МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ РАДИОЭЛЕКТРОНИКИ

КАФЕДРА СИСТЕМОТЕХНИКИ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к лабораторным занятиям по дисциплине

«INTERNET-ТЕХНОЛОГИИ РАСПРЕДЕЛЕННОЙ ОБРАБОТКИ ИНФОРМАЦИИ»

для студентов специальности

6.050101 – КОМПЬЮТЕРНЫЕ НАУКИ

ЛАБОРАТОРНАЯ РАБОТА № 2
СВЯЗЫВАНИЕ XML-ДАННЫХ С ОБЪЕКТАМИ JAVA

Утверждено
на заседании кафедры «Системотехники»
Протокол № 1 от 28 сентября 2017 г.

Харьков 2017

УДК 681.3.07
ББК 32.973.26-018.2.75
I2

I12

Internet-технологии распределенной обработки информации. Методические указания к практическим занятиям для студентов всех форм обучения специальности 6.050101 – «Компьютерные науки» [Электронное издание] / ХНУРЭ; Сост. Ю.В.Мищеряков, А.И.Коваленко, В.М. Решетник – Харьков, 2017. – 43с.

УДК 681.3.07
ББК 32.973.26-018.2.75

СОДЕРЖАНИЕ

2 СВЯЗЫВАНИЕ XML-ДАННЫХ С ОБЪЕКТАМИ JAVA

2.1 Цели работы

Цели работы:

- изучить программные интерфейсы средств, использующиеся для маршализации и демаршализации данных XML и объектов Java;

- получить практические навыки по маршализации и демаршализации данных XML и объектов Java с помощью интерфейса API JAXB (Java Architecture for XML Binding);

- получить практические навыки по маршализации и демаршализации данных XML и объектов Java с помощью интерфейса DOM API (Document Object Model API);

- получить практические навыки по маршализации объектов Java в XML-документ с помощью интерфейса SAX (Simple API for XML);

- получить практические навыки по маршализации объектов Java в XML-документ с помощью интерфейса StAX (Streaming API for XML);

- получить практические навыки по использованию интегрированной среды разработки Eclipse.

2.2 Методические указания по организации самостоятельной работы студентов

Во время подготовки к выполнению лабораторной работы необходимо:

- ознакомится с интерфейсом и функциональными возможностями интегрированной среды разработки Eclipse;

- изучить набор интерфейсов и классов JAXP (Java API for XML Processing), в составе: API JAXB, DOM API, SAX, StAX;

- изучить порядок использования интерфейса API JAXB в IDE Eclipse для маршализации и демаршализации данных XML и объектов Java;

- изучить порядок использования интерфейса DOM API в IDE Eclipse для маршализации и демаршализации данных XML и объектов Java;

- изучить порядок использования интерфейса SAX в IDE Eclipse для маршализации объектов Java;

- изучить порядок использования интерфейса StAX в IDE Eclipse для маршализации объектов Java;

Для подготовки к занятию необходимо использовать конспект лекций [1], методические указания к самостоятельной работе [2] и рекомендованную

литературу [3,4] (Эллиот Расти Гарольд, СкоттМинс У. XML. Справочник, главы 14, 15; Хабибуллин И. Разработка WEB-СЛУЖБ средствами Java, глава 1 – Обработка документов XML, стр. 14-76).

2.3 Описание лабораторной установки

В качестве лабораторной установки используется персональный компьютер типа IBM PC. Выполнение заданий лабораторной работы осуществляется с помощью интегрированной среды разработки Eclipse, поддерживающей технологию разработки Java Enterprise Edition (JEE) с соответствующим пакетом Java Development Kit (JDK).

2.4 Порядок выполнения работы и методические указания по ее выполнению

2.4.1 Задание на практическую работу

В процессе выполнения работы (с использованием созданных на практическом занятии № 1 XML-документа и его XSD-схемы), необходимо выполнить следующие задания:

- с помощью интерфейса JAXB на основе XSD-схемы провести демаршалинг и сгенерировать Java классы.
- разработать парсер, реализующий демаршалинг с валидацией данных XML-документа в объекты классов Java с помощью интерфейсов:
  - API JAXB (Java Architecture for XML Binding);
  - DOM API (Document Object Model API);
  - SAX (Simple API for XML);
  - StAX (Streaming API for XML);
- разработать парсер, реализующий маршалинг с валидацией данных из объектов Java в XML-документ с помощью интерфейсов:
  - API JAXB (Java Architecture for XML Binding);
  - DOM API (Document Object Model API);
  - SAX (Simple API for XML);
  - StAX (Streaming API for XML);
- разработать консольное приложение Java для преобразования XML-файла в HTML-файл;
- оформить отчет о проделанной работе.

2.4.2 Теоретические сведения

2.4.2.1 Обзор интерфейсов, использующихся для связывания данных XML с объектами Java

Язык XML определяет способ описания древовидной (иерархической) структуры данных и не содержит инструкций по их отображению или обработке. Для использования этих данных необходимы специальные программы-анализаторы, проводящие разбор XML-документов. Программы-анализаторы разделяют по их предназначению на программы-сканеры (scanners) и программы-парсеры (parsers).

Программы-сканеры предназначены для проведения лексического анализа (lexical parsing). Для этого анализируемый XML-документ разбивается на отдельные неделимые элементы (tokens), которыми являются теги, служебные слова, разделители, текстовые константы. При анализе проводится проверка правильности определения элементов и их связей.

Программы-парсеры предназначены для проведения грамматического анализа (grammar parsing). При этом анализируется логическая структура XML-документа, составляются выражения, выражения объединяются в блоки, блоки – в модули, которыми могут являться абзацы, параграфы, пункты, главы.

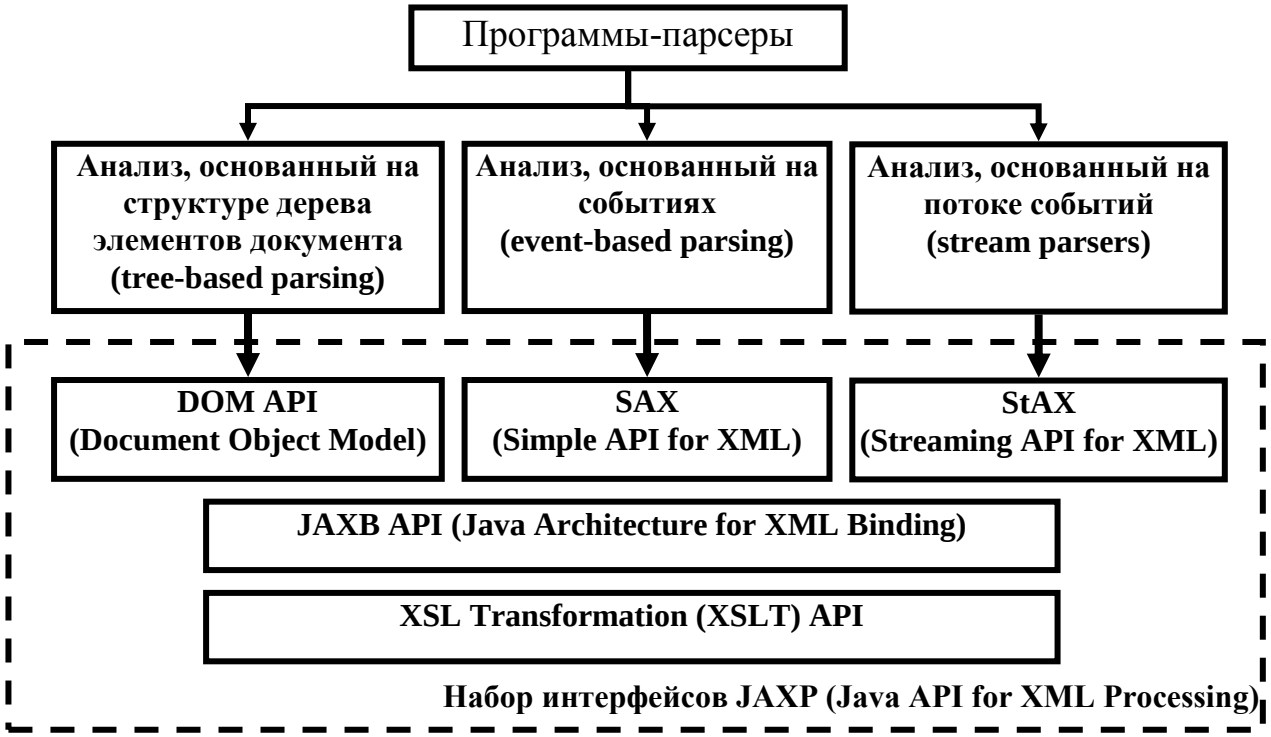

Рисунок 2.1 – Набор интерфейсов JAXP (Java API for XML Processing)

Программы-парсеры делятся на три группы (рис. 2.1):

– парсеры, проводящие анализ, основываясь на структуре дерева, отражающего вложенность элементов документа (tree-based parsing). Дерево документа перед разбором строится в оперативной памяти, что является недостатком данного метода. Необходимость частого просмотра узлов дерева замедляет работу парсера, что также является его недостатком;

– парсеры, проводящие анализ, основываясь на событиях (event-based parsing). Синтаксический анализатор последовательно просматривает XML-документ, фиксируя события просмотра. Событием считается появление очередного элемента XML: открывающего или закрывающего тега, текста, содержащегося в теле элемента. При возникновении события вызывается соответствующий метод его обработки;

– потоковые парсеры (stream parsers), обрабатывают XML как набор событий. Потоковый парсер запрашивает такие события одно за другим, в отличие от парсера «event-based parsing», который получает события от синтаксического анализатора в определяемом самим анализатором порядке.

В стандартную поставку Java Enterprise Edition входит набор интерфейсов и классов для создания парсеров и преобразования документов XML, называемый JAXP (Java API for XML Processing). Последняя редакция стандарта JAXP была опубликована в марте 2004 года с номером версии 1.4.

Набор JAXP содержит следующие интерфейсы API (Application Programming Interface – интерфейс прикладного программирования):

– DOM API (Document Object Model – объектная модель документа) – интерфейс, позволяющий создавать парсеры, основанных на структуре дерева элементов XML-документа;

– SAX (Simple API for XML – простой интерфейс для XML) – интерфейс, позволяющий создавать парсеры, основанных на событиях при анализе элементов XML-документа;

– StAX (Streaming API for XML – потоковый интерфейс для XML) – интерфейс, позволяющий создавать потоковые парсеры XML-документов;

– API JAXB API (Java Architecture for XML Binding – архитектура Java для XML) – интерфейс, позволяющий автоматизировать процесс разбора XML-документов;

XSLT API (XSLT-процессор) – прикладной интерфейс программирования связанный с расширяемым языком преобразования XML-документов (XSLT, eXtensible Stylesheet Language Transformations). XSLT-процессор позволяет преобразовать XML-документы в другие XML-документы или другие форматы. Чтобы выполнить преобразование, как правило, используется таблица стилей XSL (eXtensible Stylesheet Language – расширяемый язык стилей). Таблицы стилей XSL определяют способ отображения XML данных. Для доступа к отдельным частям преобразуемого XML-документа и для поддержки вычислений XSLT- процессор использует язык запросов к элементам XML-документа XPath (XML Path Language).

2.4.2.2 Интерфейс JAXB (Java Architecture for XML Binding)

API JAXB (Java Architecture for XML Binding) – это набор интерфейсов и классов для маршализации (маршалинга, marshal) и демаршализации (демаршалинга, unmarshal) объектов Java и XML-документов.

Маршаллизация – это процесс преобразования находящихся в памяти данных в формат их хранения. Для Java и XML, маршаллизация представляет собой преобразование некоторого набора Java-объектов в XML-документ (или документы). Для баз данных – размещение Java-данных в базе данных.

Демаршаллизация – это процесс преобразования данных из формата среды хранения в память, то есть прямо противоположный маршаллизации. Другими словами, демаршиллизация – это преобразование XML-документа в объект (объекты) Java.

Набор интерфейсов JAXB позволяет связать классы и объекты Java с их XML-представлением непосредственно, без преобразований через промежуточные интерфейсы, что является их главным преимуществом. Архитектура интерфейсов JAXB представлена на рис. 2.2.

JAXB предоставляет следующие интерфейсы:

– «schema compiler» – интерфейс командной строки xjc (xjc – XSD Java Compiler) генератора кода классов Java на основе XSD-схемы XML-документа;

– «schema generator» – интерфейс командной строки schemagen – генератора XSD-схемы XML-документа на основе кода класса (классов) Java;

– «unmarshal» – интерфейс валидации и демаршализации данных XML-документа в существующие объекты классов Java;

– «marshal» – интерфейс валидации и маршализации данных из объектов Java в XML-документ.

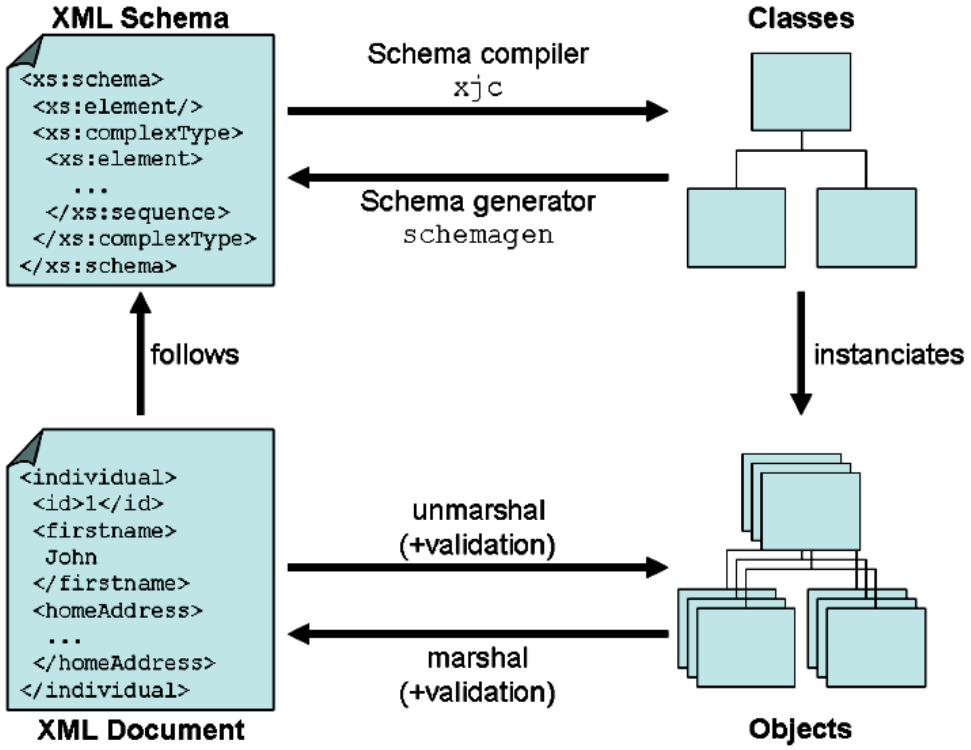

Рисунок 2.2 – Набор интерфейсов JAXB (Java Architecture for XML Binding)

2.4.2.2.1 Интерфейсы компиляторов «xjc» и «schemagen»

Компиляторы «xjc» и «schemagen» находятся в папке «bin» каталога JDK. Синтаксис использования интерфейса компилятора «xjc» в командной строке имеет вид:

```
xjc [-options ...] <schema file/URL/dir/jar> ... [-b <file>] ...,
```

где:

– «schema file/URL/dir» – определяет путь и имя файла с расширением «*.XSD» (файлу XML Schema);

– «-b <file>» – указывает один или несколько файлов «binding declaration». Инструкции в данных файлах позволяют переопределить правила связи XML-имен с Java-именами;

– «-options» – опции компилятора xjc:

- «-d <dir>» – определяет каталог, в который будут помещены сгенерированные Java-классы;
- «-p <pkg>» – переопределяет имя пакета для Java-классов;
- «-no-header» – подавляет генерацию комментариев в заголовках создаваемых файлов;
- «-xmlschema» – обрабатывает XSD-схему как W3C XML Schema (по умолчанию);
- «-quiet» – подавляет вывод компилятором информации о процессе компиляции и предупреждений;
- «-verbose» – включает вывод дополнительной информации в процессе компиляции;
- «-help» – вывод справки по интерфейсу командной строки;
- «-version» – вывод версии компилятора.

Пример использования синтаксиса интерфейса компилятора «xjc» в командной строке:

```
C:\Eclipse\workspace\bookShop>xjc -d src xsd\bookShop.xsd
```

где:

– «C:\Eclipse\workspace\bookShop» – путь к папке проекта Java с именем «bookShop», из которой запущен командный интерпретатор «cmd.exe»;

– «-d src» – определяет каталог «src», в который будут помещены сгенерированные Java-классы;

– «xsd\bookShop.xsd» – определяет каталог «xsd», в котором находится XSD-файл «bookShop.xsd».

Синтаксис использования компилятора «schemagen» в командной строке имеет вид:

```
schemagen [-options ...] <java files>
```

где: «<java files>» – путь и имена файлов с классами Java, а «-options» – опции компилятора «schemagen»:

– «-d <path>» – определяет исходящий каталог для компилятора;

– «-cp <path>» или «-classpath <path>» – указывает расположение связанных Java-классов;

– «-episode <file>» – компилятор генерирует episode-файл, обеспечивающий раздельную компиляцию

– «-help» – вывод справки по интерфейсу командной строки;

– «-version» – вывод версии компилятора.

В IDE Eclipse реализован доступ к интерфейсу компилятора «xjc» из контекстного меню «Generate» / «JAXB Classes…», связанного с XSD-схемами проекта Java (с файлами, имеющими расширение «*.XSD»).

В табл. 2.1 приведены соответствия типов данных XSD-схемы и типов данных Java при использовании компиляторов «xjc» и «schemagen» JAXB.

Таблица 2.1 –Соответствия типов данных XSD-схемы и объектов Java

Типы XSD-схемы	Типы объектов Java
xsd: string	**java.lang.String**
xsd: integer	**java.math.BigInteger**
xsd: positiveInteger	**java.math.BigInteger**
xsd: int	**Int**
xsd: long	**Long**
xsd: short	**Short**
xsd: decimal	**java.math.BigDecimal**
xsd: float	**Float**
xsd: double	**Double**
xsd: boolean	**Boolean**
xsd: byte	**byte**
xsd:QName	**javax.xml.namespace.QName**
xsd: dateTime	**javax.xml.datatype.XMLGregorianCalendar**
xsd: base64Binary	**byte[]**
xsd: hexBinary	**byte[]**
xsd: unsignedInt	**long**
xsd: unsignedShort	**int**
xsd: unsignedByte	**short**
xsd: unsignedLong	**java.math.BigDecimal**
xsd: time	**javax.xml.datatype.XMLGregorianCalendar**
xsd: date	**javax.xml.datatype.XMLGregorianCalendar**
xsd: g	**javax.xml.datatype.XMLGregorianCalendar**
для элементов xsd: anySimpleType	**java.lang.Object**
для атрибутов xsd: anySimpleType	**java.lang.String**
xsd: duration	**javax.xml.datatype.Duration**
xsd:NOTATION	**javax.xml.namespace.QName**

2.4.2.2.2 Маршализация и демаршализация с помощью интерфейса JAXB API

JAXB API реализуется пакетом javax.xml.bind. Перечень основных интерфейсов и классов, входящих в пакет javax.xml.bind и используемых для создания XML-документов, а также генерации классов Java, приведены в табл. 2.2.

Таблица 2.2 – Пакеты, входящие в javax.xml.bind

Пакет	Описание
javax.xml.bind	**Обеспечивает платформу для демаршализации, маршализации и проверки XML-документов в Java-приложениях**
javax.xml.bind.annotation	**Содержит аннотации для настройки преобразований между программой Java и XML-данными**
javax.xml.bind.annotation.adapters	**Классы-адаптеры JAXB**
javax.xml.bind.attachment	**Обеспечивает маршализацию/демаршализацию XML-документов с вложениями бинарных данных;**
javax.xml.bind.helpers	**Содержит частичные стандартные реализации некоторых интерфейсов javax.xml.binding**
javax.xml.bind.util	**Позволяет комбинировать JAXB с другими Java/XML-технологиями.**

2.4.3 Порядок выполнения работы

Задания лабораторной работы необходимо выполнять поэтапно. Примерная последовательность этапов работы:

1. Рассмотреть примеры парсеров XML-документа, созданных в проекте LB2ParserDemo (URL: https://github.com/engsyst/ws).

2. С использованием IDE Eclipse создать проект, включающий XML-документ и его XSD-схему.

3. С использованием интерфейса компилятора «xjc» из командной строки или с использованием контекстного меню IDE Eclipse на основе XSD-схемы XML-документа сгенерировать Java-классы.

4. Разработать DOM-парсер, реализующий демаршалинг с валидацией данных XML-документа в объекты классов Java. Отобразить передаваемые данные XML-документа с помощью консоли.

5. Разработать DOM-парсер, реализующий маршалинг с валидацией данных из объектов Java в XML-документ.

6. Разработать SAX-парсер, реализующий демаршалинг с валидацией данных XML-документа в объекты классов Java. Отобразить передаваемые данные XML-документа с помощью консоли.

7. Разработать SAX-парсер, реализующий маршалинг с валидацией данных из объектов Java в XML-документ.

8. Разработать StAX-парсер, реализующий демаршалинг с валидацией данных XML-документа в объекты классов Java. Отобразить передаваемые данные XML-документа с помощью консоли.

9. Разработать StAX-парсер, реализующий маршалинг с валидацией данных из объектов Java в XML-документ.

10. Разработать JAXB-парсер, реализующий демаршалинг с валидацией данных XML-документа в объекты классов Java. Отобразить передаваемые данные XML-документа с помощью консоли.

11. Разработать JAXB-парсер, реализующий маршалинг с валидацией данных из объектов Java в XML-документ.

12. Разработать консольное приложение Java для преобразования XML-файла в HTML-файл;

13. Оформить отчет о проделанной работе.

Критерии оценки для оформленного отчета (без теории)

- «**удовлетворительно**» – маршалинг и демаршалинг с валидацией данных и использованием интерфейсов
  - DOM API (Document Object Model API);
  - SAX (Simple API for XML);
  - StAX (Streaming API for XML);

- «**хорошо**» –маршалинг и демаршалинг с валидацией данных и использованием интерфейсов:
  - DOM API (Document Object Model API);
  - SAX (Simple API for XML)
  - StAX (Streaming API for XML);
  - API JAXB (Java Architecture for XML Binding);

- «**отлично**» – маршалинг и демаршалинг с валидацией данных и использованием интерфейсов:
  - DOM API (Document Object Model API);
  - SAX (Simple API for XML)
  - StAX (Streaming API for XML);
  - API JAXB (Java Architecture for XML Binding);
  - разработать консольное приложение Java для преобразования XML-файла в HTML-файл;

2.5 Содержание отчета

Отчет должен содержать:

- цель работы;
- постановку задачи;
- описание предметной области;
- код XML-файла, с внесенными данными;
- код файла с XSD-схемой XML-документа;
- код классов Java, сгенерированный с помощью интерфейса JAXB на основе XSD-схемы;
- код DOM-парсера (с реализованной валидацией) для демаршалинга данных из XML-документа в объекты Java Скриншот процесса демаршалинга;
- код DOM-парсера (с реализованной валидацией) для маршалинга данных из объектов Java в XML-документ. Код полученного XML-документа;
- код SAX-парсера (с реализованной валидацией) для демаршалинга данных из XML-документа в объекты Java Скриншот процесса демаршалинга;
- код SAX-парсера (с реализованной валидацией) для маршалинга данных из объектов Java в XML-документ. Код полученного XML-документа;
- код StAX-парсера (с реализованной валидацией) для демаршалинга данных из XML-документа в объекты Java Скриншот процесса демаршалинга;
- код StAX-парсера (с реализованной валидацией) для маршалинга данных из объектов Java в XML-документ. Код полученного XML-документа;
- код JAXB-парсера (с реализованной валидацией) для демаршалинга данных из XML-документа в объекты Java Скриншот процесса демаршалинга;
- код JAXB-парсера (с реализованной валидацией) для маршалинга данных из объектов Java в XML-документ. Код полученного XML-документа;
- выводы по работе.

2.6 Контрольные вопросы и задания

1. В чем отличие DOM-парсера от SAX?
2. В чем отличие SAX-парсера от StAX?
3. Какие методы DOM, SAX и StAX парсеров Вы применяли для чтения данных из XML?
4. Какие методы DOM, SAX и StAX парсеров Вы применяли при записи данных в XML?

5 Назовите основные методы интерфейса «org.xml.sax.ContentHandler».

Учебное издание / Электронное учебное издание

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
для практических занятий по дисциплине
«INTERNET-ТЕХНОЛОГИИ РАСПРЕДЕЛЕННОЙ ОБРАБОТКИ ИНФОРМАЦИИ»
для студентов специальности
6.050101 – КОМПЬЮТЕРНЫЕ НАУКИ

Составители: МИЩЕРЯКОВ Юрий Валентинович
КОВАЛЕНКО Андрей Иванович
РЕШЕТНИК Виктор Михайлович

Ответственный выпускающий: Гребенник И.В.
Редактор: Мищеряков Ю.В.
Компьютерная верстка: Коваленко А.И.

План 2017 (первое полугодие), поз. 9
Подп. к печ. 30.01.2017. Формат 60x84 1/16. Способ печати – ризография
Усл. печ. лист. 9,5. Учет. изд.лист. 8,4 Тираж 50 экз.
Цена договорная Зам. №1-9

ХНУРЭ. Украина. 61166, Харьков, просп. Науки, 14

Отпечатано в учебно-научном
издательско-полиграфическом центре ХНУРЭ
61166, Харьков, просп. Науки, 14