**Postprocessing dát extrahovaných z webu**

**Autor:** Samuel Koprda

**Vedúci práce:** RNDr. Peter Gurský, PhD.

**Projekt Kapsa - kedysi**

Projekt, ktorého pôvodným cieľom bolo vytvoriť katalóg produktov ponúkaných internetovými obchodmi, ktorý mal umožniť porovnávanie produktov podľa vlastností produktov, skúseností používateľov a ponuky predajcov. Pomocou nastroja Exago, ktorý je súčasťou projektu bolo možné anotovať ľubovoľný produkt v ľubovoľnom internetovom obchode na základe čoho bolo možné spustiť extrakciu produktov, ktoré spĺňali definované požiadavky.

**Motivácia**

Nakoľko sa rozhodlo, že Kapsa chce rozšíriť funkcionalitu na spracovávanie nielen produktov z internetových obchodov, ale aj ľubovoľných štruktúrovaných dát, bolo potrebne urobiť zmeny. Prvou zásadnou zmenou, ktorú bolo treba vykonať bolo, aby Exago bolo schopne extrahovať ľubovoľné štruktúrované dáta. Aby sa takéto dáta mohli uložiť, bolo nutne pozmeniť aj tú časť projektu, ktorá bola zodpovedná za spracovanie a uchovávanie dát, teda server a databázu. Avšak extrahované dáta treba niekedy modifikovať aby spĺňali potreby používateľa. Ak je tento úkon pravidelný používateľ nebude chcieť pri každej extrakcii dáta modifikovať ručne a preto sme sa rozhodli že tento proces zautomatizujeme a to tak, že na dáta aplikujeme pravidlá, ktoré sa následne použijú pri každom ďalšom extrahovaní. Tieto pravidlá si používateľ nastaví na vzorke extrahovaných dát, aby vyhovovali jeho potrebám a aplikuje ich na zvyšné dáta, pričom pri každej ďalšej extrakcii sa dane pravidlá použijú na všetky dáta, aby sa zabránilo opakovanému vytváraniu pravidiel. Mojou úlohou bolo navrhnúť a implementovať server zvoliť vhodnú databázu, navrhnúť dátovú štruktúru a navrhnúť a implementovať modulárny systém na konverziu dát za pomoci používateľom zvolených konverzných pravidiel, ku ktorému bolo treba vytvoriť aj vhodne používateľské rozhranie.

**Databáza**

Pôvodná databáza, v ktorej boli uložené všetky dáta a metadáta potrebne na chod projektu Kapsa, bola MySQL, ktorá v čase keď Kapsa pracovala výhradne s produktmi perfektne vyhovovala. Avšak zmeny v spracovaných dátach si vyžiadali zamyslieť sa nad tým či bude MySQL vhodná aj pre všeobecné dáta. Ukázalo sa, že takéto dáta by MySQL bola schopná uchovávať, ale práca s nimi by bola veľmi neefektívna, preto sa rozhodlo že pre naše účely bude vyhovovať NoSQL databáza MongoDB, ktorá poskytuje nasledujúce výhody:

1. **Škálovateľnosť** – jedna z hlavných výhod MongoDB je možnosť rozdistribuovať záťaž spracovania požiadaviek, čím klesne čas potrebný napríklad na vkladanie údajov, čo je vidieť hlavne pri používaní databázy viacerými používateľmi súčasne
2. **Kolekcie** – nevyhnutná výhoda je radenie dát do tzv. kolekcií, ktoré v sebe obsahujú tzv. dokumenty umožňujúce uložiť objekty typu JSON a teda ľubovoľne vnorené objekty obsahujúce primitívne typy, polia a ďalšie objekty, potrebné na dosiahnutie našich cieľov
3. **Big data** – databáza je navrhnutá tak, aby dokázala zvládnuť tento fenomén, čo je pre nás veľmi dôležité nakoľko extrahovanie veľkých webových portálov prináša veľké množstvo dát

S týmito výhodami sme schopní dosiahnuť efektívnejšie spracovanie dát a tým aj zvýšiť zážitok z používania a kvalitu projektu.

**Návrh databázy**

Databázu sme rozdelili do nasledovných kolekcii:

1. **Country** a **Language** – číselníky obsahujúce potrebne informácie pre fungovanie nastroja Exago
2. **Source** – číselník ktorý slúži na uchovávanie adries portálov (zdrojov) z ktorých sa extrahovali dáta
3. **Wrapper** – kolekcia, ktorá obsahuje tzv. wrapper-y; wrapper predstavuje výsledok po anotácii zapísaný do štruktúrovanej podoby
4. **Download**– kolekcia obsahujúca informácie o jednotlivých procesoch crawlingu a extrakcie (začiatok, koniec, stav, atď.)
5. **IDAssigner** – slúži na inkrementálne prideľovanie jedinečných číselných identifikátorov pre niektoré kolekcie, nakoľko MongoDB štandardne prideľuje hexadecimálne identifikátory a pre nás bolo potrebné aby dokumenty v určitých kolekciách mali 64 bitový číselný identifikátor
6. **Dátové kolekcie** – ak sa pre nejaký wrapper spusti download tak sa vytvorí kolekcia s jedinečným názvom, do ktorej budú následne uložené všetky extrahované dáta
7. **Kolekcie s pravidlami**  - takisto ako pre dátové kolekcie aj pre kolekcie s pravidlami platí že sa vytvárajú dynamicky; kolekcia s pravidlami vznikne práve vtedy keď používateľ vytvorí sadu pravidiel pre svoje dáta. Tato kolekcia bude následne použitá na konverziu prichádzajúcich dát z nasledujúcich downloadov pre ten istý wrapper

Pomocou takéhoto návrhu databázy sme schopní úplne nahradiť MySQL návrh a potrebu naďalej používať MySQL databázu.

**Server**

Komunikáciu medzi Exago-m a databázou a databázou a používateľským rozhraním sprostredkuje server. Zmenené požiadavky od projektu Kapsa si vyžiadali aj zmenu servera, ktorá spočívala vo vytvorení novej perzistentnej vrstvy nakoľko sa menil aj typ databázy z MySQL na MongoDB. Server je implementovaný v programovacom jazyku Java pomocou Spring frameworku ktorý poskytuje širokú skalu nástrojov na prácu z dátami medzi ktoré patri vytváranie spojení s rôznymi typmi databáz a tým poskytuje výhody pre vývojárov ktorým umožňuje dosiahnuť požadované akcie s minimálnym úsilím. Pomocou Spring framework-u sme taktiež vytvorili REST server na poskytnutie API pre Exago a Webové rozhranie na zobrazovanie a modifikáciu dát. Spring umožňuje rozdelenie serveru na samostatné časti z ktorých každá je nezávislá a tým aj ľahko manažovateľná.

**Webové používateľské rozhranie**

Dáta, ktoré používateľ extrahuje bude chcieť aj nejakým spôsobom vidieť pripadne modifikovať a práve pre tieto účely je súčasťou mojej prace aj webové rozhranie v ktorom bude môcť používateľ robiť potrebné akcie. Používateľské rozhranie je rozdelené na niekoľko samostatných modulov:

1. **Download modul**

* modul, ktorý slúži na výber extrahovaných dát pričom je potrebné aby používateľ zvolil tzv. Source, Wrapper a Download, z ktorých povinný údaj je len Download na základe ktorého systém dokáže určiť ktoré dáta používateľ požaduje
* Source a Wrapper slúžia na zúženie výberu Downloadu pričom platí, že pre každý Source existuje niekoľko Wrapper-ov a pre každý Wrapper existuje práve jeden Source a analogicky pre Wrapper a Download
* po výbere Download-u je používateľovi ponúknutá možnosť kliknúť na tlačidlo ktoré spustí doručovanie požadovaných dát a presmerovanie na obrazovku zobrazovania a úpravy dát

A screenshot of a cell phone

Description automatically generated

1. **DataSelect modul**

* modul, slúžiaci na zobrazenie dát v podobe tabuľky a stromovej štruktúry modelu (pre rýchlejšiu navigáciu)
* z tabuľky si potom používateľ dokáže vybrať ľubovoľný atribút ktorý chce modifikovať z rôznych dôvodov (napr. „<h1>Herci</h1>“ namiesto „Herci“)

1. **Conversion modul**

* posledný modul slúži na zobrazovanie a riadenie úpravy dát (konverzie)
* používateľ si z vybraných atribútov vyrobí nové pravidlo ktoré sa aplikuje na vzorku dát a ak s nim bude používateľ spokojný jednoducho ho uloží kedy bude vyzvaný či si chce vybrať nové atribúty na úpravu alebo ukončiť modifikáciu a aplikovať pravidla na všetky dáta pre daný wrapper

**Systém pravidiel**

V systéme rozlišujeme dva pojmy:

1. **Typ pravidla**

* hovorí o tom na aké typy atribútov sa da použiť
* šablóna na základe ktorej sa budú vytvárať konkrétne **pravidlá**

1. **Pravidlo**

* konkrétne použitie **typu pravidla** na konkrétne dáta
* na základe neho sa budú upravovať dáta

Ako už bolo povedané, jednotlivé pravidlá sa budú písať pre dáta ku konkrétnemu wrapper-u a budú uložené v samostatnej kolekcii určenej práve na tento účel. Tento systém ma nasledovne vlastnosti:

1. Pravidlá na vstupe dostanú **existujúci atribút** a na výstupe vznikne **novy atribút** ktorému používateľ zvoli cieľovú destináciu v modeli aktuálnych dát, kde bude priradený.
2. Pravidlá budú na dáta aplikovane presne **v poradí** v akom boli vytvárané čim sa umožní pri vytváraní pravidiel použiť atribút, ktorý vznikol práve na základe nejakého pravidla.
3. V prípade že si používateľ bude chcieť zobraziť dáta pre ktoré už existujú pravidlá budú mu zobrazené dáta po aplikovaní týchto pravidiel a jednoducho bude môcť dane pravidlá rozšíriť o nove pravidlá.
4. Pravidlá budú mat pridelený **identifikátor wrapper-u** a teda bude jednoduché spárovať ich s dátami pre ktoré boli navrhnuté.

**Modularita systému**

Jednou z hlavných požiadaviek na náš systém je modularita. Modularita systému znamená že pridanie nového typu pravidla bude jednoduché a nebude vyžadovať zmenu návrhu projektu. Pridanie nového typu pravidla bude vyžadovať:

1. **Zmena na server-i**
   * na serveri sa bude diať proces konverzie dát teda ku každému typu pravidla bude potrebne napísať čo sa ma udiať keď prídu dáta a k nim konkrétny popis z typu pravidla
   * každý typ pravidla bude reprezentovaný samostatnou triedou implementujúcou rozhranie „RuleType“, obsahujúce metódu „convert“, ktorá bude na vstupe brat zoznam obsahujúci dáta
   * tuto triedu potom bude potrebne pridať do zoznamu typov pravidiel
2. **Zmena vo webovom rozhraní**
   * na pridanie nového typu pravidla bude vo webovom používateľskom rozhraní potrebne vytvoriť **nový komponent** na určené miesto
   * komponent v sebe môže obsahovať iné komponenty ktoré budú taktiež uložené na dohodnutom mieste