УДК 004.62

І. В. Стешов, *програмний інженер у «Глобаллоджик Юкрейн»*
вул. Нікольська, 25, м. Миколаїв, Україна
ivanlucky22@gmail.com
М. Т. Фісун, *д.т.н., професор*
Чорноморський державний університет ім. Петра Могили
вул. 68 Десантників, 10, м. Миколаїв, Україна
mykola.fisun@gmail.com

РОЗРОБКА ПАРСЕРА ЛІТЕРАТУРНИХ ДЖЕРЕЛ ЗАСОБАМИ ANTLR ТА HIBERNATE ДЛЯ ФОРМУВАННЯ БІБЛІОГРАФІЧНОЇ БАЗИ ДАНИХ

Розглядаються правила бібліографічного опису літературних джерел, розпізнаються основні частини, з яких складаються джерела. На основі цих частин створюється структура бази даних та зв'язки між таблицями. Пояснюється мета фреймворка Hibernate та технологій, що використовуються поряд з ним. Розглядаються лексичний та синтаксичний аналізи та їх взаємодія у процесі розпізнавання. Наводяться правила для реалізації парсера літературних джерел та інструмент виконання правил – генератор парсерів ANTLR.

Ключові слова: *бібліографічний опис, літературний ресурс, база даних, HSQLDB, Hibernate, Spring Data, ANTLR, парсер, лексер.*

Вступ. Однією з трудомістких процедур в інформаційних бібліотечних системах (ІБС) є каталогізація літературних джерел інформації [1–3]. Тому в ІБС зазвичай окремо виділяють підсистему «Каталог». Відповідні працівники бібліотеки після надходження видання в бібліотеку заповнюють форми за стандартами (правилами) бібліографічного опису видань [4, 5]. Але певну частину інформації для такої форми можна отримати автоматизовано шляхом синтаксичного аналізу тексту бібліографічного опису за відповідним стандартом [6]. Тому цей напрям автоматизації в ІБС можна вважати актуальним. У статті пропонується один із підходів для вирішення такого завдання.

Розробку парсера літературних джерел засобами ANTLR та Hibernate для формування бази даних можна виконати як шляхом розробки «власного» програмного забезпечення, використовуючи складні регулярні вирази, так і за допомогою інструментальних засобів побудови компіляторів, таких як JavaCC, BIZON, YACC, ZUBR [7], ANTLR [9] тощо. В результаті проведеного аналізу обрано генератор парсерів ANTLR, виходячи з того, що автори вже мали досвід розробки кількох компіляторів на цій платформі [10]. Питання автоматизації бібліотек розглядалося багатьма дослідниками. Вітчизняний вчений М. Слободяник вважає, що запровадження інноваційної технології спрямоване на «підвищення дієвості бібліотеки» [11].

Польським дослідником А. Радванським у статті «Biblioteki w nowoczesnym spoleczenstwie» [12] аналізується досвід застосування новітніх технологій у бібліотеках Польщі та наголошується, що автоматизація бібліотечних процесів та створення користувачам можливостей доступу до інформації в усіх її сучасних формах є однією з нагальних проблем бібліотек у сучасному суспільстві.

Постановка задачі. Основна ідея проекту полягає у створенні програми, яка зможе розпізнати бібліографічні описи літературних ресурсів та заповнити ними базу даних.

Основні *завдання*, що потребують вирішення в процесі роботи, пов'язані зі створенням структури бази даних, відображенням таблиць з бази даних на Java класи, створенням правил для парсингу списку літературних джерел. Існує низка варіантів можливих бібліографічних описів: окрім наукових статей та книг, існують такі види, як збірники, словники, атласи, підручники, каталоги, електронні ресурси, законодавчі матеріали, стандарти та ін. Враховуючи те, що розпізнають вісім основних областей опису, де не всі області є обов'язковими та кожна область може

поділятися на декілька підобластей, то множина можливих структур бібліографічних описів буде дуже значною, і час на створення універсального програмного забезпечення (ПЗ), яке б змогло розпізнати будь-який опис, може бути довгим. Тому, ще однією проблемою буде вибір підмножини правил, яка буде підтримувати систему, що розроблюється.

Правила складання бібліографічних описів у переліку посилань. Бібліографічні описи літературних джерел [13] виконуються відповідно до національного стандарту ДСТУ ГОСТ 7.1:2006, який набув чинності 1 липня 2007 р. на заміну ГОСТ 7.1–84 та є міждержавним стандартом для 10 країн. Новий державний стандарт покликаний забезпечити впровадження сучасних автоматизованих технологій опрацювання документів, ведення інформаційних баз даних; ефективність пошуку та використання документів усіх видів і типів.

До бібліографічного опису входять такі вісім областей: область заголовку та відомостей про відповідальність; область видання; область специфічних відомостей; область вихідних даних; область фізичної характеристики; область серії; область приміток; область стандартного номера (чи його альтернативи) та умов доступності.

Області опису складаються з елементів, котрі підрозділяються на обов'язкові та факультативні. В описі можуть бути лише обов'язкові елементи або обов'язкові та факультативні (необов'язкові).

Структура бази даних зображена на рис. 1. Для відображення джерел достатньо лише однієї таблиці, де повна назва ресурсу була б унікальним ключем, але для того щоб мати гнучкість та зручність [14, 15] у використанні програми, було прийнято рішення виділяти авторів та видавництва в окремі таблиці.

Рис. 1. Структура бази даних літературних джерел

Ця особливість зробить можливим перегляд статей у розрізі авторів та видавництв. Таким чином, якщо користувач забуде назву ресурсу, але буде пам'ятати автора, він легко зможе знайти публікацію, просто змінивши відображення ресурсів на відображення у розрізі авторів. Таблиця «Автор» має унікальний ключ, унікальний ПІП та список його публікацій. Таблиця «Публікація» має унікальний ключ, список авторів, що працювали над нею, назву, позначення, місто публікації видання, рік видання, кількість сторінок, серію та випуск. Таблиця «Видавництво» має унікальний ключ, назву, місто в якому видавництво розташовано, рік його створення, та список публікацій, що були видані за участю цього видавництва. Інші бібліографічні описи, які можуть зустрітися, поки що проігноровані, але додадуться у наступних версіях програми.

Технології, що були використанні під час створення проекту:

1. Java 8 – основна мова програмування;
2. Spring Framework 4 – Inversion of Control, контейнер, контроль за транзакціями до бази даних, тестування;
3. HSQLDB – база даних;
4. Hibernate 4 – JPA implementation, ORM технологія для відображення таблиць база даних у вигляді Java класів;
5. ANTLR 4 – генератор парсерів;
6. JavaFX – основний інструмент для графічного інтерфейсу;
7. Tiwulfx – бібліотека для графічного інтерфейсу, що має спеціальний компонент для представлення тих класів, що відображають таблиці з бази даних;
8. Log4j – бібліотека для логування;
9. Maven – технологія для складання проекту та завантаження залежностей бібліотек з глобальних репозиторіїв;
10. Junit – бібліотека для проведення тестування.

Структура парсера літературних джерел зображена на рис. 2, з якого видно, що для автоматизації розробки лексичного аналізатора було використано одну з найбільш популярних систем – ANTLR, вхідною мовою якого є регулярні вирази [9]. ANTLR, як і багато інших подібних засобів, складається з бібліотеки класів, що полегшують основні операції при розборі (буферизація, пошук і т. д.), і утилітів, які генерують код парсера на основі файлу, що

описує граматику мови, яка розбирається. Сам ANTLR написаний мовою Java [9], але дозволяє генерувати код Java, C++, C#, JavaScrip та ще деякими популярними мовами.

Програма побудована на платформі Spring Framework, що дозволяє контролювати життєвий цикл усіх об'єктів та має гнучку конфігурацію усього додатка, а також розгорнуту конфігурацію, що налаштовує взаємодію з базою даних.

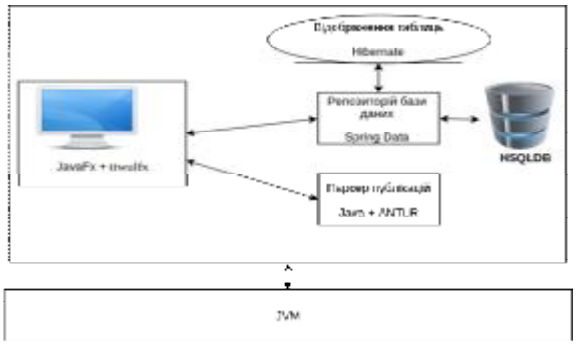

Рис. 2. Загальна структура парсера літературних джерел

Особливістю фреймворка Hibernate є реалізація відображення зі зв'язками таблиць з бази даних у Java об'єкти (ORM – Object-Relational Mapping). Такі Java класи називаються Entity і зазвичай не повинні містити ніякої логіки, вони мають тільки описувати об'єкт, тобто бути тим, що називається POJO (Plain Old Java Object) або Java Bean. Spring конфігурація налаштована таким чином, що база даних створюється за вже існуючими Entity, а не навпаки.

Spring Data є основним шаром, що зв'язує Entity з базою даних, його особливість полягає в економії часу та коду при написанні запитів. Більшість загальних запитів до бази вже реалізовані у цій бібліотеці, а для того щоб додати новий простий запит, наприклад вибірка по полю, потрібно просто описати сигнатуру методу, дотримуючись конвенції, в інтерфейсі без її реалізації.

Лексичний аналіз. Основна задача лексичного аналізу – розбити вхідний текст, що складається з послідовності одиночних символів, на послідовність слів чи лексем, тобто виділити ці слова з безперервної послідовності символів [4]. Усі символи вхідної послідовності, з цієї точки зору, розділяються на символи, що належать яким-небудь лексемам, і символи, що поділяють лексеми (роздільники).

Як джерело даних використовується клас потоку, який дозволяє розбирати дані, що знаходяться у файлі, в пам'яті процесу або приходять через мережу [7]. Результатом роботи ANTLR є два класи – лексер і парсер. Лексер розбиває потік символів на потік токенів відповідно до правил, а парсер обробляє потік токенів відповідно до інших правил. ANTLR відноситься до так званих LL(*)-аналізаторів – не обмежених скінченною кількістю лексем для попереднього перегляду, а здатних приймати рішення, визначаючи, чи належать вхідні лексеми регулярній мові.

Лексичний аналізатор, генерований ANTLR, взаємодіє з синтаксичним аналізатором таким чином: при виклику його синтаксичним аналізатором лексичний аналізатор посимвольно читає залишок входу [16], поки не знаходить найдовший префікс, що може бути зіставлений одному з регулярних виразів. Потім він виконує відповідну дію. Як правило, дія повертає керування синтаксичному аналізатору. Якщо це не так, тобто у відповідній дії немає повернення, то лексичний аналізатор продовжує пошук лексем доти, поки дія не поверне керування синтаксичному аналізатору. Повторний пошук лексем аж до явної передачі керування дозволяє лексичному аналізатору правильно обробляти проміжки та коментарі. Синтаксичному аналізатору лексичний аналізатор повертає єдине значення – тип лексеми [17].

У файлі правил ANTLR усі лексичні правила повинні починатися з великої літери. Ключовим словом «*fragment*» позначаються найменші неподільні лексичні правила, або токени. Наведемо частину ANTLR-файлу, що реалізує функцію лексичного аналізу парсера літературних джерел:

```
DASH: '–';//'\u2013';
PYPHEN:'-';//'\u002d';
INT : [0-9]+ ;
SYMBOL: '.' | ''| PYPHEN;
LETTER : [a-zA-Z\u0400-\u04FF];
WS : [\r\t\n]+ -> skip ;
ErrorCharacter : . ;
```

У назві ресурсу або інформації про нього можуть бути крапки або дефіси і при розборі тексту треба розпізнавати тире – як символ відокремлення однієї частини опису від іншої і дефіс – як частину опису. Ці два

символи у мові Java та у правилах ANTLR виглядають однаково, але мають різні юнікоди: тире – це '\u002d', дефіс – '\u2013'.

Синтаксичний аналіз. Ієрархічний аналіз називається розбором (parsing) або синтаксичним аналізом, котрий включає групування токенів початкової програми в граматичні фрази, які використовуються компілятором (чи інтерпретатором) для синтезу виводу [16].

Ієрархічна структура програми зазвичай виражається рекурсивними правилами. Наприклад, при обумовленні виразів можна дотримуватись таких правил:

Будь-який ідентифікатор (*identifier*) є виразом (*expression*).

Будь-яке число (*number*) є виразом (*expression*).

Якщо *expression1* та *expression2* є виразами, то виразами також є:

expression1 - expression2 *(1)*
expression1 expression2* *(2)*
(expression1). *(3)*

Правила (1) і (2) є базовими (нерекурсивними), в той час як (3) обумовлює вираз за допомогою операторів, які застосовуються до інших виразів.

У файлі правил ANTLR всі синтаксичні правила починаються з маленької букви та описуються за допомогою граматики у вигляді форми Бекуса-Наура (БНФ) [17]. Результатом роботи ANTLR є два Java класи – *Lexer* та *Parser*.

Ядром ANTLR-специфікації є набір граматичних правил. Кожне правило описує синтаксичну конструкцію і дає їй ім'я:

```
parse returns [List<PublicationView> publications]
@init{$publications = new
LinkedList<PublicationView>();}
:
   ( INT ')' publication '.'
{$publications.add($publication.publicationView);} )+
   EOF;
publication returns [PublicationView publicationView]
@init{$publicationView = new PublicationView();}
:
  authors
{$publicationView.setАвтори($authors.authorsStr.trim());}
  sentence
{$publicationView.setНазва($sentence.text.trim());}
  (':' marking
{$publicationView.setПозначення($marking.text.trim());})
?
  ('/' responsible
{$publicationView.setВідповідачі($responsible.text.trim());
})?
  DASH city
{$publicationView.setМісто($city.text.trim());}
  (':' edition
{$publicationView.setВидання($edition.text.trim());})?
  (',' INT '.'
{$publicationView.setРік($INT.text.trim());} )
  (DASH INT 'c.'
{$publicationView.setСторінок($INT.text.trim());} )
  (DASH  '(' series ';'
{$publicationView.setСерія($series.text.trim());}
  'вип. №' INT ')'
{$publicationView.setВипуск($INT.text.trim());} );
name: sentence;
author: word ',' LETTER '.' LETTER '.' ;
city: word;
edition: sentence;
series: sentence;
marking: sentence;
responsible: sentence;

authors returns [String authorsStr]
@init{  StringBuilder sb = new StringBuilder();  }
@after{ $authorsStr = sb.toString();             }:
    author
{sb.append($author.text).append(Constants.AUTHORS_DE
LIMITER);}
    (',' author
{sb.append($author.text).append(Constants.AUTHORS_DE
LIMITER);})*;
sentence  returns [String text]
@init{  StringBuilder sb = new StringBuilder();  }
@after{ $text = sb.toString();  }:
  ( signedWord  {sb.append($signedWord.text);sb.append("
");})+;
signedWord: (word | SYMBOL)+;
word: LETTER+;
```

У кожному правилі можна використовувати звичайний Java-код, який можливо писати тільки у фігурних дужках. Також ANTLR надає можливість оперувати його ж змінними, ставлячи знак «$» на початку змінної [9]. Завдяки цьому коду можна зрозуміти процес трансляції виразів. Головним правилом, з якого починається процес трансляції, є правило *parse,* що повертає список об'єктів PublicationView, які є відображенням таблиці з точки зору JavaFX таблиці. Правило *publication* поділено на підправила, які відповідають областям бібліографічного опису.

Таким чином, згідно з продемонстрованими правилами парсер враховує, що у публікації можуть бути кілька авторів, назва, позначення, список відповідальних осіб або наукові заходи (конференції та ін.), місто, видання, рік, кількість сторінок, серія і випуск. Не усі області є обов'язковими, тому після деяких підправил стоїть знак «?».

Для демонстрації розглядається приклад з чотирма літературними ресурсами, але в цих ресурсах заповнені не всі області. На рис. 3 добре видно вхідні параметри: коли все готово для розбору, треба натиснути кнопку «Парсити», після чого таблиця у нижній половині екрану заповниться. Оскільки розпізнані публікації ще не збережені, то поле Id в усіх буде дорівнювати нулю. На рис. 4 зображено продовження таблиці, з якої помітно, що у деяких публікаціях відсутні деякі бібліографічні описи. Наприклад, ресурси 3 та 4 не мають відповідальних осіб, серії та номеру випуску, 3-й ресурс також не має ще й видання, а другий, хоч і не має серії та номеру, зате має перелік відповідальних осіб, на відміну від інших ресурсів.

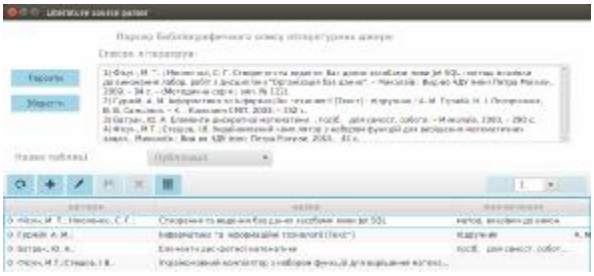

Рис. 3. Вигляд розпарсених публікацій

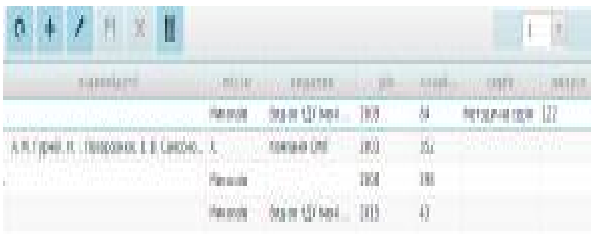

Рис. 4. Продовження таблиці розпарсених публікацій

Для того щоб оброблені джерела зберегти в базу даних, потрібно натиснути кнопку «Зберегти» і у таблиці з'явиться трохи інше відображення цих же джерел: з'явиться унікальний ідентифікатор та автори разом з видавництвами для кожної публікації будуть перелічені у стовпчик (рис. 5).

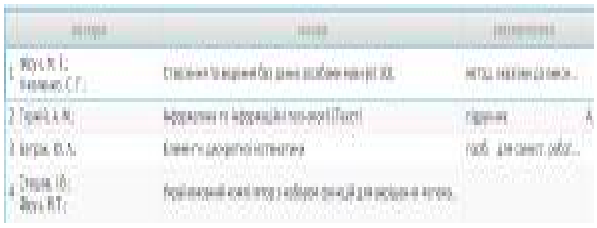

Рис. 5. Вигляд збережених публікацій

Після збереження публікацій заповнюється не лише таблиця «Публікації», але й таблиці «Автори» та «Видавництва» і тепер, оскільки база даних не порожня, то можна проглянути інформацію у розрізі авторів та у розрізі видавництв, для чого треба вибрати відповідну таблицю у компоненті комбобокс поряд з написом «Назва таблиці». Якщо обрати таблицю з авторами (рис. 6), то з'явиться таблиця, в якій автори будуть унікальними, а у полі публікації може бути декілька або, якщо заповнювати базу власноруч, жодної публікації. Для зручності також додане поле «кількість публікацій», яке дозволить краще збирати статистику.

Так само, як і для таблиці авторів, таблиця з видавництвами також має поле з унікальними назвами видавництв. Програма автоматично враховує, що місто, в якому стаття опублікована, і є рідним містом видавництва. У прикладі публікації під номерами 1 та 4 були видані у видавництві ЧДУ імені Петра Могили, що програма і демонструє на рис. 7.

Рис. 6. Відображення таблиці авторів

Також для того, щоб пришвидшити пошук, публікації кожного видавництва відображаються у стовпчик. Рік створення видавництва не належить до бібліографічного опису літературного ресурсу, тому це поле, при існуванні потреби, можна вписати власноруч, натиснувши на олівець на табличній панелі.

Рис. 7. Відображення таблиці видавництв

Усі таблиці можна редагувати та доповнювати, не використовуючи парсер, тому що на табличній панелі є всі необхідні інструменти. Якщо парсер робить помилку, то її легко виправити, просто змінивши запис

після натиснення на іконку олівця. За додавання запису відповідає іконка з плюсом, а за видалення – з червоним хрестиком.

Інтерфейс програми створено на основі JavaFX бібліотеки, що вбудована у JDK, починаючи з восьмої версії. В цій бібліотеці є всі необхідні примітиви інтерфейсу сучасного графічного програмного продукту. Більш того, графічний інтерфейс можливо створювати у спеціальному *fxml* файлі, що дуже нагадує створення веб-сторінки засобами HTML. Також компоненти JavaFX підтримують CSS для стилізації. На додаток до FXML використовується бібліотека Tiwulfx, саме в якій і є графічний компонент таблиці, що має функціонал для відображення Entity об'єктів.

Висновки. Засобами компілятора ANTLR розроблено синтаксичні правила для розбору літературних джерел відповідно до стандартів бібліографічного опису [1]. Використовуючи базу даних HSQLDB та мову Java з технологіями, що застосовуються разом з нею для спілкування з базою даних (Hibernate, Spring Data), створено засоби для зберігання літературних ресурсів і зручний графічний інструмент для заповнення та модифікування даних у базі даних.

Планується запропонувати використання такої системи як додатковий модуль ІБС університету.

Список літератури

1. Использование компьютерных технологий в библиотечно-информационном обслуживании предприятия [Электронный ресурс]. – Режим доступа : URL : http://referat-ok.com.ua/kulturologiya-ta-mistectvo/vikoristannya-komp096yuternih-tehnologii-v-bibliotechno-informaciinomu-obslugovuvanni-pidprijemstva.
2. Карпенко І. Інформаційно-бібліотечна система UFD / І. Карпенко // Бібліотечний форум України. – 2003. – № 2. – С. 15–17.
3. Доценко С. Інтегрована бібліотечна система ALEPH 500 / С. Доценко. // Бібліотечний форум України. – 2004. – № 1.– С. 11–14.
4. Воройский Ф. С. Основы проектирования автоматизированных библиотечно-информационных систем / Ф. С. Воройский. – М. : Физматлит, 2002.
5. UNIMARC to MARC 21 Conversion Specifications [Internet].– Available from : URL: http://www.loc.gov/marc/unimarctomarc21.html
6. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления : (ГОСТ 7.1-2003, IDT) : ДСТУ ГОСТ 7.1 : 2006. – [Взамен ГОСТ 7.1-84]. – [Действующий с 2007-07-01]. – М. : Госпотребстандарт Украины , 2007. – 47 с.
7. Публикации. Новые правила библиографического описания [Электронный ресурс] / [А. Устинникова, П. Сенько, С. Юлдашева и др.] // ДСТУ ГОСТ 7.1 : 2006 в Украине ; Гос. науч. учреждение «Кн. Палата Украины им. Ивана Федорова». – Режим доступа : URL : http://www.ukrbook.net/DSTU.htm. – Загол. с экрана.
8. Костелтсев А. В. Построение компиляторов и интерпретаторов. Использование программ BIZON, BYACC, ZUBR / А. В. Костелтсев. – С.Пб : Наука и техника, 2001. – 219 с.
9. Terence Parr: Definitive ANTLR Reference. «ANTLR для разработки компилятора на языке Java» [Электронный ресурс]. – Режим доступа : URL : https://vimeo.com/25753384. – Загол. с экрана.
10. Стешов І. В. Україномовний компілятор з набором функцій для вирішення математичних задач / І. В. Стешов, М. Т. Фісун // Інтелектуальні інформаційні системи : Всеукр. наук.-практ. конф. молодих вчених, аспірантів і студентів. – Миколаїв : ЧДУ ім. П. Могили, 2015. – С. 43–44.
11. Слободяник М. Наукова бібліотека: еволюція структури і функцій / М. Слободяник. – К., 1995. – С. 140.
12. Radwanski A. Biblioteki w nowoczesnym spoleczenstwie / A. Radwanski // Bibliotekarz. – 2007. – Nr. 11. – S. 4–7.
13. Вікіпедія: Посилання на джерела [Електронний ресурс]. – Режим доступу : URL : https://uk.wikipedia.org/wiki/Вікіпедія:Посилання_на_джерела. – Загол. з екрану.
14. Дейт Кристофер Дж. Введение в систему баз данных / Кристофер Дж. Дейт. – М. : Издат. дом «Williams», 2001. – 1071 с.
15. Пасічник В. В. Організація баз даних та знань : підручник для ВУЗів / В. В. Пасічник, В. А. Резніченко. – К. : БХВ, 2006. – 384 с.

16. Kompiliatory: printsypy, technlogii, instrumenty : Pereklad z angl. / Alfred V. Aho, Monica S. Lam, Ravi Seth i dr. – M. : Издат. дом «Williams», 2001. – 768 с.
17. Богданов В. Л. Практичний доступ написання синтаксичного аналізатора мови програмування Cobol, 2000 / В. Л. Богданов, В. С. Гордеєв. – 7 с.

References

1. Ispol'zovaniye komp'yuternykh tekhnologiy v bibliotechno-informatsionnom obsluzhivanii predpriyatiya [Elektronnyi resurs]. Regym dostypa: URL: http://referat-ok.com.ua/kulturologiya-ta-mistectvo/vikoristannya-komp096yuternih-tehnologii-v-bibliotechno-informaciinomu-obslugovuvanni-pidprijemstva.
2. Karpenko, I. (2003) Informatsiyno-bibliotechna systema UFD. *Bibliotechnyy forum Ukrainy*, (2), s. 15–17 [in Ukrainian].
3. Dotsenko, S. (2004) Integrovana bibliotechna systema ALEPH 500. *Bibliotechnyy forum Ukrainy*, (1), s. 11–14 [in Ukrainian].
4. Voroyskiy, F. S. (2002) Osnovy proyektirovaniya avtomatizirovannykh bibliotechno-informatsionnykh sistem. Moscow: Fizmatlit [in Russian].
5. UNIMARC to MARC 21 Conversion Specifications [Internet]. Available from: URL: http://www.loc.gov/marc/unimarc-tomarc21.html
6. Sistema standartov po informatsii, bibliotechnomu i izdatel'skomu delu. Bibliograficheskaya zapis'. Bibliograficheskoye opisaniye. Obshchiye trebovaniya i pravila sostavleniya (2007) : (GOST 7.1-2003, IDT) : DSTU GOST 7.1: 2006. [Vzamen GOST 7.1-84]. [Deystvuyushchiy s 2007-07-01]. Moscow: Gospotrebstandart Ukrainy, 47 s. [in Russian].
7. Ustinnikova, A., Sen'ko, P., Yuldasheva, S. i dr. Publikatsii: Novyye pravila bibliograficheskogo opisaniya [Elektronnyy resurs]. DSTU GOST 7.1: 2006 v Ukraine; Gos. nauch. uchrezhdeniye "Kn. palata Ukrainy im. Ivana Fedorova". Rezhim dostupa: URL: http://www.ukrbook.net/DSTU.htm. – Zagol. s ekrana.
8. Kosteltsev, A. V. (2001) Postroeniye interpretatorov i kompiliatorov. Ispolzovaniye programm BIZON, BYACC, ZUBR. St. Petersburg: Nauka i tehnika, 219 s. [in Russian].
9. Terence Parr: Definitive ANTLR Reference. «ANTLR dlia razrabotki kompiliatora na jadzyke Java» [Elektronnyi resurs]. – Regym dostypa: URL: https://vimeo.com/25753384. – Zagol. s ekrana.
10. Steshov, I. V., Fisun, M. T. (2015) Ukrainomovnyi kompiliator z naborom funktsiy dlia vyrishennia matematychnyh zadach. Intelectualni informatciyni systemy: Vseukr. nauk.-pract. konf. molodyh vchenyh, aspirantiv i studentiv. Mykolaiv: ChDU im. P. Mogyly, s. 43–44 [in Ukrainian].
11. Slobodyanyk, M. (1995) Naukova biblioteka: evolyutsiya structury i funktsiy. Kyiv, s. 140. [in Ukrainian].
12. Radwanski, A. (2007) Biblioteki w nowoczesnym spoleczenstwie. *Bibliotekarz*, (11), s. 4–7.
13. Vikipediya: Posylannya na dzherela [Elektronnyi resurs]. Regym dostypy: URL: https://uk.wikipedia.org/wiki/Posylannya_na_dzherela Zagol. z ekranu
14. Date, Christopher J. (2001) Vvedeniye v sistemu baz dannyh Moscow: Izdat. dom «Williams», 1071 s. [in Russian].
15. Pasichnik, V. V., Reznichenko, V. A. (2006) Organizatcia baz danyh ta znan. Kyiv: BHV, 384 s. [in Ukrainian].
16. Aho, Alfred V., Lam, Monica S., Sethi, Ravi i dr. (2001) Kompiliatory: printsypy, technlogii, instrumenty.: Pereklad z angl. Moscow: Izdat. dom «Williams», 768 s.
17. Bogdanov, V. L., Gordeev, V. S. Practychnyy dostup napysannya syntaksychnogo analizatora movy programuvannia Cobol, 2000, 7 s. [in Ukrainian].

I. V. Steshov, software engineer at «Globallogic Ukraine»,
Nikolska str., 25, Mykolayiv, Ukraine
ivanlucky22@gmail.com
M. T. Fisun, *D.Sc., professor*
doctor of technical science, professor
Black Sea State University named after Petro Mohyla,
68 Desantnykiv str., 10, Mykolayiv, Ukraine
mykola.fisun@gmail.com

DEVELOPMENT OF LITERATURE RESOURCES PARSER USING ANTLR AND HIBERNATE FOR BIBLIOGRAPHIC DATABASE FORMATION

Introduction. *Cataloging of literary sources is one of tedious procedures of library information systems (LIS). Therefore, separate subsystem "Catalogue" is usually allocated in LIS. Corresponding library staff after receiving publications in library fill forms by the standards (rules) of bibliographic description of publications. But some of the information for such a form can be automated by parsing the text of bibliographic description by the relevant standard. That's why automation in LIS can be considered as actual.*

Purpose. *The main goal of the work is to create syntactic and semantic rules and a program as a whole that can recognize bibliographic descriptions of literary resources and fill it's database.*

The problem. *The main problem described in the article is the creation of database structure, performing Object Relational Mapping (ORM) with Java classes, creating syntactic and semantic rules to recognize the list of references.*

Main material. *To create any application first one needs to do the subject exploring and to determine technologies to be used. Afterwards – application planning and specifications. On the first step bibliographic resources of writing rules and their variations have been explored. It is also determined the next stack of technologies: programming language Java; Spring Framework as a platform for application; HSQLDB – embedded database that can be created in the runtime of the program and does not require any additional resources; Hibernate – implementation of JPA specification in Java Enterprise Edition, which allows to describe Java classes as if every class is a table (ORM), to establish relationships between classes and perform all CRUD operations interacting exclusively with Java objects without touching the database directly; Spring Data – a framework that is designed to connect any implementation of JPA (in our case – Hibernate) and database; ANTLR as a tool for generating parser based on special rules, similar to Backus-Naur notation form; JavaFX, a library as the GUI; and some less important tools. Further, it is determined that is is possible to perfectly describe any possible variation of bibliographic description, but it takes a long time to perform that, so it has been decided to describe the most common variation references and have the following elements: an area of title and statement of responsibility; an area of publication; an area of specific information; output data area; an area of physical characteristics of the region; series area; an area of notes. Not all areas are mandatory in bibliographic description required and this also is taken into account.*

As a result, a program that takes as an input a list of references and as output demonstrates the same data, but in a structured way, stored in three tables ("Edition", "Author", "Publication"), is developed. The application allows to view data in three sections: from the author's, addition's and publication's point of view, making search and view of bibliographic information much easier.

In the article an approach for automation of recognition is offered, this approach can be used to structure and store in a database any full article, magazine, book or any other text information.

Keywords: *bibliographic description, reference, database, HSQLDB, Hibernate, Spring Data, ANTLR, parser, lexer.*

Рецензенти: М. П. Мусієнко, д.т.н., професор,
І. І. Коваленко, д.т.н., професор