Dokumentacja końcowa

Temat: Ekstraktor treści HTML

Przedmiot: TKOM

Autor: Patryk Karbownik (nr indeksu: 293113)

1. Cel projektu

W ramach projektu stworzono bibliotekę pozwalającą na ekstrakcję treści z dokumentu HTML do instancji klasy zdefiniowanej przez programistę przy pomocy opracowanego języka pozwalającego na definiowanie znaczników do wydobycia.

2. Wymagania funkcjonalne i niefunkcjonalne

Wymagania funkcjonalne:

- parsowanie języka konfiguracji
- parsowanie dokumentów HTML
- możliwość definiowania elementów do wydobycia
- wydobywanie zadanych elementów z dokumentu
-tworzenie obiektów klasy zdefiniowanej przez użytkownika zawierającej wydobyte elementy typu *String* lub *java.awt.image.BufferedImage* (tylko w przypadku, gdy podana jest pełna ścieżka do zasobu)
- zwracanie zbioru stworzonych obiektów
- pobieranie danych wejściowych z abstrakcyjnych źródeł

Wymagania niefunkcjonalne:

- zapewnianie dokładnych komunikatów błędów
- wykonanie przy użyciu języka Java

3. Opis implementacji

Projekt został zrealizowany w języku Java z wykorzystaniem środowiska IntelliJ IDEA. Do stworzenia testów jednostkowych wykorzystano framework JUnit, z kolei biblioteka Apache Common Text znalazła zastosowanie w przetwarzaniu tekstu wydobytego z dokumentu HTML. Do zarządzania zależnościami użyto narzędzia Maven.

Projekt dzieli się na następujące moduły:

a) Moduł obsługi źródła

Odpowiada za wczytywanie pojedynczych znaków ze źródła oraz obsługę aktualnej pozycji. Jego funkcjonalność została zaimplementowana w klasach: *Position* oraz *Reader* z pakietu *reader*.

b) Analizatory leksykalne

W ramach projektu zaimplementowano dwa analizatory leksykalne. Jeden dla zdefiniowanego języka oraz drugi dla HTML. Odpowiadają za generację tokenów z ciągów znaków dostarczanych przez moduł obsługi źródła. Na wyjściu przekazują je do odpowiednich parserów. Klasy odpowiadające za tą funkcjonalność znajdują się w pakiecie *lexer*.

c) Analizatory składniowe

Na wejściu pobierają tokeny z odpowiadających im analizatorów leksykalnych. Następnie budują drzewa wyprowadzeń na podstawie gramatyki, które ostatecznie trafiają do modułu wydobywającego dane. Klasy odpowiadające za funkcjonalność parsowania dokumentów HTML znajdują się w pakiecie *parserhtml*, z kolei w pakiecie parser znajduje się implementacja analizatora dla zdefiniowanego w ramach projektu języka.

d) Moduł wydobywający dane

Odpowiada za wydobywanie zadanych zasobów dla zdefiniowanych klas na podstawie dostarczonych drzew składniowych. Podczas implementacji znalazły zastosowanie wzorce projektowe fabryka oraz wizytator. Pierwszy posłużył do tworzenia zadanych obiektów przy wykorzystaniu mechanizmu refleksji. Drugi wykorzystano do implementacji logiki związanej z poruszaniem się po drzewach składniowych. Klasy odpowiadające za tą funkcjonalność znajdują się w pakietach: *extractor*, *factory* oraz *visitor*.

e) Klasy wyjątków

Znajdują się w pakiecie *exceptions*. Zdefiniowano wyjątki dla analizatorów leksykalnych, składniowych oraz modułu wydobywającego dane.

4. Metody modułu wydobywającego dane udostępnione programiście

- **Extractor**(**HashMap<String, Resource>** resources**, Root** *htmlDocumentRoot*)

  Konstruktor klasy ekstraktora pozwalający na przekazanie zarówno zbioru z definicjami zasobów, jak i dokumentu HTML.

- **void** setHtmlDocumentRoot(**Root** *htmlDocumentRoot*)

  Metoda pozwalająca na przekazania dokumentu HTML.

- **void** setResources(**HashMap<String, Resource>** *resources*)

  Metoda pozwalająca na przekazanie zbioru z definicjami zasobów.

- **void** extract(**String** *resourceName*)

  Metoda pozwalająca na wydobycie zasobu o zadanej nazwie.

- **ArrayList<Object>** getExtractedObjects()

  Motoda dokonuje ekstracji zasobów

5. Testowanie

Zostały napisanie testy jednostkowe dla klasy obsługującej źródło, analizatorów leksykalnych oraz składniowych. Na potrzeby testów całej biblioteki utworzono zbiór definicji zasobów wraz z odpowiadającymi im klasami oraz zaimplementowano dwa proste programy prezentujące uzyskany rezultat.

6. Gramatyka

resource = "resource" identifier definition_block { "resource" identifier definition_block } ;

definition_block = "{" tag_sentence ["conditions" conditions_block] class_line
set_fields field_definition_block
(amount_sentence | range_sentence) "}";

tag_sentence = "tag" "=" identifier ";" ;

conditions_block = "{" condition_sentence {condition_sentece} "} ";

condition_sentence = "if" condition ";";

condition = term { “or” term };

term = factor { "and" factor };

factor = path condition_operator factor_object | “(“ condition “)”;

condition_operator = “has”;

factor_object = ["not"] ("tag" identifier | the_subject (identifier | “class”) [comparison_object];

the_subject = "attribute";

comparison_object = comparision_operator string | “in” value_set;

values_set = "{" string { "," string } "}";

comparison_operator = "==" | "!=";

path = "self" | "ancestor" | "descendant" | "parent" {"." , "parent"}

|"child" [, relative_condition] [{, "." , "child" [, relative_condition] };

relative_condtion = “(“ condition “)”;

class_line = "export" "to" "class" "=" string ";" ;

set_fields = "set" "fields";

field_definition_block = "{" field_definition { field_definition } "}";

field_definition = "field" identifier "=" path_to_resource ([, "." , "attribute" , "[",
(identifier | “class”), "]" [, "." , ”img"]] | “.”, “resource”, “[“ identifier “]”
)";" ;

path_to_resoruce = "from", "(", ("self" | tag_path_element {, "." , tag_path_element }) ")”;

tag_path_element = "tag", "[" identifier "]" [, "[" number "]"];

amount_sentence = "amount" "=" (number | "every") ";" ;

range_sentence = “range” “=” “(“ number “,” number “)” “;”;

number = zero | digit , {all_digits};

identifier = (letter | basic_special_character) {, letter | basic_special_character | all_digits

};

string = ‘"’ , string_content , ‘"’ ;

string_content = (letter | all_digits | special_character) { , (letter | all_digits | special_character)};

all_digits = zero | digit;

digit = "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" ;

zero = "0";

special_character = basic_special_character |"." | empty_character | "-";

basic_special_character = "_" |"$" | "!" | ;

empty_character = " ";

letter = big_letter | small_letter ;

big_letter = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" | "L" | "M" | "N" | "O" |"P" | "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z" ;

small_letter = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" | "r" | "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z";

7. Przykładowe skrypty

Przykład dla kodu:

```
▼<a class="sc-1nkodm-3 cJHTjK sc-1k2mbc5-0 fqAXAi" href="https://wiadomosci.wp.pl/aleksiej-nawalny-czlowiek-ktorego-putin-boi-sie-najbardziej-6630749852429088a" data-st-clk="{"teaserId":6630762962237569,"sgcat":201,"sgcatidx":4}" id="6630762962237569" data-testid="teaserStandard"> flex
  ▼<div class="sc-1c7s2z7-0 ldnOVe sc-1u95i6z-0 bDLqm"> flex
     <img src="https://v.wpimg.pl/NjRkMTljYQszCTtnQxFsHnBRbz0FSGJIJ0l3dkNZfloqWnVkQwwn…xx RTYCPiQACzdFNwYvMQhFLQtlXHhtDw5jU2QOfnlYWXldf1J7N1lHKFwwXXtgCFt3CTZZbyk" alt role="presentation" class="apty8v-0 ffvnAM">
    </div>
  ▼<div class="wnncvq-0 kXpmFO"> flex
    ▼<div class="sc-1faltli-0 cuTRJo"> flex
      ▼<div class="sc-1xz7k4j-0 Dsujm"> flex
          <span class="pfxg2j-0 hXdnAK sc-1xz7k4j-1 kUvdQj">JAN ROJEWSKI</span> flex
        </div>
      </div>
      <div class="sc-1qdlbrk-0 ftEaYY sc-1k2mbc5-1">Co dalej z Nawalnym? "Wszystko na jedną kartę"
      </div>
      ::after
    </div>
  </a>
```

Skrypt (względem tagu a):

```
Resource aResource
{
        tag = a;

        export to class = “Resource”;
        set fields
        {
                field img = from(tag[div].tag[img]).attribute[src].asImg;
                field author = from(tag[div][1].tag[div].tag[div].tag[span]);
                field description = from(tag[div][1].tag[div][1]);
        }
        amount = every;
}
```

Przykłady dla kodu:

Skrypt (względem tagu img):

```
Resource divResource
{
	tag = img;

	conditions
	{
		if parent.parent has attribute class == “media-content”;
		if parent has attribute title;
		if self has attribute alt;
	}

	export to class = “Link”;
	set fields
	{
		field link_to_image = from(self).attribute[src];
		field description = from(self).attribute[alt];
	}
	amount = every;
}
```