XML

© Составление, Будаев Д.С., Гаврилов А.В., 2013

Лекция 13

УНЦ «Инфоком»
Самара
2013

NetCracker®
Transforming the Service Layer™

План лекции

- Общие принципы
- Document type definition
- SAX и DOM
- Работа с SAX и DOM в Java
- Запись XML в Java
- XML-сериализация в Java

У истоков

- Standard Generalized Markup Language (SGML)
  - Предназначался для описания структуры сложных документов
  - Был разработан в 1970 году
  - Основные цели:
    - Все документы должны быть выполнены в строгом соответствии с правилами
    - Уменьшение количества документации

Наследники

- Hypertext Markup Language (HTML)
  Язык разметки гипертекста (описание представления Web-страницы)

- Extensible Markup Language (XML)
  Язык для описания иерархических данных (портируемое хранение данных)
  http://www.w3.org/http://www.w3.org/XML/

Отличия XML от HTML

- XML чувствителен к регистру
- В XML нужно закрывать тэги
- В XML часто встречаются тэги, одновременно открывающие и закрывающие
  `<img src="coffeecup.png" />`
- В XML значения атрибутов должны быть заключены в кавычки
- В XML все атрибуты должны иметь значения

Пример XML

```
<configuration>
  <title>
    <font> <name>Helvetica</name> <size>36</size> </font>
  </title>
  <body> <name>Times Roman</name> <size>12</size> </body>
  <window> <width>400</width> <height>200</height> </window>
  <color>
    <red>0</red>
    <green>50</green>
    <blue>100</blue>
  </color>
  <menu>
    <item>Times Roman</item>
    <item>Helvetica</item>
  </menu>
</configuration>
```

Структура XML-документа

- Заголовок

```
<?xml version="1.0"?>
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="ex01-1.xsl"?>
```

- Объявления типа документа

```
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.
//DTD Web Application 2.2//EN"
"http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
```

- Корневой элемент

```
<configuration>
</configuration>
```

Структура XML-документа

- Смешанное наполнение не рекомендуется

```
<font>
    Helvetica
    <size>36</size>
</font>
```

- Существуют атрибуты

```
<font>
    <name>Helvetica</name>
    <size unit="pt">36</size>
</font>
```

Некоторые инструкции

- Символы

```
&#233, &#x2122
```

- Стандартные символы

```
&lt; &gt; &amp; &quot; &apos;
```

- Инструкции обработки

```
<?xml version="1.0"?>
```

- Комментарии

```
<!-- This is a comment. -->
```

Правильный документ

- Начинается с объявления
- Содержит один уникальный корневой элемент
- Все открытые теги закрываются
- Учтена чувствительность к регистру
- Теги корректно вложены друг в друга
- Значения всех атрибутов заключены в кавычки
- Специальные символы задаются с помощью инструкций

Document Type Definition (DTD)

- Содержит правила, описывающие структуру документа
- Транслятор может автоматически проверять документ на соответствие этим правилам
- Описывает дочерние элементы и атрибуты для каждого элемента
- Включение в XML-документ

```
<!DOCTYPE имя [правила]>

<!DOCTYPE configuration SYSTEM "config.dtd">

<!DOCTYPE configuration SYSTEM
"http://myserver.com/config.dtd">
```

Регулярные выражения

Правило	Смысл
E*	0 или больше вхождений E
E+	1 или больше вхождений E
E?	0 или 1 вхождение E
E1|E2|…|En	Одно из E1, E2, …, En
E1, E2, …, En	Последовательность E1, E2, …, En
#PCDATA	Текст
(#PCDATA|E1|…|En)*	Смешанное наполнение
ANY	Произвольный дочерний тэг
EMPTY	Нет дочерних тэгов

Примеры выражений

```
Описание меню
<!ELEMENT menu (item)*>

Описание шрифта
<!ELEMENT font (name,size)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT size (#PCDATA)>

Описание главы в книге
<!ELEMENT chapter
(intro,(heading,(para|image|table|note)+)+)>
```

Описание атрибутов: типы

Тип	Смысл
CDATA	Произвольная строка
(A1|A2|…|An)	Один из строковых атрибутов A1, A2, …, An
NMTOKEN, NMTOKENS	Одна или более строк, записанных по правилам имен
ID	Уникальный ID
IDREF, IDREFS	Одна или более ссылка на уникальный ID
ENTITY, ENTITIES	Ссылки на внешние сущности

Описание атрибутов: значения

Значение	Смысл
#REQUIRED	Атрибут обязателен
#IMPLIED	Атрибут опционален
A	Атрибут опционален, если значение не указано, то принимается равным A
#FIXED A	Атрибут не указывается или равен A

Примеры выражений

```
<!ATTLIST font style (plain|bold|italic|bold-italic)
plain>
<!ATTLIST size unit CDATA #IMPLIED>

<!ELEMENT gridbag (row)*>
<!ELEMENT row (cell)*>
<!ATTLIST cell gridwidth CDATA "1">
<!ATTLIST cell gridheight CDATA "1">
<!ATTLIST cell fill (NONE|BOTH|HORIZONTAL|VERTICAL)
"NONE">
<!ATTLIST cell anchor (CENTER|NORTH|NORTHEAST|EAST
|SOUTHEAST|SOUTH|SOUTHWEST|WEST|NORTHWEST)
"CENTER">
<!ATTLIST cell ipadx CDATA "0">
<!ATTLIST cell ipady CDATA "0">
```

XML Schema

- Предназначена для того же, что и DTD
- Для описания правил используется непосредственно XML
- Имеет более гибкие возможности, чем DTD
  - Расширяема
  - Более гибкие возможности
  - Есть понятие типа данных
  - Есть понятие пространства имен
- Сложнее в восприятии и программировании средств, ее обрабатывающих
- www.w3.org/XML/Schema
  http://www.w3schools.com/Schema/default.asp

Поддержка типов данных

- Проще описывать допустимое содержимое документа
- Проще проверять корректность данных
- Проще накладывать ограничения на данные
- Проще определять формат данных

XML Schema описывается на XML

- Не требуется изучение еще одного языка
- Вы можете использовать свой любимый XML-редактор для работы со схемой
- Вы можете работать со схемой программно
- Вы можете изменять свою схему с помощью XSLT

Документ и тип DTD

```
<?xml version="1.0"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

```
<!ELEMENT note (to, from, heading, body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

XML Schema для документа

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">

<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

Указание типа документа

```
<?xml version="1.0"?>

<!DOCTYPE note SYSTEM "http://www.w3schools.com/dtd/note.dtd">

<note>
  <!-- Some content -->
</note>
```

```
<?xml version="1.0"?>

<note
xmlns="http://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3schools.com note.xsd">
 <!-- Some content -->
</note>
```

Extensible Stylesheet Language (XSL)

- Комплекс технологий, связанных с преобразованием и представлением XML-документов
- Обычно используется для преобразования документов в XML, HTML, текст и PDF (XSL-FO)
- XSL Transformations (XSLT) – язык, на котором описываются правила преобразования
- XPath – язык, позволяющий формулировать используемые в процессе преобразования выражения, использующие различные фрагменты документа
- http://www.w3.org/Style/XSL/
  http://www.w3schools.com/xsl/

XPath

- Вспомогательный язык, позволяющий обращаться к элементам документа
- Имя элемента представляется в виде пути `/bookstore/book/title`
- Обращение может происходить и к атрибутам
- http://www.w3.org/TR/xpath
  http://www.w3schools.com/Xpath/default.asp

Примеры выражений XPath

Выражение	Результат
`bookstore`	Все дочерние элементы для элемента bookstore
`/bookstore`	Корневой элемент bookstore
`bookstore/book`	Все элементы book, дочерние для bookstore
`//book`	Все элементы book в документе
`bookstore//book`	Все элементы book в рамках элемента bookstore
`@lang`	Атрибуты lang
`.`	Текущий элемент
`..`	Родительский элемент

Принципы XSL

- Контекстно-зависимый язык
- Основные элементы – выводимый текст и шаблоны
  - Текст просто выводится
  - Шаблоны описывают некоторые действия
    - Могут быть вызваны явно
    - Могут быть вызваны неявно, по условию совпадения шаблона
- Имеются средства управления ходом выполнения
- Позволяет создавать и вызывать бибилиотеки с помощью тега `<xsl:include href="..."/>`

Пример XML (catalog.xml)

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="tranformation.xsl" ?>
  <catalog>
    <cd>
      <title>Empire Burlesque</title>
      <artist>Bob Dylan</artist>
      <country>USA</country>
      <company>Columbia</company>
      <price>10.90</price>
      <year>1985</year>
    </cd>
    <cd>
      <title>Hide your heart</title>
      <artist>Bonnie Tyler</artist>
      <country>UK</country>
      <company>CBS Records</company>
      <price>9.90</price>
      <year>1988</year>
    </cd>
    …
  </catalog>
```

Пример XSL (tranformation.xsl)

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
        <tr>
          <td><xsl:value-of select="title"/></td>
          <td><xsl:value-of select="artist"/></td>
        </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Результат трансформации

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti
One night only	Bee Gees
Sylvias Mother	Dr.Hook
Maggie May	Rod Stewart
Romanza	Andrea Bocelli
When a man loves a woman	Percy Sledge
Black angel	Savage Rose
1999 Grammy Nominees	Many
For the good times	Kenny Rogers
Big Willie style	Will Smith

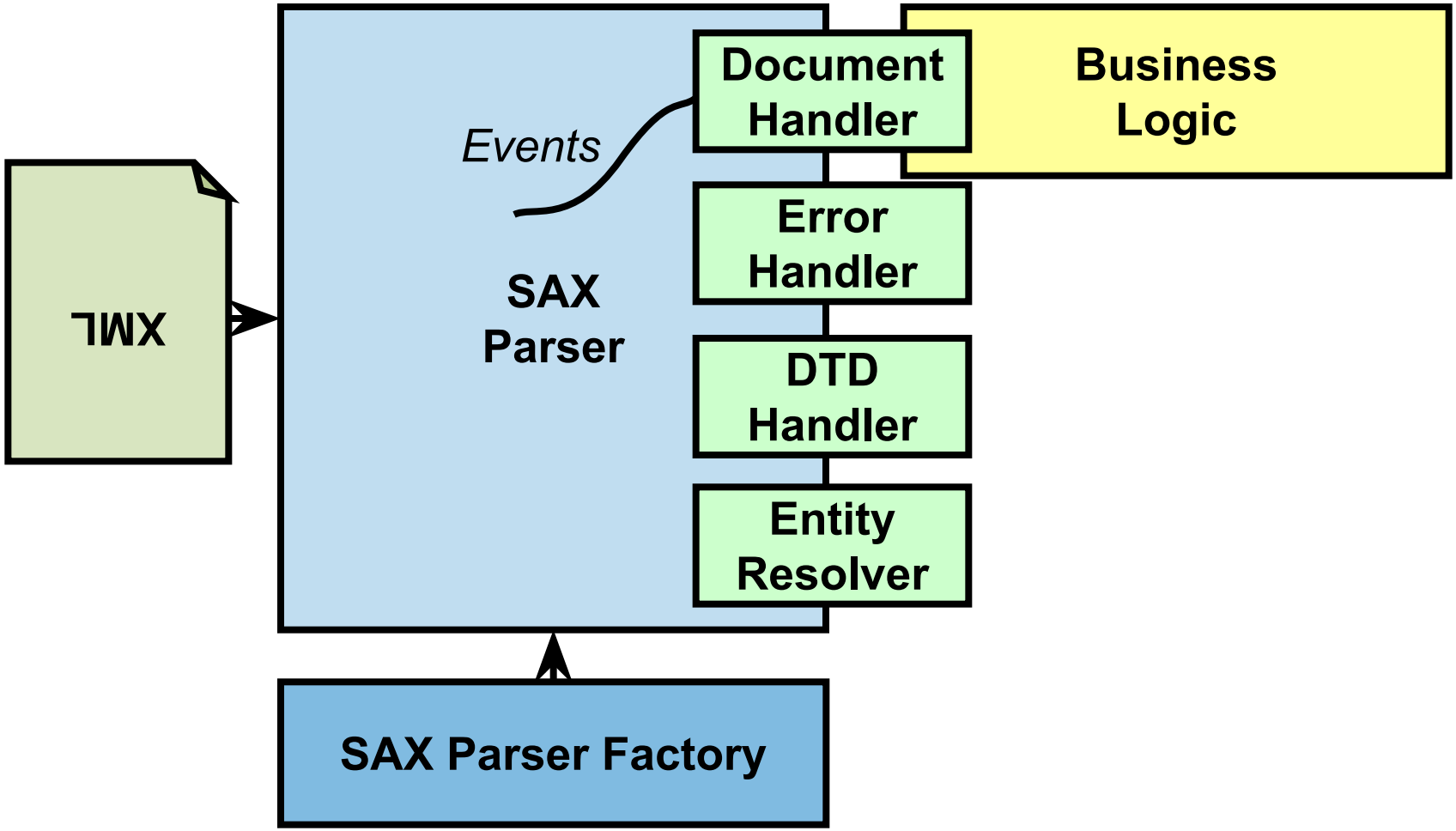
Обработка XML

Два подхода

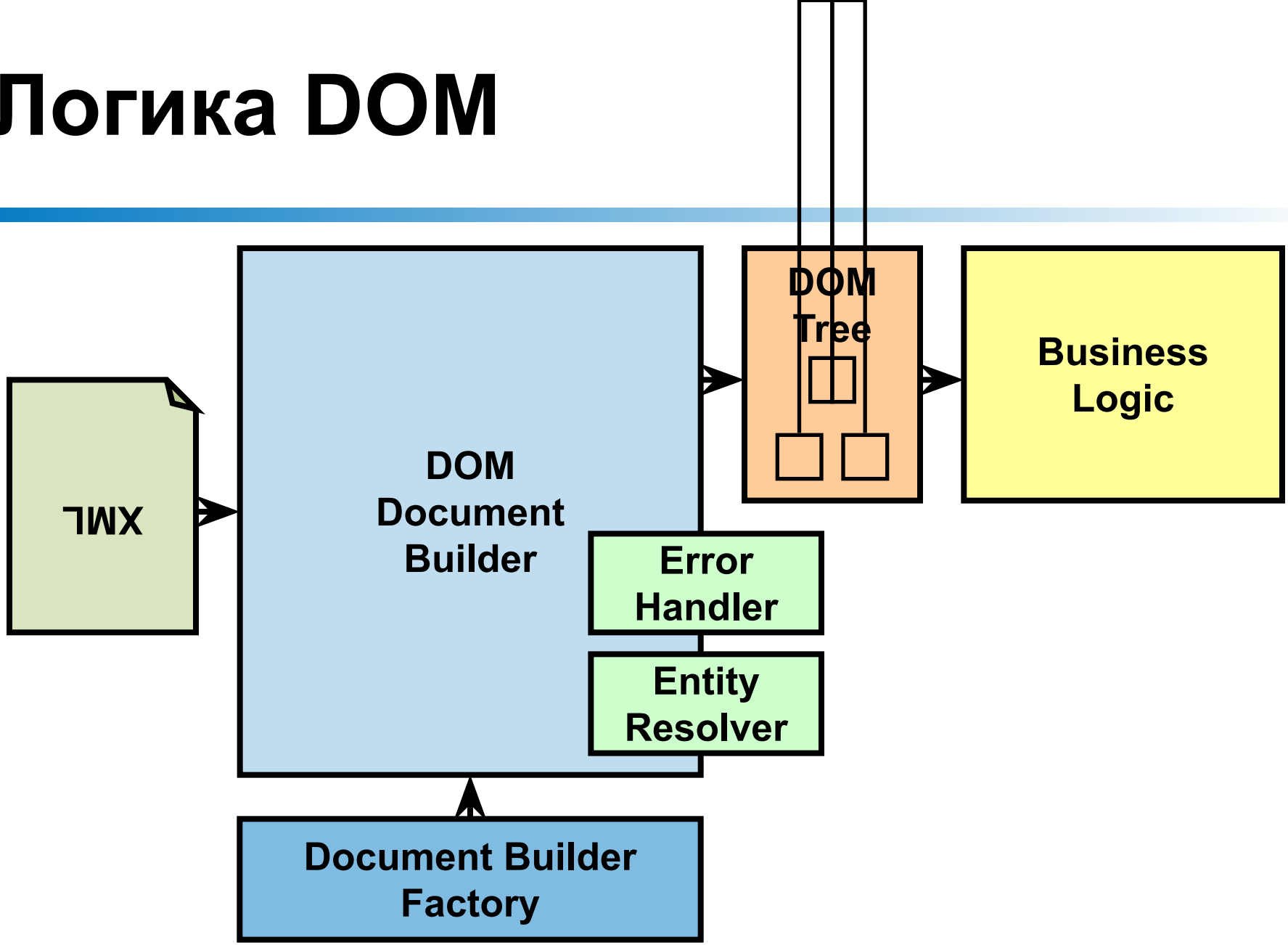
- **Simple API for XML (SAX)**
  Порождает события в процессе чтения XML документа

- **Document Object Model (DOM)**
  Представляет XML документ в форме древовидной структуры элементов

Логика SAX


Document Handler
Business Logic
Events
Error Handler
XML
SAX Parser
DTD Handler
Entity Resolver
SAX Parser Factory

Логика DOM


XML
DOM
Document
Builder
DOM
Tree
Business
Logic
Error
Handler
Entity
Resolver
Document Builder
Factory

Особенности SAX и DOM

SAX	DOM
Модель обработки событий	Древовидная структура данных
Последовательный доступ (поток событий)	Произвольный доступ (структура данных в памяти)
Используется мало памяти (порождаются только события)	Используется много памяти (документ загружен полностью)
Для обработки частей документа (обработка релевантных событий)	Для редактирования документа (обработка данных в памяти)
Для однократной обработки документа	Для многократной обработки документа

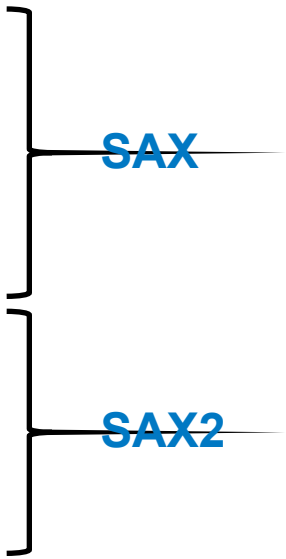
Работа с XML в Java

- Стандартные интерфейсы
  - В оригинале описаны на Interface Definition Language (OMG IDL)
  - Пакет `org.w3c.dom`
  - Пакет `org.xml.sax`
- Реализующие классы
  - Предоставляются отдельно…
  - JAVA API for XML Processing (JAXP)
    Пакет `javax.xml`

Работа с SAX

- Обработку документа производит транслятор, передающий информацию зарегистрировавшимся обработчикам событий

- Обработчики должны реализовывать интерфейсы
  - `org.xml.sax.ContentHandler`
  - `org.xml.sax.DTDHandler`
  - `org.xml.sax.EntityResolver`
  - `org.xml.sax.ErrorHandler`
  - `org.xml.sax.Locator`
  - `org.xml.sax.ext.DeclHandler`
  - `org.xml.sax.ext.EntityResolver2`
  - `org.xml.sax.ext.LexicalHandler`
  - `org.xml.sax.ext.Locator2`



Пакет javax.xml.parsers

- Класс **`SAXParserFactory`**
  Образец проектирования Singleton, позволяет настроить и получить экземпляр фабрики для производства **`SAXParser`**

- Класс **`SAXParser`**
  Непосредственно транслятор, экземпляры получаются от фабрики **`SAXParserFactory`**

Семантика документа

- Возникающие события
  - `startElement / endElement`
    Открывающий и закрывающий тэг
  - `characters`
    Символы
  - `startDocument / endDocument`
    Начало и конец документа
- Интерфейс `ContentHandler`
  - `startElement()`
  - `endElement()`
  - `characters()`
  - `startDocument()`
  - `endDocument()`

Создание обработчика событий

- Реализация нужного интерфейса, настройка на него используемого транслятора

- Использование класса `org.xml.sax.helpers.DefaultHandler` или `org.xml.sax.helpers.DefaultHandler2` реализующих все интерфейсы обработки событий (все методы имеют пустые тела)

Пример. Файл XML

```
<?xml version="1.0"?>
<company>
    <staff>
        <firstname>Tom</firstname>
        <lastname>King</lastname>
        <nickname>Boss</nickname>
        <salary>100500</salary>
    </staff>
    <staff>
        <firstname>Ben</firstname>
        <lastname>Gum</lastname>
        <nickname>Bubble</nickname>
        <salary>100000</salary>
    </staff>
</company>
```

Пример. Часть 1

```
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

public class ReadXMLFile {
  public static void main(String argv[]) {
    try {
      SAXParserFactory factory = SAXParserFactory.newInstance();
      SAXParser saxParser = factory.newSAXParser();

      DefaultHandler handler = new DefaultHandler() {
      boolean bfname = false;
      boolean blname = false;
      boolean bnname = false;
      boolean bsalary = false;
      ...
```

Пример. Часть 2

```
public void startElement(String uri, String localName, String qName,
                         Attributes attributes) throws SAXException {
  System.out.println("Start Element: " + qName);
  if (qName.equalsIgnoreCase("FIRSTNAME")) {
    bfname = true;
  }
  if (qName.equalsIgnoreCase("LASTNAME")) {
    blname = true;
  }
  if (qName.equalsIgnoreCase("NICKNAME")) {
    bnname = true;
  }
  if (qName.equalsIgnoreCase("SALARY")) {
    bsalary = true;
  }
}
...
```

Пример. Часть 3

```
  ...
  public void endElement(String uri, String localName,
                         String qName) throws SAXException {
    System.out.println("End Element: " + qName);
  }
  public void characters(char ch[], int start, int length)
                          throws SAXException {
    if (bfname) {
      System.out.println("First Name: " +
                    new String(ch, start, length));
      bfname = false;
    }
    if (blname) {
      System.out.println("Last Name: " + new String(ch, start,
length));
      blname = false;
    }
    ...
```

Пример. Часть 4

```
    ...
    if (bnname) {
      System.out.println("Nick Name: " +
                  new String(ch, start, length));
      bnname = false;
    }
    if (bsalary) {
      System.out.println("Salary: " + new String(ch, start, length));
      bsalary = false;
    }
  }
};

saxParser.parse("xmlfile.xml", handler);

} catch (Exception e) {
    e.printStackTrace();
  }
}
```

Пример. Результат

```
Start Element: company
Start Element: staff
Start Element: firstname
First Name: Tom
End Element: firstname
Start Element: lastname
Last Name: King
End Element: lastname
Start Element: nickname
Nick Name: Boss
End Element: nickname
Start Element: salary
Salary: 100500
End Element: salary
End Element: staff
...
```

```
...
Start Element: staff
Start Element: firstname
First Name: Ben
End Element: firstname
Start Element: lastname
Last Name: Gum
End Element: lastname
Start Element: nickname
Nick Name: Bubble
End Element: nickname
Start Element: salary
Salary: 100000
End Element: salary
End Element: staff
End Element: company
```

Работа с DOM

- Считывание документа, опять же, реализует транслятор
- Результат считывания возвращается в виде дерева объектов, реализующих интерфейс `org.w3c.dom.Node`
- Дальнейшая обработка ведется уже на уровне бизнес-логики

Пакет org.w3c.dom

- Базовый интерфейс **`Node`**, содержит основные методы работы с узлом
- От него наследуют специфические интерфейсы для конкретных видов узлов:
  - **`Document`**
  - **`Element`**
  - **`Text`**
  - **`Comment`**
  - **`Attr`**
  - и др.
- Каждый интерфейс добавляет новую функциональность (например **`Document`**, является фабрикой для создания остальных узлов)

Пакет javax.xml.parsers

- Класс `DocumentBuilderFactory`
  Образец проектирования Singleton, позволяет настроить и получить экземпляр фабрики для производства `DocumentBuilder`

- Класс `DocumentBuilder`
  Непосредственно транслятор, экземпляры получаются от фабрики `DocumentBuilderFactory`

Пример. Файл XML

```
<?xml version="1.0"?>
<company>
    <staff id="1001">
        <firstname>Tom</firstname>
        <lastname>King</lastname>
        <nickname>Boss</nickname>
        <salary>100500</salary>
    </staff>
    <staff id="1002">
        <firstname>Ben</firstname>
        <lastname>Gum</lastname>
        <nickname>Bubble</nickname>
        <salary>100000</salary>
    </staff>
</company>
```

Пример. Часть 1

```
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.*;

public class ReadXMLFile_DOM {
  public static void main(String argv[]) {
    try {
      File fXmlFile = new File("xmlfile.xml");
      DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
      DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
      Document doc = dBuilder.parse(fXmlFile);
      System.out.println("Root element : " +
        doc.getDocumentElement().getNodeName());

      NodeList nList = doc.getElementsByTagName("staff");
      System.out.println("----------------------------");
      ...
```

Пример. Часть 2

```
    ...
    for (int temp = 0; temp < nList.getLength(); temp++) {
      Node nNode = nList.item(temp);
      System.out.println("\nCurrent Element : " +
nNode.getNodeName());

      if (nNode.getNodeType() == Node.ELEMENT_NODE) {
        Element eElement = (Element) nNode;

        System.out.println("Staff id : " +
               eElement
               .getAttribute("id"));

        System.out.println("First Name : " +
               eElement
               .getElementsByTagName("firstname")
               .item(0)
               .getTextContent());
        ...
```

Пример. Часть 3

```
            System.out.println("Last Name : " +
                    eElement
                    .getElementsByTagName("lastname")
                    .item(0).getTextContent());

            System.out.println("Nick Name : " +
                    eElement
                    .getElementsByTagName("nickname")
                    .item(0).getTextContent());

            System.out.println("Salary : " +
                    eElement
                    .getElementsByTagName("salary")
                    .item(0)
                    .getTextContent());
        }
    }
} catch (Exception e) {
    e.printStackTrace();
  }
 }
}
```

Пример. Результат

```
Root element : company
----------------------------

Current Element : staff
Staff id : 1001
First Name : Tom
Last Name : King
Nick Name : Boss
Salary : 100500

Current Element : staff
Staff id : 1002
First Name : Ben
Last Name : Gum
Nick Name : Bubble
Salary : 100000
```

Запись XML

- Средствами пакета `javax.xml.transform`

- Средствами API третьих фирм JDOM (www.jdom.org)
  - Тот же DOM, но реализованный более дружелюбно для Java
  - Поддерживает XPath и XSLT

Пример. Часть 1

```
import org.w3c.dom.*;
import org.xml.sax.SAXException;
import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.*;
import javax.xml.transform.stream.*;
import java.io.IOException;

public class WriterDOM {
  public static void main(String[] args) {
    try {
      DocumentBuilderFactory factory =
                  DocumentBuilderFactory.newInstance();
      DocumentBuilder builder = factory.newDocumentBuilder();
      Document document = builder.parse("test.xml");
      DOMSource dom_source = new DOMSource(document);
      StreamResult out_stream = new StreamResult("test2.xml");
      ...
```

Пример. Часть 2

```
      ...
      TransformerFactory tFactory = TransformerFactory.newInstance();
      Transformer transformer = tFactory.newTransformer(/* !!!! */);

      // Вспомогательные действия, связанные с тем, что такая
      // элементарная трансформация не "копирует" директиву
      // !DOCTYPE. В зависимости от PUBLIC- или SYSTEM-описания DTD,
      // можно использовать разные свойства transformer'a

      DocumentType docType = document.getDoctype();
      if (docType != null) {
        String systemID = docType.getSystemId();
        String publicID = docType.getPublicId();
        String res = publicID + "\" \"" + systemID;
        transformer.setOutputProperty(OutputKeys.DOCTYPE_SYSTEM,
systemID);
        //transformer.setOutputProperty(OutputKeys.DOCTYPE_PUBLIC,
res);
      }
      ...
```

Пример. Часть 3

```
    ...
    // Прочие настройки преобразователя

      transformer.transform (dom_source, out_stream);
    } catch (ParserConfigurationException e) {
      e.printStackTrace();
    } catch (TransformerConfigurationException e){
      e.printStackTrace();
    } catch (TransformerException e) {
      e.printStackTrace();
    } catch (IOException e) {
      e.printStackTrace();
    } catch (SAXException e) {
      e.printStackTrace();
    }
  }
}
```

Настройка преобразователя

- Метод создания объекта преобразователя `TransformerFactory.newTransformer()` имеет 2 формы:
  - без аргументов – будет создаваться «копия» исходного документа
  - с аргументом типа `Source` – ссылка на загруженный объект xml-документа, в котором описано XSL-преобразование
- Метод `Transformer.setOutputProperty()` позволяет настроить некоторые параметры вывода (см. класс `OutputKeys`)

Пример XSL-преобразования

```
import java.io.*;
import java.net.URISyntaxException;
import javax.xml.transform.*;
import javax.xml.transform.stream.*;

public class XSLTWriterDOM {
  public static void main(String[] args) throws IOException,
        URISyntaxException, TransformerException {

  TransformerFactory factory = TransformerFactory.newInstance();
  Source xslt = new StreamSource(new File("transformation.xsl"));
  Transformer transformer = factory.newTransformer(xslt);

  Source text = new StreamSource(new File("catalog.xml"));
  transformer.transform(text, new StreamResult(new File("out.xml")));
  }
}
```

Размышления на тему

- Итак, что мы научились делать:
  - Считывать информацию из XML-документов
    - SAX
    - DOM
  - Записывать информацию в XML-документы
- Какой еще инструмент был бы удобен?..
- А если бы мы умели записывать и считывать из XML непосредственно объекты Java?..

Шаг 1. Сохранение JavaBeans

- В версии JavaSE 1.4 для объектов JavaBeans появились механизмы, сходные с сериализацией
- Реализовывали их классы `java.beans.XMLEncoder` и `java.beans.XMLDecoder`
- Недостаток: механизм основан на интроспекции, требует соблюдения правил именования и т.д.

Пример. Часть 1

```
XMLEncoder e = new XMLEncoder(
                new BufferedOutputStream(
                 new FileOutputStream("Test.xml")));
e.writeObject(new JButton("Hello, world"));
e.close();
```

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.0" class="java.beans.XMLDecoder">
  <object class="javax.swing.JFrame">
    <void property="name">
      <string>frame1</string>
    </void>
    <void property="bounds">
      <object class="java.awt.Rectangle">
        <int>0</int> <int>0</int>
        <int>200</int> <int>200</int>
      </object>
    </void>
```

Пример. Часть 2

```
 <void property="contentPane">
      <void method="add">
        <object class="javax.swing.JButton">
          <void property="label">
            <string>Hello</string>
          </void>
        </object>
      </void>
    </void>
    <void property="visible">
      <boolean>true</boolean>
    </void>
  </object>
</java>
```

Шаг 2. Java Architecture for XML Binding (JAXB)

- В версии JavaSE 1.5 появились новые механизмы JAXB
- Связанные с ними классы находятся в пакете **`javax.xml.bind`**
- Позволяют производить «сериализацию» объектов и их структур в XML
- Классы объектов должны быть специальным образом подготовлены
- Активно использует механизм аннотаций…

Пример. RootClass

```
import javax.xml.bind.annotation.*;

@XmlRootElement
public class RootClass {
  private int value;

  @XmlElement
  private NodeClass name = new NodeClass();

  public RootClass() {
    value = 0;
    name.setInnerValue("");
  }

  public NodeClass getName() { return name; }
  public int getValue() { return value; }
  public void setValue(int newValue) { value = newValue; }
}
```

Пример. NodeClass (1)

```
public class NodeClass {
  private String innerValue = "";
  private double rval = Math.random();

  public String getInnerValue() {
    return innerValue;
  }

  public void setInnerValue(String newInnerValue) {
    innerValue = newInnerValue;
  }

  public void print() {
    System.out.println(rval);
  }
}
```

Пример. WriterJAXB

```
import javax.xml.bind.*;
import java.io.*;

public class WriterJAXB {
  public static void main(String[] args){
    try {
      RootClass object1 = new RootClass();
      object1.setValue(5);
      object1.getName().setInnerValue("ABC");
      JAXBContext jc = JAXBContext.newInstance(RootClass.class);
      Marshaller m = jc.createMarshaller();
      OutputStream os = new FileOutputStream("test.xml");
      m.marshal(object1, os);
      os.close();
    }
    catch (JAXBException e) {e.printStackTrace();}
    catch (FileNotFoundException e) {e.printStackTrace();}
    catch (IOException e) {e.printStackTrace();}
  }
}
```

Содержимое файла после выполнения (1)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<rootClass>
  <name>
    <innerValue>ABC</innerValue>
  </name>
  <value>5</value>
</rootClass>
```

- Что сохранено:
  - значение **`name`**, помеченное аннотацией
    - значение **`innerValue`**, не помеченное аннотацией
    - значение **`rval`** не сохранено
  - значение **`value`**, не помеченное аннотацией
- Сохранились элементы, являющиеся свойствами JavaBeans

Пример. NodeClass (2)

```
import javax.xml.bind.annotation.*;
public class NodeClass {
  private String innerValue = "";
  @XmlElement
  private double rval = Math.random();

  public String getInnerValue() {
    return innerValue;
  }

  public void setInnerValue(String newInnerValue) {
    innerValue = newInnerValue;
  }

  public void print() {
    System.out.println(rval);
  }
}
```

Содержимое файла после выполнения (2)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<rootClass>
  <name>
    <rval>0.9878295088863659</rval>
    <innerValue>ABC</innerValue>
  </name>
  <value>5</value>
</rootClass>
```

- Сохранились элементы, являющиеся свойствами JavaBeans
- Сохранились элементы, помеченные аннотациями

Пакет javax.xml.bind.annotation

- Содержит разнообразнейшие аннотации, описывающие параметры маршалинга и анмаршалинга
- `@XmlRootElement`
  Обозначает корневой элемент сохраняемой структуры
- `@XmlElement`
  Обозначает поля и свойства (для JavaBeans)
- `@XmlTransient`
  Обозначает то, что поле не будет сохраняться

Пример. ReaderJAXB

```
import javax.xml.bind.*;
import java.io.*;

public class ReaderJAXB {
  public static void main(String[] args) {
    try {
      JAXBContext jc = JAXBContext.newInstance(RootClass.class);
      InputStream is = new FileInputStream("test.xml");
      Unmarshaller um = jc.createUnmarshaller();
      RootClass object2 = (RootClass) um.unmarshal(is);
      System.out.println(object2.getValue());
      System.out.println(object2.getName().getInnerValue());
      is.close();
    }
    catch (JAXBException e) {e.printStackTrace();}
    catch (FileNotFoundException e) {e.printStackTrace();}
    catch (IOException e) {e.printStackTrace();}
  }
}
```

Спасибо за внимание!

NetCracker®
Transforming the Service Layer™

Дополнительные источники

- Арнолд, К. Язык программирования Java [Текст] / Кен Арнолд, Джеймс Гослинг, Дэвид Холмс. – М. : Издательский дом «Вильямс», 2001. – 624 с.
- Вязовик, Н.А. Программирование на Java. Курс лекций [Текст] / Н.А. Вязовик. – М. : Интернет-университет информационных технологий, 2003. – 592 с.
- Эккель, Б. Философия Java [Текст] / Брюс Эккель. – СПб. : Питер, 2011. – 640 с.
- Шилдт, Г. Java 2, v5.0 (Tiger). Новые возможности [Текст] / Герберт Шилдт. – СПб. : БХВ-Петербург, 2005. – 206 с.
- JavaSE at a Glance [Электронный ресурс]. – Режим доступа: http://www.oracle.com/technetwork/java/javase/overview/index.html, дата доступа: 21.10.2011.
- JavaSE APIs & Documentation [Электронный ресурс]. – Режим доступа: http://www.oracle.com/technetwork/java/javase/documentation/api-jsp-136079.html, дата доступа: 21.10.2011.