# ДЕКЛАРАТИВНЫЙ ЯЗЫК ОПИСАНИЯ АЛГОРИТМОВ ИЗВЛЕЧЕНИЯ ДАННЫХ ИЗ XML-ДОКУМЕНТОВ

**Алексей Владыкин**
*Санкт-Петербургский государственный университет информационных технологий, механики и оптики, Санкт-Петербург*
*e-mail: vladykin@gmail.com*

**Аннотация**

Предложен непроцедурный язык для описания обработчиков XML-документов, который целесообразно применять для извлечения данных из этих документов. Обработчики реализованы на основе конечных автоматов, что упрощает их структуру и понимание.

## 1. ВВЕДЕНИЕ

В настоящее время язык XML [1] широко используется для разнообразных целей: от хранения настроек программ до описания векторных изображений. Поэтому все больше программ требуют поддержки этого языка, а программистам все чаще приходится писать код, который считывает XML-документ и выделяет из него фрагменты, представляющие тот или иной интерес.

Нередко встречаются XML-документы размером в десятки и сотни гигабайт. Один из таких документов — полная база данных Википедии, доступная для скачивания в виде XML-файла [2]. Размер этого файла, включающего все статьи Википедии с историей их изменений, в сжатом архиватором bzip2 виде составляет 148 гигабайт.

Как прочитать такой документ и извлечь из него полезную информацию, например, число перекрестных ссылок между статьями? Единственным вариантом в данном случае является использование технологии Simple API for XML (SAX) [3] и написание соответствующего обработчика, который позволит решить поставленную задачу. Известны также и другие технологии доступа к XML-данным, например Document Object Model (DOM) [3] или Java API for XML Binding (JAXB) [4], но они требуют загрузки всего документа в оперативную память. Это при таких размерах документов не представляется возможным.

Недостатком технологии SAX, затрудняющим его использование, является сложность написания кода для реализации указанных обработчиков. В данной работе предложена технология, позволяющая значительно упростить написание обработчиков этого класса. При этом предложен непроцедурный язык описания поведения SAX-об-

работчика. Обработчик реализуется на основе автоматного подхода, его код генерируется автоматически.

Насколько известно автору, идея автоматической генерации SAX-обработчиков для выборочного разбора XML-документов предлагается впервые. Однако следует отметить несколько работ, более или менее близких к данной теме.

В работе [5] предпринята попытка преодолеть сложности работы с SAX при помощи функционального языка и декларативных описаний. Для императивных языков в работе [6] предложена идея автоматической генерации эффективных SAX-подобных разборщиков по заданной XML схеме. Авторы работы [7] применили конечные автоматы для фильтрации XML-документов по заданному множеству XPath-выражений с использованием SAX-разбора.

## 2. SIMPLE API FOR XML

SAX — это стандартный способ разбора XML, при котором XML-разборщик (XML parser) последовательно читает XML-документ и уведомляет пользовательский код (SAX-обработчик, SAX handler) о каждом встречающемся открывающем или закрывающем теге и о текстовом контенте между ними.

Стандартный SAX-обработчик на языке Java расширяет класс DefaultHandler и переопределяет следующие три метода:

```
void startElement(String uri, String localName,
String qName, Attributes attrs) throws
SAXException;
void endElement(String uri, String localName,
String qName) throws SAXException;
void characters(char[] ch, int start, int length)
throws SAXException;
```

Метод `startElement` вызывается разборщиком, когда тот обнаруживает открывающий тег, а метод `endElement` — в ответ на закрывающий тег, и, наконец, метод `characters` — для текстового контента между тегами.

## 3. ТЕКСТОВЫЙ ЯЗЫК ОПИСАНИЯ SAX-ОБРАБОТЧИКОВ

Рассмотрим задачу извлечения списка персоналий из таблицы, находящейся в XHTML-документе известной структуры. XHTML — это HTML, соответствующий всем правилам XML. Поэтому для разбора

XHTML-документа можно применить технологию SAX. Эта задача выбрана в качестве примера, поскольку на ней можно продемонстрировать сложность написания SAX-обработчика при традиционном подходе к программированию.

Пусть алгоритм извлечения списка персоналий выглядит следующим образом:

- найти таблицу (`<table>`) с атрибутом `class="header"`;
- найти далее таблицу без атрибута `class`;
- в таблице без атрибута пропустить первую строку (`<tr>`) и все строки, у которых первая ячейка имеет атрибут `class="banner"`;
- из оставшихся строк взять вторую и четвертую ячейки (`<td>`) как фамилию и имя соответственно.

В соответствии с предлагаемым подходом, опишем SAX-обработчик на непроцедурном текстовом языке, специально разработанном для этой цели.

Язык этого описания состоит из следующих элементов:

- открывающий и закрывающий теги, являющиеся для обработчика входными воздействиями. Открывающий тег может иметь ограничение на значения атрибутов (например, `<td class=="banner">`). Выходное воздействие задается после тега в фигурных скобках (`<td> { object.newPerson(); }`);
- скобки, используемые для группировки тегов.

Со скобками могут использоваться следующие операции:

- альтернатива: (...|...);
- повторение: (...)*
- опциональность: (...)?

Описание на предлагаемом языке SAX-обработчика, реализующего приведенный выше алгоритм, имеет вид:

```
<table class=="header"> </table>
<table class==null>
<tr> </tr>
(
<tr>
(
<td class=="banner"> </td> |
<td> { object.newPerson(); } </td>
<td> { capture(); } </td> { object.
setLastName(captured()); }
<td> </td>
```

```
<td> { capture(); } </td> { object.setFirstName(c
aptured()); }
)
</tr>
)*
</table>
```

В соответствии с автоматной парадигмой [8], обработчик будет состоять из двух частей: объекта управления и автомата управления.

Объект управления предоставляет управляющей системе некоторый интерфейс. В этом примере — функции `newPerson()`, `setLastName()` и `setFirstName()`. Эти функции программист пишет вручную.

Автомат управления вызывает функции объекта управления в соответствии с выходными воздействиями, заданными в описании SAX-обработчика. При этом могут использоваться два вспомогательных метода, предоставляемых автоматом управления: `capture()` и `captured()`. Метод `capture()` инициирует захват поступающего текстового контента во временный буфер автомата. Метод `captured()` возвращает захваченный с предыдущего вызова `capture()` контент и очищает буфер.

## 4. АЛГОРИТМ ПОСТРОЕНИЯ АВТОМАТА УПРАВЛЕНИЯ

Генерация управляющей части SAX-обработчика требует выделения управляющих состояний и построения переходов между ними.

Выделение состояний является достаточно простой задачей, так как состояния уже неявно выделены при составлении текстового описания XML-обработчика. Подробности алгоритма можно изучить по исходным кодам его реализации [9].

## 5. АВТОМАТИЧЕСКАЯ ГЕНЕРАЦИЯ КОДА АВТОМАТА SAX-ОБРАБОТЧИКА

Имея описание управляющего автомата в виде множества состояний и переходов между ними, можно выполнить автоматическую генерацию кода автомата SAX-обработчика для произвольного языка программирования. В рамках данной работы была реализована кодогенерация автомата SAX-обработчика на языке Java.

Результатом кодогенерации является класс, который расширяет `DefaultHandler` и реализует указанные выше методы `startElement`, `endElement` и `characters`. При этом все пе-

реходы автомата по открывающим тегам размещаются в методе `startElement`, а по закрывающим — в методе `endElement`. Автомат имеет всего две внутренние переменные: текущее состояние и буфер для временного хранения текстового контента. Таким образом, расход памяти минимален.

## 6. ЗАКЛЮЧЕНИЕ

В данной работе предложен декларативный язык описания SAX-обработчиков и система автоматической генерации кода на его основе. Входными данными для системы является простое и наглядное текстовое описание обработчика. На основе этого описания строится управляющий автомат обработчика, который затем преобразуется в код на том или ином языке программирования. На настоящий момент реализовано преобразование автомата в код на языке Java. Остальной код обработчика, соответствующий объекту управления, пишется вручную. Предложенный подход отличается одновременно простотой написания кода и эффективностью использования ресурсов компьютера.

Описанный подход был использован при разработке приложения, с помощью которого извлечены данные из тысяч HTML-страниц, содержащих несколько типов таблиц сложной структуры.

## ЛИТЕРАТУРА

1. Official website for SAX, http://www.saxproject.org/
2. Wikipedia Database Download, http://en.wikipedia.org/wiki/Wikipedia_database.
3. W3C Document Object Model, http://www.w3.org/DOM/.
4. JAXB Reference Implementation, https://jaxb.dev.java.net/.
5. **Kiselyov O.** A better XML parser through functional programming. LCNS, Springer-Verlag, 2002, pp. 209-224.
6. **Chiu K., Lu W.** A Compiler-Based Approach to Schema-Specific XML Parsing. First International Workshop on High Performance XML Processing 2004.
7. **Altinel M., Franklin M.** Efficient Filtering of XML Documents for Selective Dissemination of Information. VLDB, 2000, pp. 53-64.
8. **Поликарпова Н. И., Шалыто А. А.** Автоматное программирование. – СПб.: Питер, 2009. 176 с.
9. SaxGen source code, http://code.google.com/p/saxgen/source

# DECLARATIVE LANGUAGE FOR XML DATA EXTRACTION ALGORITHMS

**Alexey Vladykin**
*Saint Petersburg State University of Information Technologies, Mechanics and Optics, Saint Petersburg, Russia*

**Abstract**

In this paper a declarative language for SAX handler definition is proposed. This language allows to describe complex XML parsing algorithms in a simple manner. An algorithm is introduced for automatic transformation of such handler descriptions into finite state machines, and then into source code. This approach reduces the complexity of SAX handler development by eliminating the greater part of error-prone manual work.