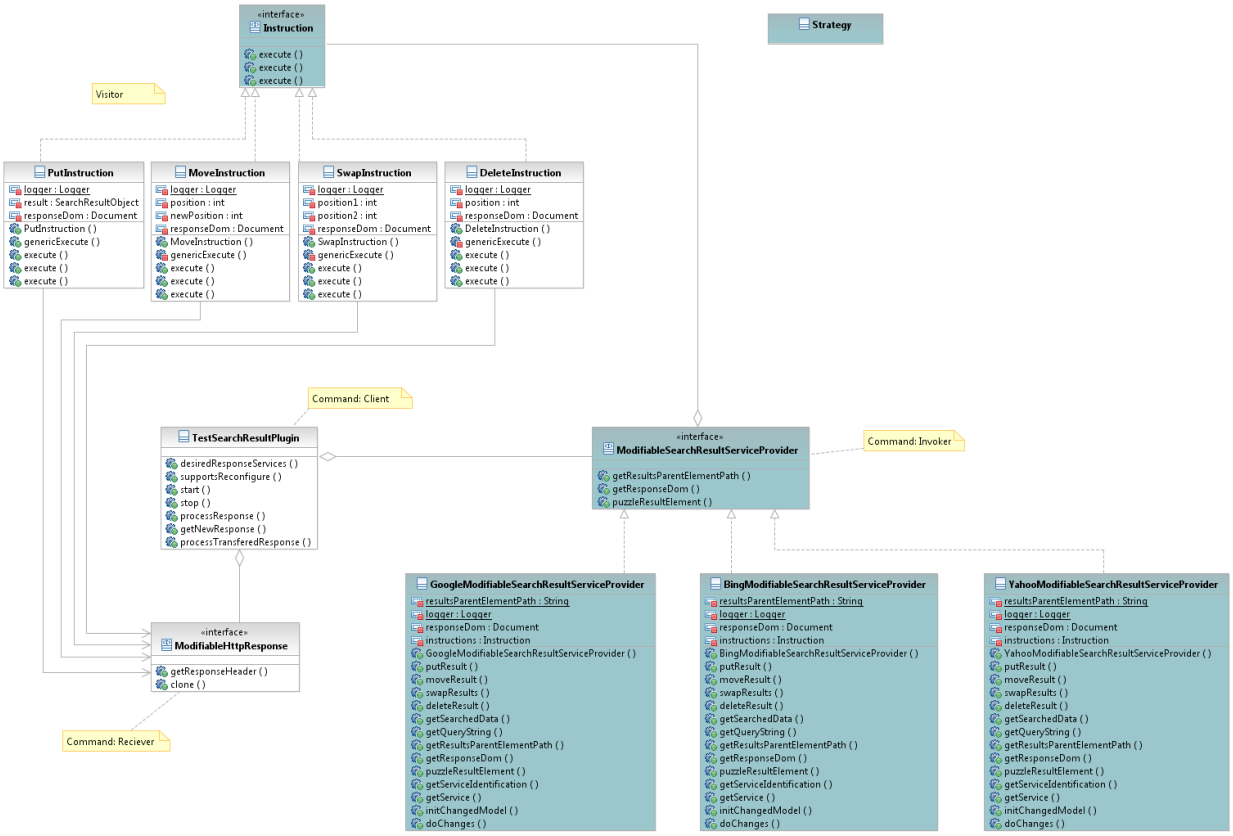
```
/html/body/div[@id='sw_page']/div[@id='sw_width']/div[@id='sw_content'
]/div/div[@id='sw_canvas']/div[@id='sw_main']/div[@id='content']/div[@
id='results_area']/div[@id='results_container']/div[@id='results']/ul
```

Návrh

Pretože pribudli dva nové providre služby, potrebovali sme implementovať double-dispach na provider a na inštrukciu. To sa nám podarilo využitím návrhových vzorov strategy, command a visitor.

Implementácia a testovanie

Riešenie bolo implementované podľa spomínaného návrhu. Pri testovaní sa ukázali minimálne nedostatky v prezentačnej vrstve.



Obr. 1: Class diagram riešenia so zvýraznenými vzormi

Proxy používa Metall

Šprint: Commodore; *zodpovedná osoba:* Ján Hudek

Analýza

Adaptívny proxy využíva funkcionalitu z projektu JKey-extractor na získavanie čistého textu z webu, a tiež na získavanie kľúčových slov. Na základe požiadaviek sa táto funkcionalita presunie na web servis Metall.

Návrh

Jkey-extractor sa používa v dvoch moduloch: ReadabilityCleartextExtractionServiseModul a CachingPageInformationProviderServiceModul. Existujúcu funkcionalitu je potrebné upraviť tak, aby prebehlo volanie webovej služby a ostalo zachované rozhranie. Pre získanie čistého textu sa volá služba readability a pre získanie kľúčových slov služba meta. Obe služby dostávajú na vstupe obsah web stránky, pre ktorú sa má služba vykonať.

Modul pre vytváranie DOM webovej stránky

Šprint: Commodore; z*odpovedná osoba: Pavol Sokol*

Analýza

Vhodným nástrojom pre vývojárov nástrojov v adaptívnom proxy je získanie web stránky ako DOM. Tým sa umožní jednoduchšia manipulácia pri tvorbe personalizovaného webu. Pri tvorbe modulu treba vybrať vhodný DOM model a hlavne parser HTML, ktorý umožní načítanie aj nevalidných stránok.

Návrh

Je potrebné otestovať viacero HTML parserov existujúcich na platforme Java. Hlavne Mozilla parser a HtmlCleaner, ktoré sa podľa dokumentácie javia ako najvhodnejšie. Následne je potrebné určiť DOM model, v ktorom budú HTML stránky reprezentované. Medzi najpoužívanejšie patria DOM, JDOM a DOM4J. Následne je potrebné vytvoriť službu ako zásuvný modul pre adaptívny proxy a funkcionalitu implementovať. Služba poskytuje rozhranie getHTMLDom pre získanie DOM modelu a setHTMLDom s DOM objektom ako parameter služby pre vrátenie zmeneného modelu klientovi.

Implementácia a testovanie

Služba dostane na vstupe HTML stránku v textovom formáte. HTMLCleaner upravý dokument podľa nastavení a serializuje ho do objektového JDOM modelu. V tomto formáte prebiehajú zmeny nad dokumentom. Výstup je spätne formátovaný do reťazca cez XMLOutputter. Zmenený dokument je vrátený klientovi cez API adaptívneho proxy. Test prebehol vytvorením a spustením ďalších služieb využívajúcich objektový model HTML dokumentu.

Nahradenie checksum za requestID

Šprint: Commodore; z*odpovedná osoba:* Juraj Spusta

Analýza

Pri zaznamenávaní aktivity na stránkach sa vytvárajú záznamy v databáze, ktoré sa vždy viažu na používateľa a danú URL. Väzba na stránku sa ale zisťuje zbytočne zložito preto je potrebné vytvoriť jednoduchší spôsob.

Implementácia a testovanie

Pri riešení tejto úlohy bola upravená tabuľka access_logs, v ktorej bol zmenený primárny kľúč z integeru (autoincrement) na String. Nové ID sa totižto vygeneruje pri načítavaní stránky z proxy a zároveň sa aj zapíše do stránky ako javascript premenná. Jednotlivé aktivity sa potom logujú len na základe tohto identifikátora.

Evercookie pre všetky domény

Šprint: Commodore; z*odpovedná osoba:* Juraj Spusta

Analýza

Pri monitorovaní aktivity používateľov na stránkach, ale aj v iných prípadoch je nutné identifikovať používateľa. To sa deje na základe javascript-ovej premennej, v ktorej je uložený identifikátor. Ten je však nastavený pre doménu peweproxy. Požiadavkou na zmenu bolo nastavenie identifikátora pre všetky domény osobitne.

Implementácia a testovanie

Pri implementácií bolo potrebné pridať premennú pre každú stránku, ktorú vracia proxy. Táto premenná obsahuje požadovaný identifikátor, ktorému sa zároveň nastaví dostatočne dlhá doba expirácie. Súčasne s pridávaním bolo zabezpečené aj odstraňovanie identifikátora pri komunikácií medzi proxy serverom a Internetom.

Šprint 5 – ZX Spectrum+

Do 5. šprintu boli zahrnuté nasledovné požiadavky:

- Refactoring, prerobenie messageboard + keywords
- Servis na získavanie posielaných dát z klienta
- Fixnúť deploy

Tieto požiadavky boli následne rozdelené do používateľských príbehov (user-story):

- Refactoring, prerobenie messageboard + keywords
- Servis na získavanie posielaných dát z klienta
- Kompilácia bundlov podľa poradia
- Dostať deploy do funkčného stavu

Refactoring, prerobenie messageboard + keywords

Šprint: ZX Spectrum+; z*odpovedná osoba:* Ján Hudek

Analýza

Funkcionality messageboard a keywords využívajú v okne užívateľa to isté rozhranie. Na zobrazenej webovej stránke sa zobrazí menu, kde je možné vyvolať okno messageboard alebo keywords. Messageboard a keywords sa preto nachádzajú v jednom plugin bundli. Aj keď sú tieto funkcionality podobné, bolo by vhodnejšie aby mala každá vlastný bundel. Tento stav je zapríčinený tým, že neexistuje služba ktorá by umožňovala dvom bundlom upravovať ten istý kúsok HTML kódu vloženého do stránky.

Návrh

Riešenie spočíva v rozdelení messageboardu a keywords tak, aby každá táto funkcionalita mohla existovať vo vlastnom bundli. Je teda nutné vytvoriť službu ktorá by umožňovala pridať položku do menu. Funkcionalita obsiahnutá v samostatnom bundli by teda využívala túto službu.

Implementácia a testovanie

Funkcionalita bola rozdelená do dvoch samostatných bundlov.

- Adaptive-proxy-bundle-messageboard
- Adaptive-proxy-bundle-keywords

Oba tieto bundle využívajú službu klientMenuService. Tá umožňuje vložiť do menu nové tlačidlo. Po jeho stlačení sa zavolá javascriptová funkcia vložená už pluginom z bundlu. Toto umožňuje rozširovať funkcionalitu používateľského menu bez nutnosti upravovať už existujúci kód.

Servis na získavanie posielaných dát z klienta

Šprint: ZX Spectrum+; z*odpovedná osoba:* Juraj Spusta

Analýza

V rámci predchádzaniu duplikácie kódu je potrebné vytvoriť nový servis, ktorý bude získavať poslané dáta z klienta na proxy.

Implementácia a testovanie

Bol vytvorený nový servis modul, ktorý obsahuje danú funkciu na získavanie dát. Modul bol zaradený medzi ostatné dostupné moduly na proxy.

Celkový pohľad na riešenie po zimnom semestri

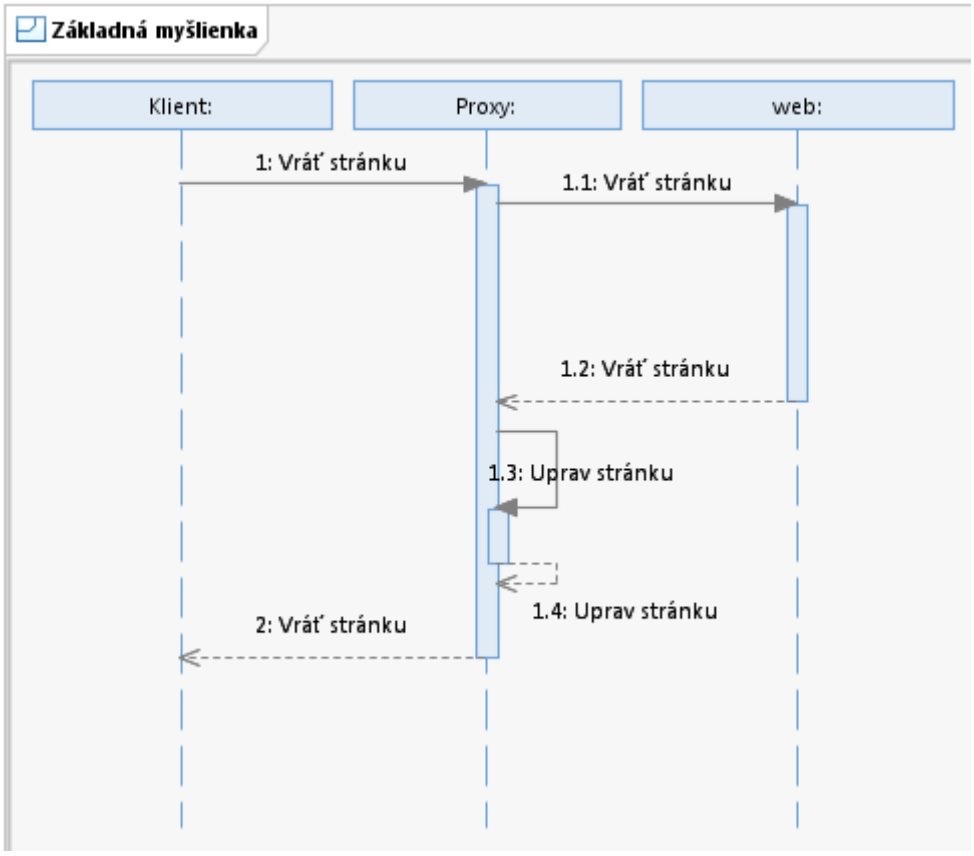
Adaptívny proxy server

Web, ako ho dnes poznáme, je veľký a anonymný. Množstvo ponúkaného obsahu neustále rastie závratným tempom a je čoraz ťažšie sa na Webe navigovať efektívne. Naopak, bežným javom je preťaženie informáciami (angl. information overload) a stratenie sa v hyperpriestore (angl. lost in hyperspace). To je spôsobené aj tým, že každý používateľ je iný, má iné záujmy, iné informačné potreby. Napriek tomu je ale všetkým ľuďom predkladaný rovnaký obsah.

Ak by sme napríklad mali nástroj, ktorý by vedel, že konkrétny človek sa zaujíma o autá, mohli by sme mu po zadaní slova „jaguar“ do vyhľadávača jeho dopyt personalizovať a doplniť výsledky vyhľadávania tak, aby sa mu na prvých miestach zobrazili autá značky Jaguar a nie šelma, operačný systém či parfum s týmto pomenovaním.

Hlavný problém súčasného webu teda vidíme v absentujúcej personalizácii.

Na Fakulte informatiky a informačných technológií vznikol zaujímavý projekt adaptívneho proxy servera, ktorý dokáže modelovať záujmy používateľa a personalizovať na ich základe surfovanie na „divokom“ webe. Tento server je prostredník medzi používateľom a webom tak, ako zobrazuje obrázok 1. Takýto server otvára používateľovi nové možnosti, súvisiace najmä so sociálne založenými odporúčaniami a vylepšeniami.



Obr. 1: Základná myšlienka práce proxy servera.

Naša práca

Náš tím dostal k dispozícii tento server, ktorý dokázal identifikovať používateľa a zaznamenávať používateľovu aktivitu. My sme ho v priebehu semestra obohatili o zaujímavé funkcie. Každý web pomocou neho obohacujeme o diskusné fórum. Dokážeme prispôsobiť výsledky vyhľadávania na známych vyhľadávačoch. Môžeme zobraziť kľúčové slová danej stránky a používateľom ponúknuť možnosť úpravy za účelom lepšieho zaradenia obsahu. Zjednodušili sme systém na identifikáciu používateľa. Celý systém vieme vďaka skriptom, na ktorých sme tiež pracovali, nasadiť jedným príkazom. Náš prínos pre proxy server popíšeme na nasledujúcich stranách.

Zjednodušenie identifikácie používateľa

Proxy server v stave, v ktorom sme ho dostali, požadoval od používateľov, aby si pomerne pracne nastavovali uid (jedinečný používateľský identifikátor) do user-agenta prehliadača. My sme vytvorili jednoduchý spôsob identifikácie, založený na technológii perzistentných cookie. Jediné, čo používateľ

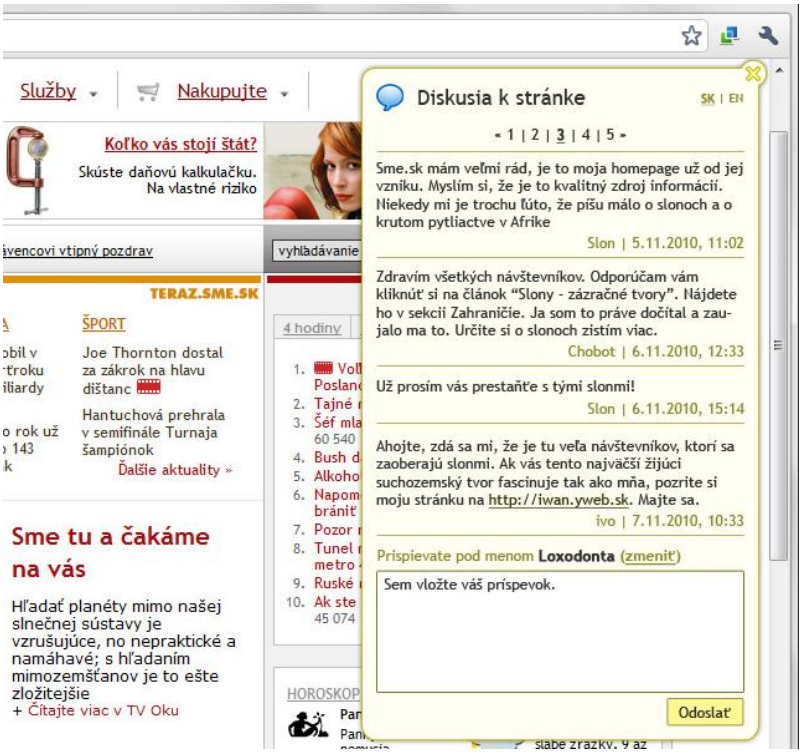
musí spraviť pre používanie proxy servera, je kliknutie na tlačidlo, ktorým si u seba nastaví cookie, ktorou ho proxy bude vedieť identifikovať.

Štruktúrovanie zdrojových súborov

Proxy server sme pre vývojárov upravili tak, aby doň mohli ľahko pridávať novú funkcionalitu. Môžu sa odraziť od šablóny, ktorú sme vytvorili. Ak dodrží jednoduché zásady, skripty, ktoré sa starajú o nasadenie proxy servera do prevádzky, sa už postarajú o to, aby nová funkcionalita bola správne spustená.

Diskusia nad stránkou

Používateľ surfujúci cez náš proxy server dostane možnosť vyjadriť sa ku každej stránke na internete. A to vďaka diskusii, ktorú mu proxy poskytne. Náhľad môžeme vidieť na obrázku 2.



Obr. 2: Diskusia k stránke

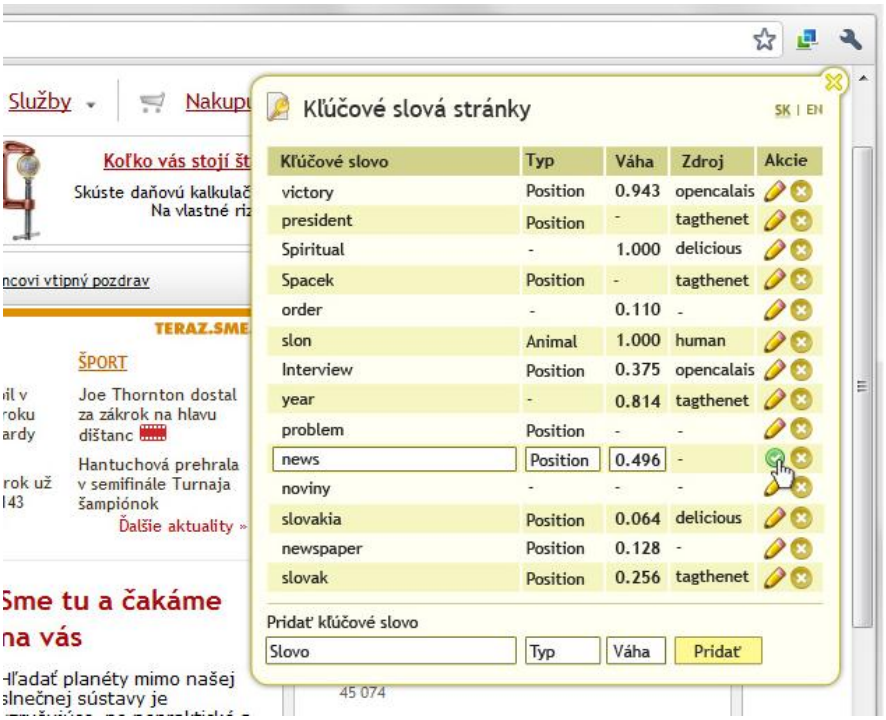
Práca s výsledkami vyhľadávania

Vytvorili sme programové rozhranie pre manipuláciu s výsledkami najpopulárnejších webových vyhľadávačov. Tým sme odbremenili tvorcov zásuvných modulov od implementačných detailov jednotlivých vyhľadávačov. Zabezpečili sme jednotné rozhranie pre zapájanie rôznych metód

operujúcich nad výsledkami vyhľadávania na webe. Implementovali sme manipuláciu s výsledkami nad Google, Yahoo! a Bing.

Zobrazenie a úprava kľúčových slov pre stránku

Proxy server dokáže vďaka rôznym webovým službám extrahovať zo stránky kľúčové slová. Tieto slová má uložené v databáze a dokáže ich používateľovi zobraziť. Ba čo viac, používateľ ich dokáže upraviť, mazať, či pridávať. Takouto činnosťou pomáhajú používatelia proxy serveru lepšie identifikovať obsah stránky a tak aj zjednodušiť následné vyhľadávanie.



Obr. 3: Okno pre zobrazenie/úpravu/mazanie/pridávanie kľúčových slov

Technológie

Základom proxy servera je Rabbit (http://www.khelekore.org/rabbit/) postavený na technológii JAVA, nad ktorým je zrealizovaná vrstva služieb pomocou ktorej je možné proxy jednoducho rozširovať o zásuvné moduly s pokročilými možnosťami práce s HTTP požiadavkami a odpoveďami. Niektoré časti funkcionality (inicializácia cookie u používateľa, prezeranie logov, informácie) sú zabezpečené webovou aplikáciou v Ruby on Rails.

Prototyp

Funkčná verzia proxy servera je k dispozícii na http://peweproxy-staging.fiit.stuba.sk. Obsahuje návod na nastavenie, rozhranie pre prezeranie a mazanie údajov, ktoré server zaznamenal o používateľovi.

Šprint 6 – PMD85

Šprint PMD85 so sebou priniesol tieto požiadavky:

- Paralelné logovanie prístupov do CouchDB
- Presýpač logov z MySQL do CouchDB
- Možnosť nahlásiť nefunkčnú stránku
- Mergovanie variables.xml

Jednotlivé požiadavky sú opísané v nasledujúcich podkapitolách.

Paralelne logovanie prístupov do couchDB

Šprint: PMD85; *zodpovedná osoba:* Pavol Sokol

Analýza

Našim cieľom je zabezpečiť, aby dáta užívateľov boli dostupné medzi jednotlivými inštanciami adaptívnych proxy serverov, čím sa dosiahne hlavne:

- kvalitnejšie výsledky adaptívnej funkcionality, vďaka väčšej množine dát,
- migrácia používateľa na inú lokalitu bez ďalšieho zásahu,
- rozšírenie adaptívneho proxy na ďalšie univerzity,
- globálne rozšírenie adaptívneho proxy.

Pre zabezpečenie týchto potrieb bol zvolený presun na nový databázový server CouchDB.

CouchDB je dokumentová databáza, ktorej obsah je vytvorený z dokumentov a pohľadov. Dokument je sprístupnený vo formáte JSON, teda môže obsahovať ďalšie dokumenty, avšak tie sú v rámci databázy identifikované jedine ako súčasti koreňového dokumentu. Uložené dokumenty sú sprístupnené klientom prostredníctvom REST požiadaviek a navrátené vo formáte JSON. Pohľady slúžia na sprístupnenie dokumentov s kľúčom iným ako ID. Vyhľadávanie je realizované modelom MapReduce.

Medzi hlavné prednosti couchDB patrí možnosť nasadenia do distribuovaného prostredia, robustnosť a replikácia s obojsmernou detekciou konfliktov.

Návrh

Úlohou je zabezpečiť logovanie dát do CouchDB, pritom však ponechať súčasné logovanie do MySQL. Samotné logovanie sa vykonáva v troch triedach:

- CachingPageInformationProviderServiceModule.java – logovanie navštívených stránok,
- UserAccessLoggingProcessingPlugin.java – logovanie prístupov používateľov,
- ActivityLoggingProcessingPlugin.java – logovanie aktívít používateľov.

Štruktúra dát sa oproti MySQL mení, a to tak, že tabuľky PagesTerms a Terms sa vnoria priamo do dokumentu Pages. Súčasťou implementácie je vytvorenie pohľadu pre vyhľadávanie stránok podľa URL.

Implementácia a testovanie

Počas implementácie došlo v súbore CachingPageInformationProviderServiceModule.java k vytvoreniu príliš veľkej triedy, avšak tento stav je len dočasný, pretože funkcionalita spojená MySQL bude odstránená. Testovanie funkčnosti prebehlo manuálne.

Možnosť nahlásenia nefunkčnej stránky

Šprint:PMD85; *zodpovedná osoba: Ján Hudek*

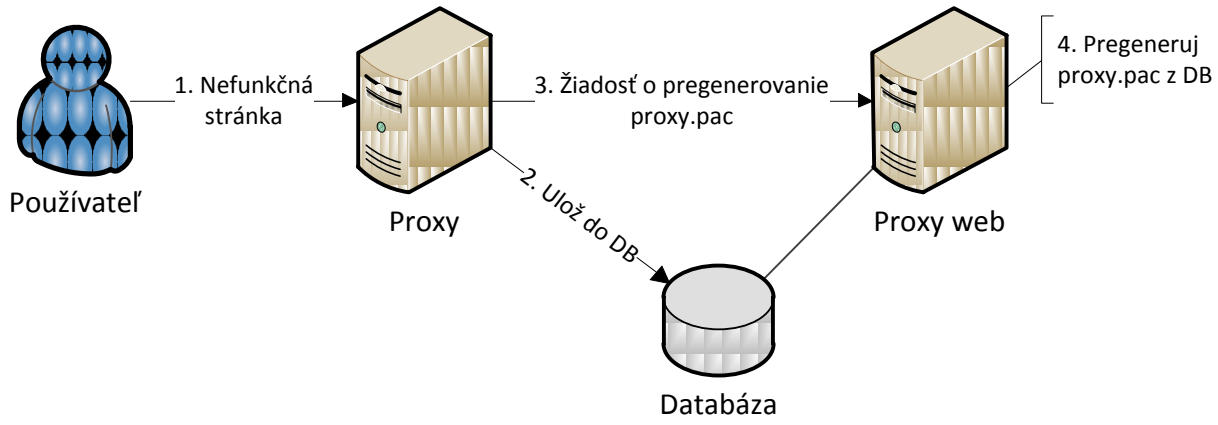
Analýza

Proxy server môže svojimi úpravami narušiť vzhľad alebo funkcionalitu stránky, do ktorej zasahuje. Pravdepodobnosť tohto nie je vysoká, no ak sa to stane, je nutné ponúknuť používateľovi možnosť nahlásiť nefunkčnú stránku.

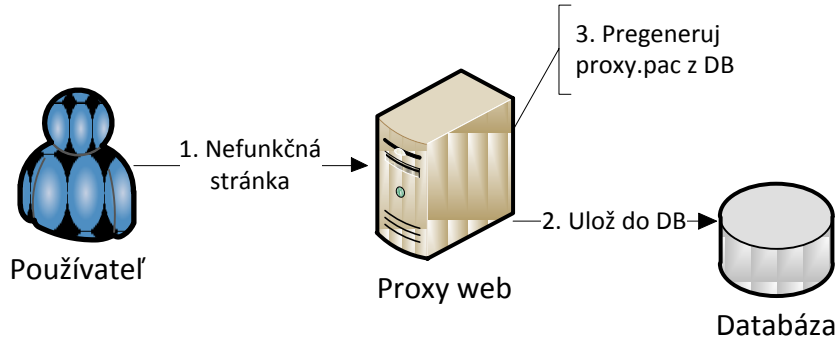
Návrh

Nefunkčnú stránku možno nahlásiť dvomi spôsobmi.

1. Pomocou „bublinky“ vkladanej do každej stránky:



2. Cez proxy web:



Implementácia

Nahlásené stránky sa ukladajú do tabuľky broken_pages. O vkladanie „bublinky“ do stránok a o spracovávanie requestov sa stará BrokenPageReporterProcessingPlugin v bundli adaptive-proxy-bundle-broken-pages-reporter. V príslušnom XML súbore k spomenutému pluginu je dôležitý parameter „pacGeneratorUrl“, ktorý obsahuje url s linkou, na ktorú treba odoslať žiadosť o pregenerovanie proxy.pac.

Na strane proxy-webu je v triede BrokenPageController implementované generovanie súboru proxy.pac na základe šablóny (proxy.pac.template) a záznamov v DB.

Mergovanie variables.xml z viacerých bundlov

Šprint: PMD85; *zodpovedná osoba:* Michal Valluš

Analýza

Každý bundel pluginov môže obsahovať súbor *variables.xml* v ktorom sú definované premenné konfigurácie. Pri nasadení je potrebné zabezpečiť, aby sa tieto premenné dostali do jedného súboru. To isté je potrebné zabezpečiť aj pri spúšťaní pomocou služby *developement-support.*

Aby sme sa vyhli riziku, že premenné z rôznych bundlov budú mať rovnaký názov, zavedieme konvenciu nazývania premenných:

`<variable name="bundle_name:variable_name">…</variable>`

Návrh

Do *developement-supportu* doplníme funkcionalitu spájania súborov *variables.xml.* Do deploy skriptu doplníme nový task *release:variables*. Funkconalitu je potrebné imlementovať tak, aby sa správala v obidvoch prípadoch identicky, aby tak nevznikali zbytočné chyby.

Implementácia a testovanie

V developement-support bola implementovaná nová trieda XMLUtils, ktorá obsahuje základnú funkcionalitu na prácu so súbormi *variables.xml.* Spracovanie v triede *Bootstrap* bolo doplnené o funkcionalitu spájania daných súborov.

Do hlavného *Rakefile* skriptu bol doplnený nový task *release:variables*, ktorý zabezpečuje funkcionalitu spájania súborov pomocou procesora *REXML*.

V obidve implementácie sa správajú tak, že nutne nevyžadujú súbor *variables.xml* v priečinku bundla. Spájajú teda iba nájdené súbory.

Šprint 7, 8 – Didaktik 1, 2

V siedmom šprinte boli identifikované nasledovné požiadavky:

- Keywords bublinka zobrazuje z couchDB
- RSS feed pre releasy proxy
- Úprava proxy-webu na CouchDB
- Cachovanie skriptov v JS Injektore
- Hromadnejšie mazanie z logov cez označenie checkboxami

Tieto požiadavky sa však v siedmom šprinte splniť nepodarilo, preto boli presunuté aj do ôsmeho šprintu. Jednotlivé požiadavky sú opísané v nasledovných podkapitolách.

Asynchrónne načítavanie JavaScript súborov

Šprint:Didaktik 2; *zodpovedná osoba: Bc. Ivan Pleško*

Analýza

Proxy server vkladá do stránok rôzne JavaScript súbory. V prípadoch, keď je proxy server vyťažený, môže poskytnutie tohto súboru trvať dlhšiu dobu, čo v konečnom dôsledku spomalí načítavanie stránky. To je samozrejme nežiaduce, nakoľko sa tým z pohľadu používateľa znižuje odozva webu. Preto je nutné JavaScript súbory načítavať asynchrónne.

Tu však vznikne problém s premennou __peweproxy_uid. Keďže aj súbor nastavujúci túto premennú sa načítava asynchrónne, nevieme zaručiť, že pre všetky skripty bude k dispozícii. Preto implementujeme callback funkciu, cez ktorú budú ostatné funkcie k tejto premennej pristupovať.

Návrh

Asynchrónne načítanie JavaScript súboru sa dá realizovať pomocou DOM funkcií samotného JavaScriptu a to tak, že pomocou týchto funkcií dynamicky vytvoríme <script src="script.js" type=text/javascript"></script> tag.

Callback funkcia bude fungovať tak, že jej parametrom bude funkcia, ktorú chceme zaregistrovať. Ak je k dispozícii premenná __peweproxy_uid, rovno sa funkcia v parametri spustí. Ak nie, pridá sa do poľa. Súbor, v ktorom sa nastavuje premenná __peweproxy_uid potom zavolá ďalšiu funkciu, ktorá prejde spomínané pole a všetky funkcie z neho zavolá.

Implementácia

Plugin JavaScriptInjectingProcessingPlugin obsahuje v konfiguračnom súbore parameter „loadAsynchronously" (implicitne true), ktorý rozhodne o tom, či sa JavaScript načíta synchrónne (vygenerovaním <script scr=...> tagu staticky) alebo asynchrónne (vygenerovanie skriptu, ktorý vygeneruje <script src=...> dynamicky).

Skript, ktorý generuje <script scr=...> dynamicky vyzerá nasledovne:

```
(function() {
      var s = document.createElement("script");
      s.type = "text/javascript";
      s.async = true;
      s.src = "cesta/ku/javascript/suboru.js";
      var x = document.getElementsByTagName("script")[0];
      x.parentNode.insertBefore(s, x);
})();
```

Proxy v plugine SetupJavaScriptEnvironmentProcessingPlugin vytvorí 2 JavaScript funkcie:

1. __ap_register_callback(function)
2. __ap_fire_callback()

Ak teda programátor potrebuje vo svojom JavaScripte, ktorý sa načítava asynchrónne, pristupovať k premennej __peweproxy_uid, urobí to nasledovne:

```
__ap_register_callback(function() {
      //telo metódy
});
```

Úprava proxy webu na CouchDB

Šprint: Didaktik; *zodpovedná osoba:* Michal Valluš

Analýza

Bolo potrebné vybrať API, pomocou ktorého budeme pristupovať ku CouchDB. Na výber boli CouchRest, Couch Potato, RelaxDB, ActiveCouch a dalšie. S pomedzi všetkých so nakoniec vybral Couch Potato, pretože ako jediná knižnica pristupovala k databáze ako ku dokumentovej databáze. Ostatné knižnice sa snažili napodobniť ActiveRecord API, ktoré bolo navrhnuté pre relačné databázy.

Prvé pokusy o implementáciu tiež odhalili, že pravdepodobne bude treba prerobiť stránkovanie, ktoré teraz funguje nad ActiveRecord API.

Návrh

V dátovom modeli aplikácie boli vytvorené nové modely, ktoré zodpovedali dokumentom v CouchDB. Boli navrhnuté pohľady (views), cez ktoré budeme dopytovať dokumenty.

```
_design/access_log:
"all_by_user": {
   "map":
      "function(doc) {
         if(doc.type == 'ACCESS_LOG') {
            emit([doc.userid,    doc.timestamp],    {page_id:    doc.page_id,
                  referer: doc.referer, timestamp: doc.timestamp});
         }
      },
   "reduce": null
}
"by_id": {
   "map":
      "function(doc) {
         if(doc.type == 'ACCESS_LOG' && doc.referer) {
            emit(doc._id, null);
         }
      }",
   "reduce": null
}
```

```
_design/page:
"by_id": {
   "map":
      "function(doc) {
         if(doc.type == 'PAGE') {
            emit(doc._id,       {pages_terms:       doc.pages_terms,       url:
                           doc.url});
         }
      },
   "reduce": null
}
```

Podľa dokumentácie ku CouchDB je vhodné robiť stránkovanie takým spôsobom, že sa nezískajú na začiatku všetky dokumenty, ale postupne sa získavajú zobrazované dokumentu pomocou parametru *startkey.* Toto je takzvané rýchle stránkovanie.

Druhý prístup je tzv. pomalé stránkovanie, kedy sa na začiatku získajú všetku dokumenty, uložia sa do poľa a stránkovanie pracuje iba z týmto poľom. Nevýhodou tohto prístupu je, že pri veľkých počtoch dokumentov môže byť dosť pomalý.

Implementácia a testovanie

Získavanie dokumentov z CouchDB fungovalo pomerne skoro a pomerne dobre. Problémy začali, keď bolo treba implementovať stránkovanie.

Ukázalo sa, že rýchle stránkovanie, hoci bolo odporúčané v dokumentácii ku CouchDB má niektoré špecifiká, ktoré sa implementačne len veľmi ťažko prekonávajú. Nakoniec bolo implementované aj toto rýchle stránkovanie s tým, že logy nie sú zoradené podľa dátumu. Tomuto by sa dalo vyhnúť tým spôsobom, že by sa nastavilo prideľovanie ID, ktoré stúpajú s časom.

Bolo implementované aj pomalé stránkovanie, a to veľmi elegantne pomocou knižnice *will_paginate.* Toto stránkovanie by bolo pravdepodobne pomalšie pri veľkých počtoch dokumentov, ale prináša niekoľko výhod: jednoduché použitie knižnice, jednoduché zoradenie logov podľa času, dopredu známy počet stránok a posielanie menšieho počtu parametrov.

Hromadnejšie mazanie z logov cez označenie checkboxami

Šprint: Didaktik; *zodpovedná osoba:* Michal Valluš

Analýza

Je potrebné vymeniť linky na zmazanie za checkboxy a pridať na stránku hlavný checkbox, ktorý bude označovať ostatné checkboxy.

Takisto bude potrebné upraviť funkciu na vymazávanie tak, aby akceptovala aj pole dokumentov.

Návrh

Hlavný checkbox na označovanie ostatných checkboxov bude implementovaný pomocou javascriptu:

```
function check_all(sender, checkboxes){
   for (i = 0; i < checkboxes.length; i++){
      checkboxes[i].checked=sender.checked
   }
}
```

Implementácia a testovanie

Funkcionalita bola implementovaná a otestovaná.

Šprint 9 – Didaktik 3

V 9. šprinte boli definované nasledovné požiadavky:

- Identifikácia cieľa vyhľadávania na webe
- Realtime Social Navigation

Jednotlivé požiadavky sú opísané v nasledovných podkapitolách.

Identifikácia cieľa vyhľadávania na webe

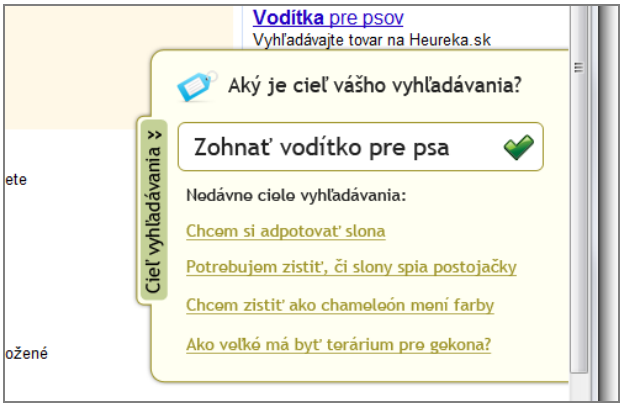
Šprint:Didaktik 3; zodpovedná osoba: Bc. Ivan Pleško

Analýza

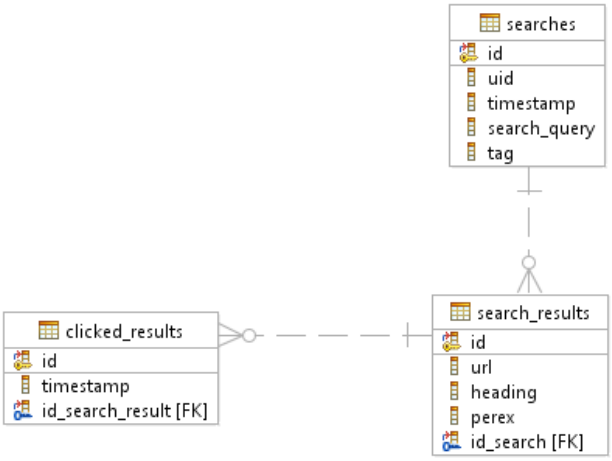
Proxy server ponúkne používateľovi možnosť identifikovať svoj cieľ vyhľadávania na webe. Ak napríklad používateľ na google zadá „vodítka“, môže proxy serveru oznámiť, že hľadal príslušenstvo pre svojho psa. Zároveň existuje požiadavka sledovať používateľovu aktivitu – teda ktorý odkaz zvolil (na ktorý klikol) za najvyhovujúcejší jeho vyhľadávaniu.

Návrh

Graficky táto funkcionalita vyzerá nasledovne:



Databáza obsahuje 3 tabuľky. Znázornené sú obrázkom nižšie.



Implementácia

Funkcionalita je implementovaná ako samostatný bundle pluginov s názvom adaptive-proxy-bundle-search-goals. Celý bundle je závislý na adaptive-proxy-bundle-search, ktorý poskytuje výsledky vyhľadávania rôznych webových vyhľadávačov.

Bundle obsahuje 3 pluginy:

- ClickTrackingInjectorProcessingPlugin – tento plugin sa stará o úpravu odkazov výsledku vyhľadávania tak, aby proxy vedel zaznamenať používateľovu aktivitu. Zároveň sa stará o vloženie JavaScriptu na odoslanie používateľovho uid späť na proxy server.
- SearchGoalsClientInjectorPlugin – v prípade, že je k dispozácii ModifiableSearchResultService (v dobe písania tohto dokumentu je to pre google.com, yahoo.com a bing.com), vkladá potrebný JavaScript súbor.
- SearchGoalsCommunicationProcessingPlugin – stará sa o spracovávanie requestov ako uloženie cieľa vyhľadávania, zaznamenanie kliknutého výsledku vyhľadávania a podobne.

Realtime social navigation

Šprint: Didaktik 3; *zodpovedná osoba:* Ján Hudek

Analýza

Celý internet je poprepájaný hypertextovými odkazmi, spájajúcimi stránky a portály. Pre používateľa by mala veľký prínos možnosť sledovať koľko ďalších aktívnych používateľov sa nachádza na adresách kam smerujú odkazy zo stránky na ktorej sa práve nachádza.

Do používateľského menu je potrebné implementovať funkcionalitu ktorá by umožňovala takéto sledovanie. Každý užívateľ by zároveň mal možnosť túto funkcionalitu vypnúť alebo zapnúť.

Návrh a implementácia

Po prechode stránky proxy serverom sa do stránky vloží JavaScript vykonávajúci funkcionalitu *social navigation*. O vloženie tejto funkcionalitu sa stará služba *clientBubbleMenuService*. Po načítaní stránky u používateľa sa zo stránky získajú všetky odkazy. Následne sa na server odošle požiadavka na získanie aktuálnych údajov. Tieto dáta sa aktualizujú každých 8 sekúnd. Na serveri je uložený zoznam odkazov s počtom klientov na každej adrese. Tento zoznam sa taktiež aktualizuje v pravidelných intervaloch. V nasledujúcej časti je opísaná komunikácia medzi klientom a serverom.

Návrh komunikácie

Zistenie či je aktivované sledovanie pohybu užívateľov

Call: adaptive-proxy/userCountCall.php?action=getPeoplemeterActivity&uid=*userUuid*
Parameter: none
Return: TRUE/FALSE

Aktivovanie/Deaktivovanie sledovania

Call: adaptive-proxy/userCountCall.php?action=setPeoplemeterActivity&activity=false&uid=*userUuid*
Parameter: none
Return: OK/FAIL

Call: adaptive-proxy/userCountCall.php?action=setPeoplemeterActivity&activity=true&uid=*userUuid*
Parameter: none
Return: OK/FAIL

Získanie počtu užívateľov k odkazom

Call: adaptive-proxy/userCountCall.php?action=updateCounts
Parameter (format JSON):
pageUrlList =

```
{ "pageUrlList" : [
        { "id":"id", "url":"url" },
        { "id":"id", "url":"url" },
        { "id":"id", "url":"url" }
]}
```

- Podčiarknuté sa nahradí reálnou hodnotou. Názov POST premennej je *pageUrlList*.

Return (format JSON):

```
{ "peopleCount" : [
        { "id": 0, "count": 2 },
        { "id": 1, "count": 12 },
        { "id": 2, "count": 9 },
        { "id": 3, "count": 0 },
        { "id": 4, "count": 10 }
]}
```

- Naspäť sa posielajú dvojice id (získané z požiadavky) a počet užívateľov.