

POLITECHNIKA WROCŁAWSKA

PROJEKT

PROJEKTOWANIE SYSTEMÓW Z DOSTĘPEM W JĘZYKU NATURALNYM

**Opracowanie aplikacji wspomagającej
tworzenie korpusu z tekstami w języku
polskim.**

Authors:

Rafał PIENIAŻEK

Jakub POMYKAŁA

Supervisor:

Dr inż. Dariusz BANASIAK

17 grudnia 2017

Spis treści

1	Cel projektu	2
2	Opis aplikacji	2
3	Lista funkcjonalności	2
4	Instrukcja uruchomienia	2
4.1	Wymagania	2
4.2	Uruchomienie	2
4.3	Menu konsolowe i domyślne parametry	3
4.4	Logi programu	4
4.5	Struktura pobranych plików	5
5	Implementacja aplikacji	5
5.1	Struktura plików w projekcie	5
5.2	Encja artykułu	7
5.3	Wywołania parserów dla kolejnych witryn	7
6	Użyte technologie	8
7	Kod źródłowy oraz zebrane dane	8
8	Wnioski	9

1 Cel projektu

Celem projektu jest stworzenie systemu do pobierania danych z serwisów internetowych. Pobrane dane będą służyć w przyszłości do analizy stylometrycznej i przetwarzania języka naturalnego. Aby wyniki tych badań były miarodajne istotne jest usunięcie z tekstów zbędnych znaków świadczących o pochodzeniu tekstu. Nie powinno być tam daty stworzenia tekstu, tagów HTML, imienia i nazwiska autora. Wspomniane metadane powinny być oddzielone od rdzennej części tekstu i przechowywane niezależnie od niego

2 Opis aplikacji

Głównym zadaniem aplikacji będzie pobieranie tekstów ze stron WWW oraz zapisywanie ich na dysku w formacie *.txt lub JSON. Aplikacja została stworzona w języku Java, a dane pobierane były pomocy biblioteki parsującej - JSOUP. Proste konsolowe menu pozwoliło użytkownikowi na dostosowanie opcji pobierania i zapisu.

3 Lista funkcjonalności

- Pobieranie tekstów z wybranych stron WWW
- Usunięcie zbędnych fragmentów tekstu (adresy URL, cytaty, itp.)
- Zapis tekstów w osobnych plikach w formacie *.txt lub JSON
- Zapis wszystkich tekstów jednym plikiem w formacie *.txt lub JSON
- Zapis tekstów w katalogach według źródła w formacie *.txt lub JSON

4 Instrukcja uruchomienia

4.1 Wymagania

Aplikacja została napisana w języku Java, dlatego też wymagany jest sprzęt z zainstalowaną Javą w wersji 8 lub wyższej.

4.2 Uruchomienie

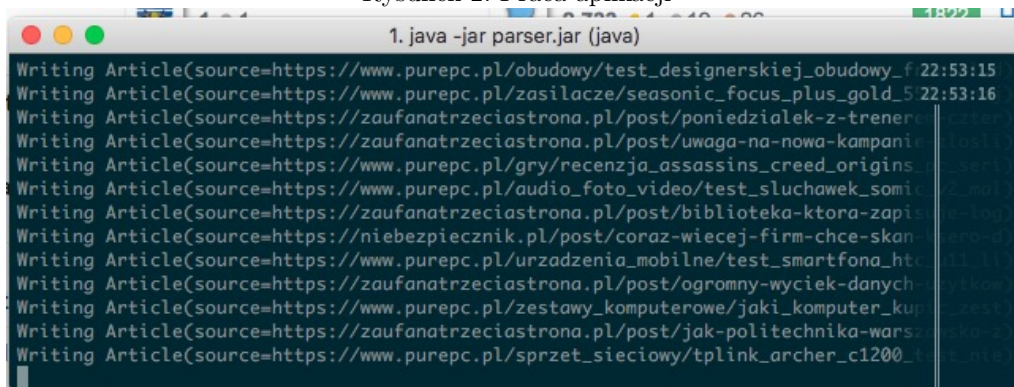
Aby uruchomić aplikację należy otworzyć emulator terminala, przejść do katalogu w którym znajduje się plik programu z rozszerzeniem *.jar. Następnie należy uruchomić aplikację poleceniem: `java -jar parser.jar` Aplikacja rozpocznie procedurę startu i automatycznie uruchomi się w trybie typu *shell*.

Rysunek 1: Tryb shell aplikacji



Aby uruchomić pobieranie artykułów z domyślnymi parametrami należy wydać polecenie *start*. Zostaną uruchomione 4 wątki pobierające i zapisujące artykuły. Wszystkie dane zostaną zapisane w katalogu o nazwie *output*, który zostanie stworzony automatycznie.

Rysunek 2: Praca aplikacji



4.3 Menu konsolowe i domyślne parametry

Aplikacja została wyposażona w możliwość wyboru parametrów przy starcie programu. Klient aplikacji może ustawić trzy parametry. Poniżej przedstawiono listę możliwych argumentów dla każdego z tych aplikacji.

- output
 - JSON
 - TXT
- resolve
 - ARTICLE - wszystkie artykuły w jednym katalogu

- SOURCE - artykuły według źródła strony
- ALL - wszystko w jednym pliku
- encoding
 - UTF-8
 - windows-1250
 - windows-1252
 - UTF-16
 - ISO-8859-x

Wszystkie wspierane kodowania znajdują się w dokumentacji Java, pod adresem: <https://docs.oracle.com/javase/8/docs/technotes/guides/intl/encoding.doc.html>

Rysunek 3: Argumenty startu

```

1. java -jar parser.jar (java)
AVAILABLE COMMANDS

Built-In Commands
  clear: Clear the shell screen.
  exit, quit: Exit the shell.
  help: Display help about available commands.
  script: Read and execute commands from a file.
  stacktrace: Display the full stacktrace of the last error.

Example Data Text
  mongotest: Test zapisywania

Writing Option Commands
  start: Uruchom parser z domyślne parametrami

shell:>start --encoding UTF-8 --output TXT --resolve ARTICLE
  
```

4.4 Logi programu

Poniżej przedstawiono logi aplikacji uruchomionej z domyślnymi parametrami. Zawarte są tam informacje jaki aktualnie artykuł został pobrany i przetworzony wraz z informacjami o URL i tytule.

Rysunek 4: Logi startu aplikacji

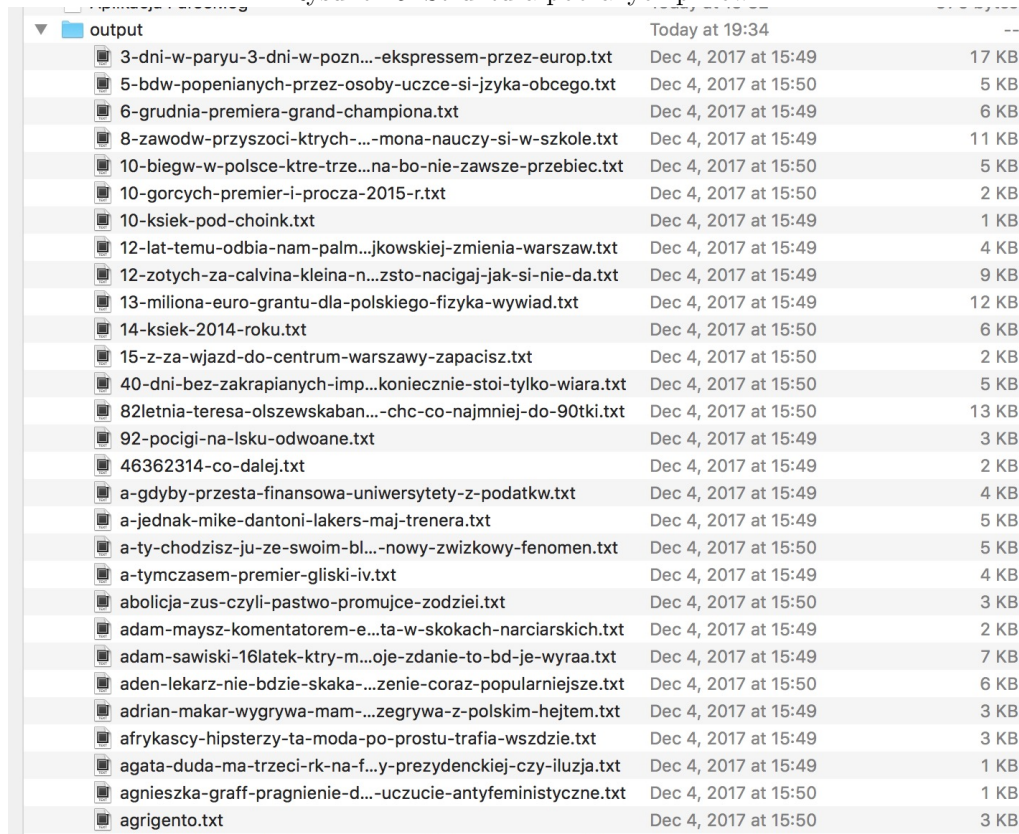
```

1. java -jar parser.jar (java)
Writing Article(source=https://www.purepc.pl/obudowy/test_designerskiej_obudowy_f 22:53:15
Writing Article(source=https://www.purepc.pl/zasilacze/seasonic_focus_plus_gold_5 22:53:16
Writing Article(source=https://zaufanatrzeciastrona.pl/post/poniedzialek-z-trenera 22:53:17
Writing Article(source=https://zaufanatrzeciastrona.pl/post/uwaga-na-nowa-kampanie 22:53:18
Writing Article(source=https://www.purepc.pl/gry/recenzja_assassins_creed_origins 22:53:19
Writing Article(source=https://www.purepc.pl/audio_foto_video/test_sluchawek_somic 22:53:20
Writing Article(source=https://zaufanatrzeciastrona.pl/post/biblioteka-ktora-zapis 22:53:21
Writing Article(source=https://niebezpiecznik.pl/post/coraz-wiecej-firm-chce-skan 22:53:22
Writing Article(source=https://www.purepc.pl/urzadzenia_mobilne/test_smartfona_htc 22:53:23
Writing Article(source=https://zaufanatrzeciastrona.pl/post/ogromny-wyciek-danych 22:53:24
Writing Article(source=https://www.purepc.pl/zestawy_komputerowe/jaki_komputer_kup 22:53:25
Writing Article(source=https://zaufanatrzeciastrona.pl/post/jak-politechnika-warsz 22:53:26
Writing Article(source=https://www.purepc.pl/sprzet_sieciowy/tplink_archer_c1200_b 22:53:27
  
```

4.5 Struktura pobranych plików

Niezależnie od struktury pobrane artykuły mają określoną strukturę. W każdym wydzielony jest tytuł oraz treść artykułu. W zależności od możliwości w pliku znajduje się również autor, data utworzenia artykułu, kategoria i lista tagów. Każdy plik z artykułem ma w nazwie tytuł artykułu.

Rysunek 5: Struktura pobranych plików



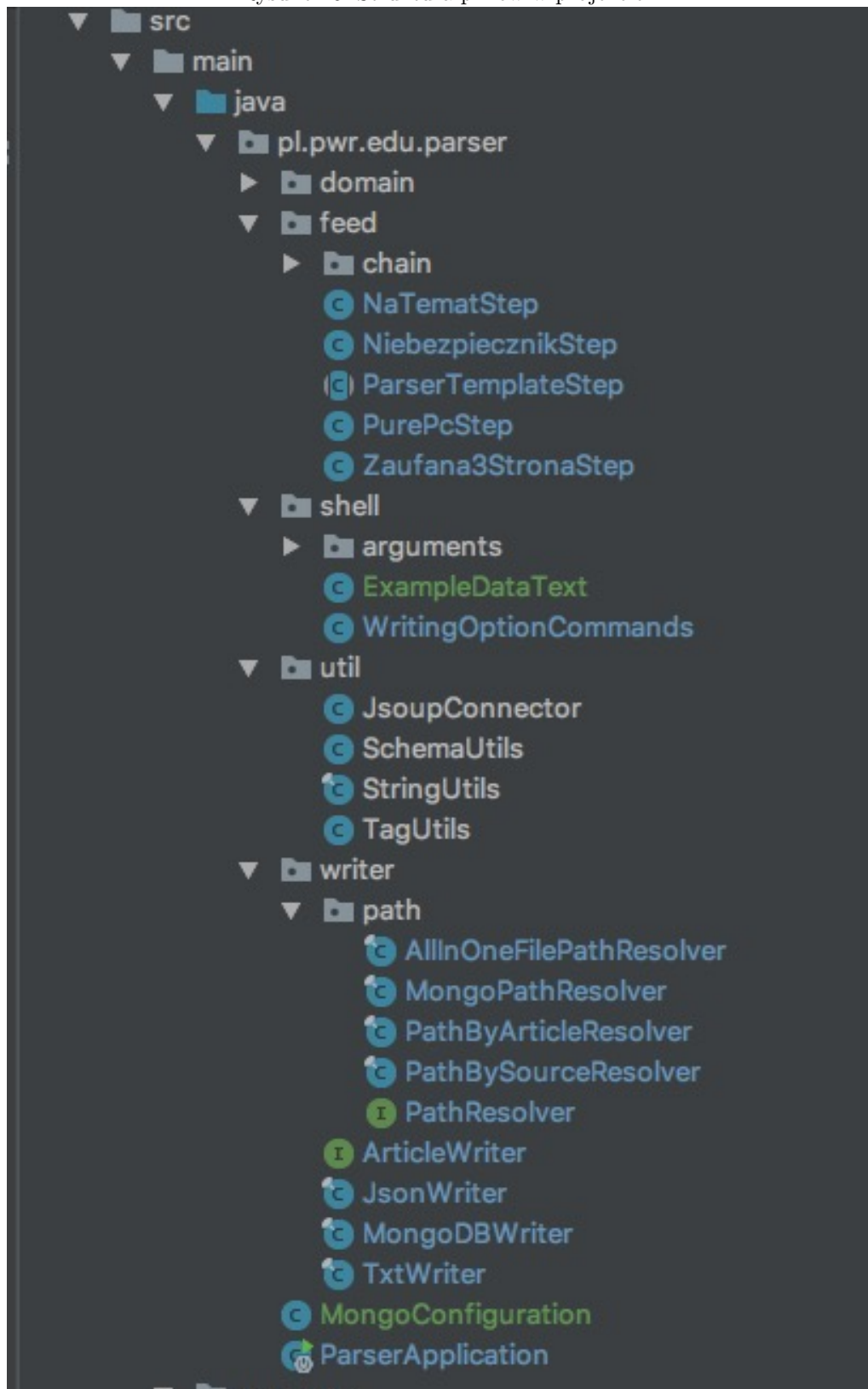
▼ output	Today at 19:34	--
3-dni-w-paryu-3-dni-w-pozn...-ekspressem-przez-europ.txt	Dec 4, 2017 at 15:49	17 KB
5-bdw-popenianych-przez-osoby-uczce-si-jzyka-obcego.txt	Dec 4, 2017 at 15:50	5 KB
6-grudnia-premiera-grand-championa.txt	Dec 4, 2017 at 15:49	6 KB
8-zawodw-przyszoci-ktrych-...-mona-nauczy-si-w-szkole.txt	Dec 4, 2017 at 15:49	11 KB
10-biegw-w-polsce-ktre-trze...na-bo-nie-zawsze-przebiec.txt	Dec 4, 2017 at 15:50	5 KB
10-gorcych-premier-i-procza-2015-r.txt	Dec 4, 2017 at 15:50	2 KB
10-ksiek-pod-choink.txt	Dec 4, 2017 at 15:49	1 KB
12-lat-temu-odbia-nam-palm...jkowskiej-zmienia-warszaw.txt	Dec 4, 2017 at 15:49	4 KB
12-zotych-za-calvina-kleina-n...zsto-nacigaj-jak-si-nie-da.txt	Dec 4, 2017 at 15:49	9 KB
13-miliona-euro-grantu-dla-polskiego-fizyka-wywiad.txt	Dec 4, 2017 at 15:49	12 KB
14-ksiek-2014-roku.txt	Dec 4, 2017 at 15:50	6 KB
15-z-za-wjazd-do-centrum-warszawy-zapacisz.txt	Dec 4, 2017 at 15:50	2 KB
40-dni-bez-zakrapianych-imp...koniecznie-stoi-tylko-wiara.txt	Dec 4, 2017 at 15:50	5 KB
82letnia-teresa-olszewska-ban...-chc-co-najmniej-do-90tki.txt	Dec 4, 2017 at 15:50	13 KB
92-pocigi-na-lsku-odwoane.txt	Dec 4, 2017 at 15:49	3 KB
46362314-co-dalej.txt	Dec 4, 2017 at 15:49	2 KB
a-gdyby-przesta-finansowa-universytety-z-podatkw.txt	Dec 4, 2017 at 15:49	4 KB
a-jednak-mike-dantoni-lakers-maj-trenera.txt	Dec 4, 2017 at 15:49	5 KB
a-ty-chodzisz-ju-ze-swoim-bl...-nowy-zwizkowy-fenomen.txt	Dec 4, 2017 at 15:50	5 KB
a-tymczasem-premier-gliski-iv.txt	Dec 4, 2017 at 15:49	4 KB
abolicja-zus-czyli-pastwo-promujce-zodziei.txt	Dec 4, 2017 at 15:50	3 KB
adam-maysz-komentatorem-e...ta-w-skokach-narciarskich.txt	Dec 4, 2017 at 15:49	2 KB
adam-sawiski-16latek-ktry-m...oje-zdanie-to-bd-je-wyraa.txt	Dec 4, 2017 at 15:49	7 KB
aden-lekarz-nie-bdzie-skaka-...zenie-coraz-popularniejsze.txt	Dec 4, 2017 at 15:50	6 KB
adrian-makar-wygrywa-mam-...zegrywa-z-polskim-hejtem.txt	Dec 4, 2017 at 15:49	3 KB
afrykascy-hipsterzy-ta-moda-po-prostu-trafia-wszdzie.txt	Dec 4, 2017 at 15:49	3 KB
agata-duda-ma-trzeci-rk-na-f...y-prezydenckiej-czy-iluzja.txt	Dec 4, 2017 at 15:49	1 KB
agnieszka-graff-pragnienie-d...uczucie-antyfeministyczne.txt	Dec 4, 2017 at 15:50	1 KB
agrigento.txt	Dec 4, 2017 at 15:50	3 KB

5 Implementacja aplikacji

5.1 Struktura plików w projekcie

Aplikacja została napisana w języku Java. Na rysunku 4 przedstawiono podział klas na pakiety.

Rysunek 6: Struktura plików w projekcie



5.2 Encja artykułu

Na rysunku 5 przedstawiono klasę modelującą artykuł. Każdy obiekt tej klasy posiada właściwości:

- source - URL strony z której został pobrany artykuł
- title - tytuł artykuł
- body - treści artykułu

Rysunek 7: Klasa modelująca artykuł

```
@Document
@Builder
@AllArgsConstructor
@NoArgsConstructor
@ToString(of = {"title", "source"})
public class Article {

    @Id
    private ObjectId id;

    private String source;
    private String title;
    private String body;
    private HashMap<String, String> metadata = new HashMap<>();

    @Version
    private Long version;

    public Article(String url) { this.source = url; }

    public Article setTitle(String title) {
        this.title = title;
        return this;
    }

    public Article setBody(String body) {
        this.body = body;
        return this;
    }

    public HashMap<String, String> getMetadata() { return metadata; }

    public void setMetadata(HashMap<String, String> metadata) { this.metadata = metadata; }

    public String getSource() { return source; }

    public String getTitle() { return title; }

    public String getBody() { return body; }
}
```

5.3 Wywołania parserów dla kolejnych witryn

Struktura aplikacji pozwala na łatwą rozbudowę o parsowanie kolejnych witryn. Każda witryna parsowana jest osobną klasą, która dziedziczy po klasie

Program w klasie ParserChain wywołuje polimorficznie metodę parse() dla każdej z klas dziedziczących po ParserTemplateStep.

Dlatego, aby obsłużyć kolejną stronę wystarczy zaimplementować klasę dziedziczącą po ParserTemplateStep oraz zaimplementować jej abstrakcyjną metodę parse(). Implementacja klasy ParserChain znajduje się na rysunku 6.

Rysunek 8: Klasa wywołująca kolejne implementacje *ArticleWriter*

```
@Component
public class ParserChain {

    private final List<ParserTemplateStep> parsingSteps;

    @Autowired
    public ParserChain(List<ParserTemplateStep> parsingSteps) { this.parsingSteps = parsingSteps; }

    public void invoke(ArticleWriter articleWriterArgument) {
        ArticleWriter articleWriter = Optional
            .ofNullable(articleWriterArgument)
            .orElseGet(this::getDefaultArticleWriter);

        parsingSteps.sort(AnnotationAwareOrderComparator.INSTANCE);
        parsingSteps.forEach(parserTemplateStep -> parserTemplateStep.setArticleWriter(articleWriter));

        System.out.println("Starting...");
        parsingSteps.forEach(ParserTemplateStep::parse);
    }

    private ArticleWriter getDefaultArticleWriter() {
        System.err.printf("Article writer not set, using default %s", JsonWriter.class.getName());
        String writePath = System.getProperty("user.dir");
        return JsonWriter.getInstance(writePath);
    }
}
```

6 Użyte technologie

- Java 8
- Spring Boot 1.5.9
- Spring Shell 2.13.7
- JSOUP
- Guava
- Jackson
- Apache Utils

7 Kod źródłowy oraz zebrane dane

Podczas testów aplikacja zdołała zebrać ponad 22 tysiące artykułów. Kod programu oraz zebrane dane są dostępne do pobrania za pośrednictwem usługi Google Drive.

- Kod programu: <https://tinyurl.com/source-pwr>
- Wersja wykonywalna: <https://tinyurl.com/aplikacja-jar-pwr>
- Wyniki: <https://tinyurl.com/output-pwr>

8 Wnioski

Prawidłowe wydzielenia tekstu ze strony internetowej jest zadaniem nietrywialnym, ale możliwym do osiągnięcia. Warto zauważyć, że praktycznie niemożliwe jest stworzenie generycznego systemu do wszystkich stron. Każdy dokument html różni się budową i fizycznym umieszczeniem treści artykułu w XPath.