Курс подготовлен при поддержке Sun Microsystems
Правила использования материалов опубликованы на www.sun.ru

Java & XML

Красько Николай
nkrasko@mail.ru

Кратко об XML

XML – eXtensible Markup Language

Язык разметки документов - это набор специальных инструкций, называемых тэгами, предназначенных для формирования в документах какой-либо структуры и определениющий отношения между различными элементами этой структуры.

Упрощение SGML(Standart Generalised Markup Language)
Расширение HTML

Подробнее: http://www.w3.org

Пример XML

```
<books >
    <book ref="www.publisher1.com">
        <author>Author1</author>
        <name>Name1</name>
    </book>
    <book ref="www.publisher2.com">
        <author>Author2</author>
        <name>Name2</name>
        <cover>Hard</cover>
    </book>
</books>
```

Simple API for XML(SAX)

Событийно-ориентированное API

Каждый раз, когда при разборе XML документа анализатор оказывается в каком-то новом состоянии - обнаруживает какую-либо синтаксическую конструкцию XML-документа (элемент, символ, шаблон, и т.д.), фиксирует начало, конец объявлений элементов документа, просматривает DTD-правила или находит ошибку, он воспринимает его как произошедшее событие и вызывает внешнюю процедуру - обработчик этого события.

В Java 1.5 - **org.xml.sax**

Подробнее: http://www.saxproject.org/

Пример SAX – Часть 1

```
public class SAXExample extends DefaultHandler{
    //Начало документа
    public void startDocument() throws SAXException{…}
    //Окончание документа
    public void endDocument() throws SAXException{…}
    //Начало элемента
    public void startElement(String namespaceURI,
                             String localName,
                             String qName,
                             Attributes attr) throws SAXException{…}
    //Окончание элемента
    public void endElement(String namespaceURI,
                           String localName,
                           String qName) throws SAXException{…}
```

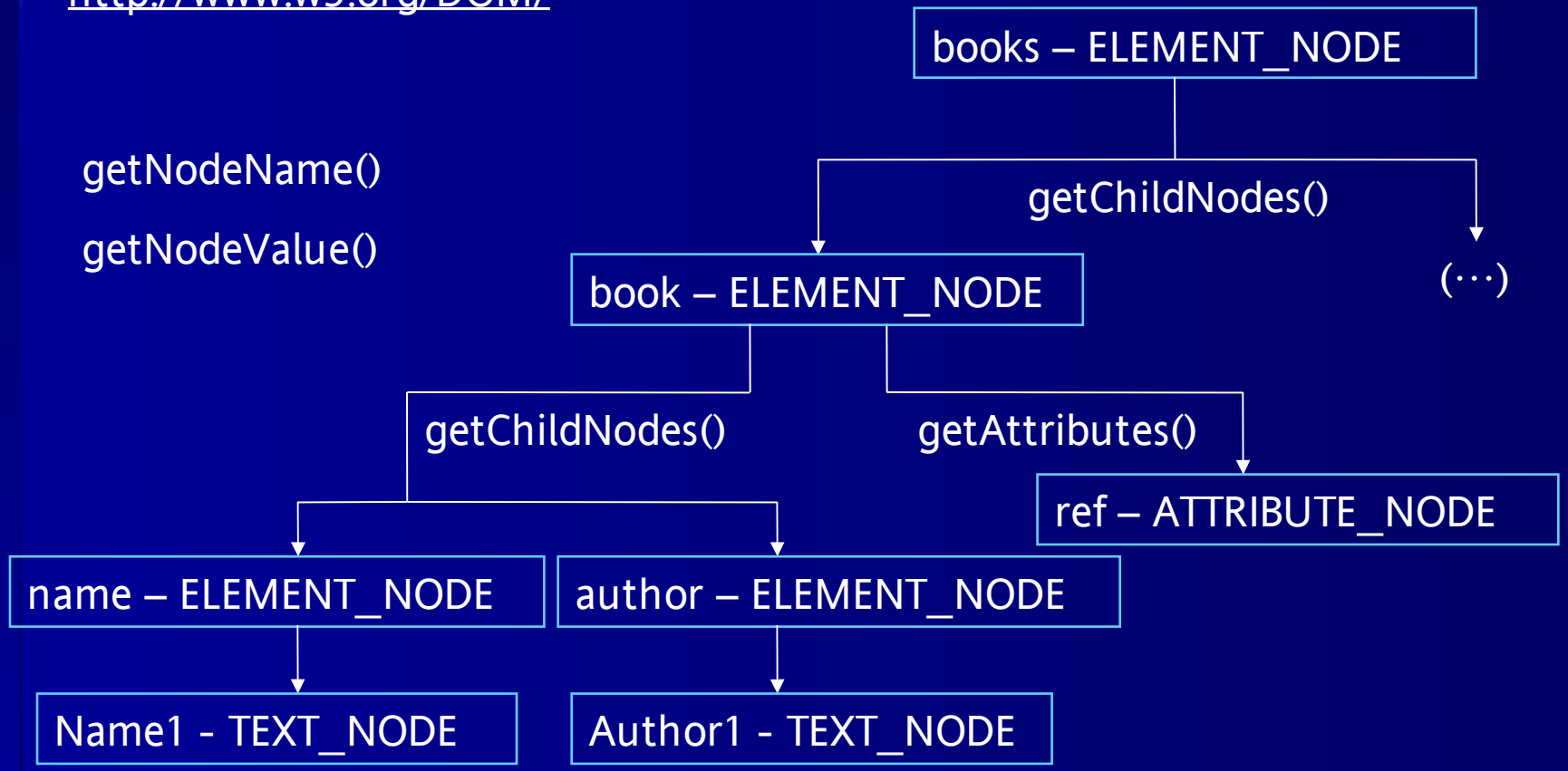
Пример SAX – Часть 2

```
    public void characters(char[] ch,
                           int start,
                           int length) throws SAXException {…}
    public static void main(String[] argv){
        System.out.println("Example SAX Events:");
        try{
            XMLReader xr = XMLReaderFactory.createXMLReader();
            xr.setContentHandler(new SAXExample());
            xr.parse(new InputSource(new FileReader("books.xml")));
        }
        catch (Exception e){
            e.printStackTrace();
        }
    }
}
```

Document Object Model (DOM)

При парсинге XML – файла происходит заполнение дерева.

http://www.w3.org/DOM/



Пример DOM

```
import org.w3c.dom.Document; //Класс представления дерева
import org.w3c.dom.Node; //Узел
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import java.io.FileReader;

public class DOMExample{
  public static void main(String[] arg) {
     DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
     DocumentBuilder db;
     try{
        db = dbf.newDocumentBuilder();
        Document xml_document= db.parse(new InputSource (
                                                  new FileReader("books.xml")));
     } catch (Exception e) {… }}}
```

Проверка XML(Validation)

Является частью Java API for XML Processing (JAXP) 1.3,
Поддержка включена в Java Platform Standard Edition (J2SE) 5.0

В JAXP 1.3 предоставляется новое API для валидации XML. Новый подход рассматривает проверку XML файлов как отдельную задачу. Раньше валидация являлась частью XML – парсеров (XMLParser, DocumentBuilder)

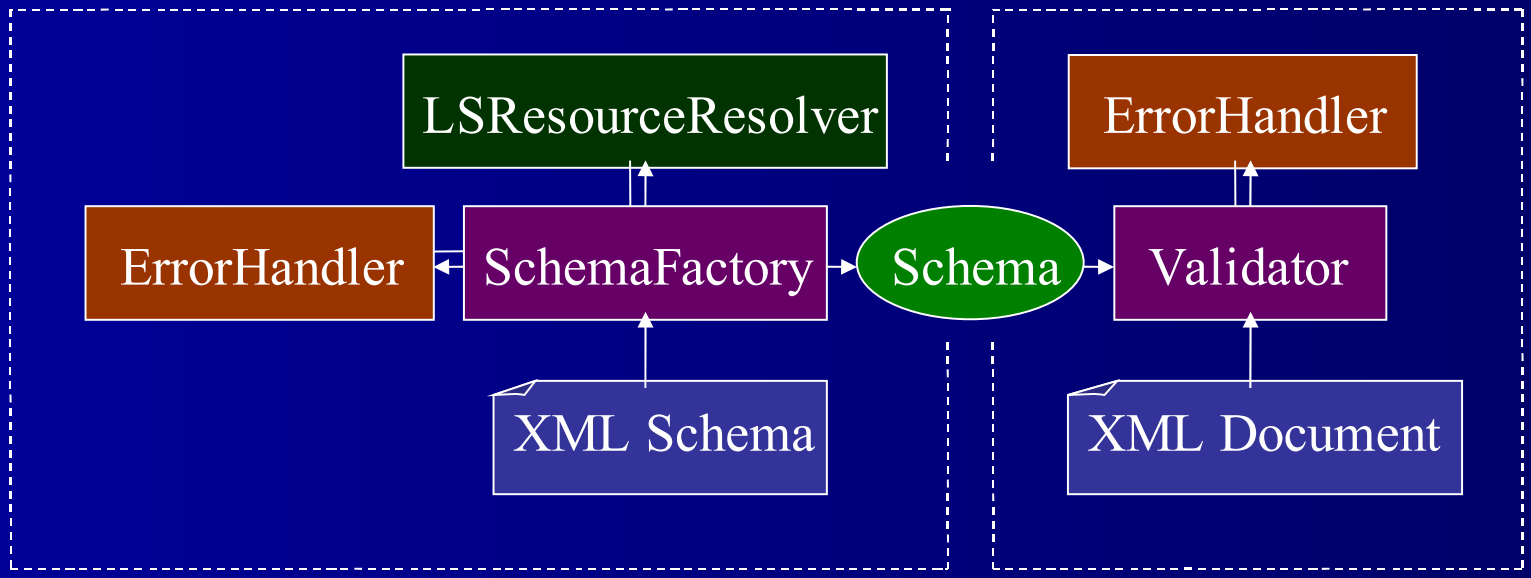
Пакет: javax.xml.validation

Структура классов

Первоначально компилируется схема. Затем при помощи нее может быть произведена проверка XML – файла на соответствие этой схеме.

Компиляция схемы:

Проверка документа:



Пример Validation

```
import org.xml.sax.SAXException;
import javax.xml.validation.Schema;
import javax.xml.validation.SchemaFactory;
import javax.xml.validation.Validator;
import javax.xml.XMLConstants;


SchemaFactory sf =SchemaFactory.newInstance(
                        XMLConstants.W3C_XML_SCHEMA_NS_URI);
sf.setErrorHandler(new MyErrorHandler());
Schema schema = sf.newSchema(new File("book_schema.xsd"));
Validator validator = schema.newValidator();
validator.setErrorHandler(new MyErrorHandler());
validator.validate(new StreamSource("books.xml"));
```

Пример Validation

```
public class MyErrorHandler implements org.xml.sax.ErrorHandler{
    public MyErrorHandler() { }
    public void error(org.xml.sax.SAXParseException
     sAXParseException) throws org.xml.sax.SAXException {
        System.out.println("ERROR: " + sAXParseException.toString());
    }
    public void fatalError(org.xml.sax.SAXParseException
     sAXParseException) throws org.xml.sax.SAXException {
        System.out.println("FATAL ERROR: " +
     sAXParseException.toString());
    }
    public void warning(org.xml.sax.SAXParseException
     sAXParseException) throws org.xml.sax.SAXException {
        System.out.println("WARNING: " + sAXParseException.toString());
    }
}
```

XPath

Доступ к информации в XML – документе может быть осуществлен при помощи простого языка XPath. Одно выражение на XPath может заменить собой множество строк кода, написанного с использованием SAX или DOM API. JAXP 1.3 поддерживает XPath версии 1.0.

Примеры конструкций:

```
/books/book/name/text()
/book/@ref
/books/book[2]/author/text()
```

http://www.w3.org/TR/xpath - спецификация XPath
http://www.rol.ru/news/it/helpdesk/xpath01.htm - русский вариант

Пример XPath

```
import javax.xml.xpath.XPath;
import javax.xml.xpath.XPathFactory;
import javax.xml.xpath.XPathConstants;
import javax.xml.xpath.XPathExpressionException;
import org.xml.sax.InputSource;
import org.w3c.dom.NodeList;

XPath xpath = XPathFactory.newInstance().newXPath();
String expression = "/books/book[2]/author/text()";
NodeList nameNodes = (NodeList) xpath.evaluate(expression, new
                InputSource("books.xml"), XPathConstants.NODESET);
for(int i = 0 ; i < nameNodes .getLength(); i++){
    System.out.println("Book author " + (i+1) + " is " +
                                nameNodes.item(i).getNodeValue());
}
```

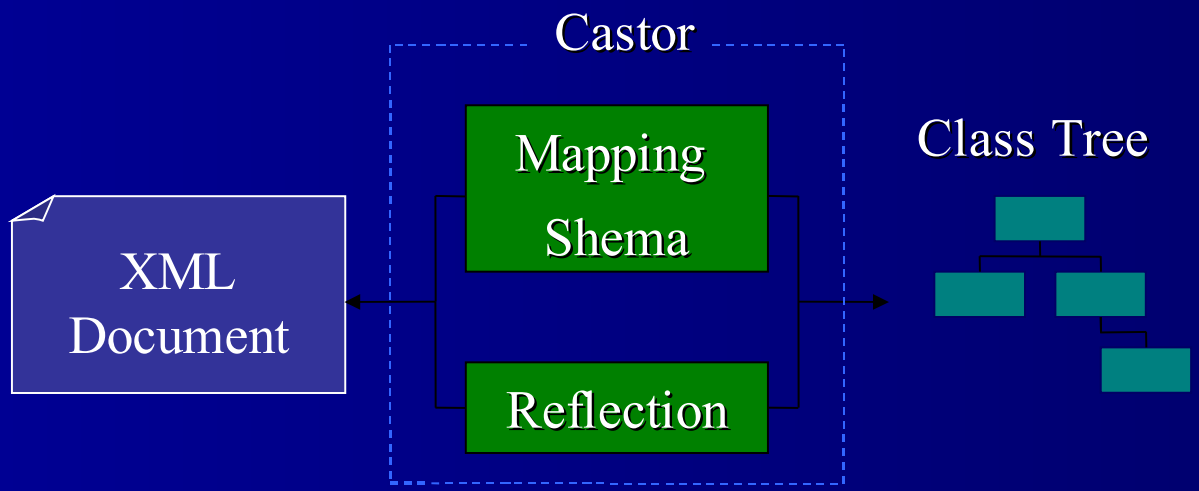
Библиотека Castor.

Castor XML – библиотека работы с XML файлами. В отличии от основных APIs (DOM и SAX), которые работают со структурой документа, библиотека Castor направлена на работу с данными. При этом происходит сопоставление XML – файла с объектной моделью, отображающей структуру документа.

Unmarshal – процесс конвертирования данных из некоторого потока (последовательность байт) в систему объектов.
Marshal – процесс сохранения данных, представленных в системе объектов в поток.

Marshal – сохранение; Unmarshal – загрузка

Marshalling/Unmarhaling



Конвертирование может быть выполнено практически для всех “bean-like” систем классов.

Основное ограничение на класс:

- Должен быть public конструктор без параметров
- Для полей, которые планируется доступными для операций Marshal\Unmarshal, необходимо обеспечить наличие подходящих методов “getter” и “setter”

Class/Field Descriptors

Конвертирование информации, содержащейся в XML – документе в систему объектов происходит при помощи интерфейсов ClassDescriptor и FieldDescriptor.

ClassDescriptor – интерфейс описания интерпретации конвертирования класса.

- getExtends – родительский класс
- getFields – дескрипторы полей класса
- getJavaClass – тип класса

FieldDescriptor – интерфейс описания интерпретации конвертирования поля класса.

Сохраняет информацию о том, к какому дескриптору класса принадлежит(getContainingClassDescriptor), каким дескриптором описывается данное поле(getClassDesciptor)