* 学习目标

* 能够掌握解析XML操作之SAX

* DOM4j--SAX

* 学会说

* 能够了解解析XML操作之Pull

* 看懂里面的代码：理解它是一种拉模式

* 能够安装VMware虚拟机

* 能够在VMware虚拟机安装XP系统

* 能够在XP系统安装Oracle数据库

* 能够使用客户端sqlplus远程登陆到服务器端Oracle数据库

* 能够理解Oracle的概述

* 能够理解SQL语言概述

*DDL DML DQL TCL DCL CCL

* 能够掌握常用的Oracle数据类型

*

varchar2(10),nvarchar2(size).char(10),nchar,number(5,3),Date,timestamp,clob,nclob,blob

---

* 回顾

* 单元测试：JUnit

* 注解：@Test，@Before，@After，@BeforeClass，@AfterClass，@Ignore

TestCase.assertEqualsXX

* 白盒测试

* XML：可扩展的标记语言（Extensable Markup Languae）

* W3C，自我描述性，...

* 语法：根，实体，闭...

* 元素和节点 ，属性

* HelloWorld XML

* 验证：DTD，Schema（导入）

* DOM：Document Object Model(文档对象模型)

* 原理，优点，缺点

* API:Document,Node,Element,Attribute,Text

* dom4j-->2-->CRUD

    * .xml---SAXReader--->Document

    * XMLWriter-->Document--File

* 能够掌握解析XML操作之SAX

* SAX（Simple API for XML）是一种XML解析的替代方法。相比于DOM，SAX是一种速度更快，更有效的方法

* SAX与DOM的比较

* DOM原理： 一次性加载xml文档到内存，不适合大容量的文件读取

* SAX原理： 加载一点，读取一点，处理一点。适合大容量文件的读取

* DOM解析可以任意进行增删改成

* SAX解析只能读取

* DOM解析任意读取任何位置的数据，甚至往回读

* SAX解析只能从上往下，按顺序读取，不能往回读

* DOM解析面向对象的编程方法（Node，Element，Attribute）,Java开发者编码比较简单。

* SAX解析基于事件的编程方法。java开发编码相对复杂。

* Sax Jdk 自带的（org.xml.sax.*）

* 核心API

    * SaxParser：用于读取和解析xml文件

    * DefaultHandler：处理事件的回调类

```
public class MDefaultHandler extends DefaultHandler {
    //文档开始
```

```
    @Override
    public void startDocument() throws SAXException {
        System.out.println("startDocument");
    }

    // 元素的开始
    @Override
    public void startElement(String uri, String localName, String qName, Attrib
        System.out.println("startElement:" + qName + ":" + attributes.getValue(
    }

    // 可以在此获得文本标签
    @Override
    public void characters(char[] ch, int start, int length) throws SAXExceptio
        System.out.println(new String(ch, start, length));
    }

    // 元素的结束
    @Override
    public void endElement(String uri, String localName, String qName) throws S
        System.out.println("endElement:" + qName);
    }

    //文档结束
    public void endDocument() throws SAXException {
        System.out.println("endDocument");
    };
}

@Test
   public void test5() throws Exception {
       // 1 构建SAXParser的对象
       SAXParser parser = SAXParserFactory.newInstance().newSAXParser();
       // 2 通过SAXParser的对象方法parse解析
       parser.parse("./bookstore.xml",new MDefaultHandler());
   }

* 结果:
startDocument
startElement:bookstore:null
```

```
43
44
45 startElement:book:1001
46
47
48 startElement:name:null
49 Java in thinking
50 endElement:name
51
52
53 startElement:price:null
54 90
55 endElement:price
56
57
58 startElement:author:null
59 xiaobai
60 endElement:author
61
62
63 endElement:book
64
65
66 startElement:book:1002
67
68
69 startElement:name:null
70 Java 从入门到奔溃
71 endElement:name
72
73
74 startElement:price:null
75 1
76 endElement:price
77
78
79 startElement:author:null
80 无名氏
81 endElement:author
82
```

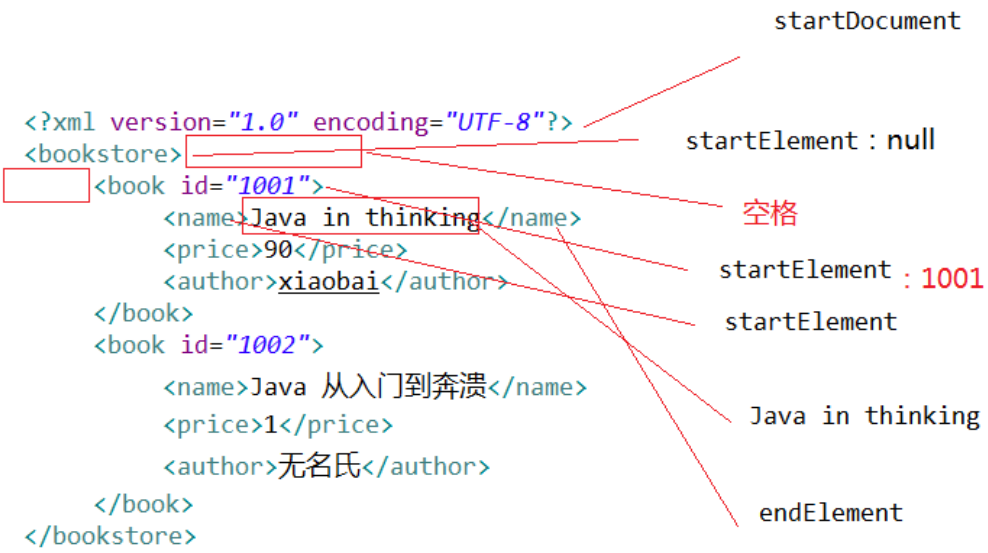
```
83
84 endElement:book
85
86
87 endElement:bookstore
88 endDocument
89
90 * 修改MDefaultHandler
91 public class MDefaultHandler extends DefaultHandler {
92     //文档开始
93     @Override
94     public void startDocument() throws SAXException {
95         System.out.println("startDocument");
96     }
97
98     // 元素的开始
99     @Override
100    public void startElement(String uri, String localName, String qName, Attrib
101        String id=attributes.getValue("id");
102        System.out.println("startElement:" + qName +(id==null?"":":"+id));
103    }
104
105    // 可以在此获得文本标签
106    @Override
107    public void characters(char[] ch, int start, int length) throws SAXExceptio
108        String value=new String(ch, start, length);
109        if(value.trim().length()!=0) {
110            System.out.println(value);
111        }
112    }
113
114    // 元素的结束
115    @Override
116    public void endElement(String uri, String localName, String qName) throws S
117        System.out.println("endElement:" + qName);
118    }
119
120    //文档结束
121    public void endDocument() throws SAXException {
122        System.out.println("endDocument");
```

```
    };
}

* 结果
startDocument
startElement:bookstore
startElement:book:1001
startElement:name
Java in thinking
endElement:name
startElement:price
90
endElement:price
startElement:author
xiaobai
endElement:author
endElement:book
startElement:book:1002
startElement:name
Java 从入门到奔溃
endElement:name
startElement:price
1
endElement:price
startElement:author
无名氏
endElement:author
endElement:book
endElement:bookstore
endDocument

```

* 画图解析



* 能够了解解析XML操作之Pull

* PULL方式解析XML是在Android中推荐使用的一种解析XML的方式。

```
<?xml version="1.0" encoding="UTF-8"?>
<employees>
    <employee id="1001">
        <name>小路</name>
        <age>29</age>
        <address>广州天河</address>
    </employee>
    <employee id="1002">
        <name>老唐</name>
        <age>35</age>
        <address>广州天河</address>
    </employee>
</employees>
package com.lg.test2;
public class Employee {

    private int id;
```

```
    private String name;
    private int age;
    private String address;

    public Employee() {
        super();
    }

    public Employee(int id, String name, int age, String address) {
        super();
        this.id = id;
        this.name = name;
        this.age = age;
        this.address = address;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

```

```
    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    @Override
    public String toString() {
        return "Employee [id=" + id + ", name=" + name + ", age=" + age
                + ", address=" + address + "]";
    }

}

public class XMLUitls {
    public static List<Employee> readXml(InputStream in, String encode) {
        List<Employee> emps = null;
        Employee emp = null;
        try {
            // 创建解析器工厂对象
            XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
            // 从工厂中获取解析器对象
            XmlPullParser parser = factory.newPullParser();
            // 设置解析器输入
            parser.setInput(in, encode);
            /* 解析 */
            // 获取解析到的事件类型
            int eventType = parser.getEventType();
            // 未解析到文档结尾，则循环解析
            while (eventType != XmlPullParser.END_DOCUMENT) {
                switch (eventType) {
                case XmlPullParser.START_DOCUMENT: // 文档开始事件
                    // 创建List集合对象
                    emps = new ArrayList<Employee>();
                    break;
                case XmlPullParser.START_TAG: // 标签开始事件
                    String nodeName = parser.getName().trim(); // 节点名称
                    if ("employee".equals(nodeName)) { // employee节点
```

```
                        emp = new Employee(); // 创建员工对象
                        // 该节点有id属性，则获取id属性
                        int id = Integer.parseInt(parser.getAttributeValue(null
                        emp.setId(id);
                    } else if ("name".equals(nodeName)) { // name节点
                        String name = parser.nextText().trim();
                        emp.setName(name);
                    } else if ("age".equals(nodeName)) { // age节点
                        int age = Integer.parseInt(parser.nextText().trim());
                        emp.setAge(age);
                    } else if ("address".equals(nodeName)) { // address节点
                        String address = parser.nextText();
                        emp.setAddress(address);
                    }
                    break;
                case XmlPullParser.END_TAG:
                    if ("employee".equals(parser.getName().trim())) {
                        emps.add(emp);
                        emp = null;
                    }
                    break;
                }

                eventType = parser.next(); // 切换到下一个解析事件
            }
        } catch (XmlPullParserException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

        return emps;
    }
}

  @Test
   public void test6() throws Exception{
       InputStream in = new FileInputStream("emp.xml");
       List<Employee> list = XMLUitls.readXml(in, "utf-8");
       for (Employee employee : list) {
```

```
            System.out.println(employee);
        }
    }

* 结果:
Employee [id=1001, name=小路, age=29, address=广州天河]
Employee [id=1002, name=老唐, age=35, address=广州天河]
```

* 能够安装VMware虚拟机
    * 参考：安装虚拟机VM
* 能够在VMware虚拟机安装XP系统
    * 参考：在虚拟机上安装XP
* 能够在XP系统安装Oracle数据库
    * 参考：Oracle11G安装
* 能够使用客户端sqlplus远程登陆到服务器端Oracle数据库
    * 参考：Oracle11G远程登陆

* 能够理解Oracle的概述
    * Oracle简介

```
 * Oracle创始人拉里·埃里森
 * 拉里·埃里森，1944年8月17日出生于美国纽约布朗克斯，俄罗斯移民的美国犹太人后裔，
   甲骨文公司（Oracle）的创始人和前任CEO。
 * 1977年6月，埃里森他们三人合伙出资2000美元成立了软件开发研究公司，埃里森拥有60%的股
 * 之后软件开发研究公司为美国中央情报局开发了名为Oracle的数据库，从此名声大噪，
   也因此公司改名为Oracle。
 * 1989年Oracle进入中国以甲骨文命名。
 * 关于拉里·埃里森介绍
  https://www.maigoo.com/mingren/4239.html?fromapp=wx
  https://blog.csdn.net/ensp1/article/details/81546047

* Oracle以数据存储量大，处理速度快，安全性高，容错性强等出色特征，
 长期以来占据着全球数据库市场的主导地位。

```

* 数据库和数据库管理系统

  * 数据库（Database）是按照数据结构来组织、存储和管理数据的仓库。

  * 数据库管理系统(Database Management System)是一种操纵和管理数据库的大型软件，用于建立、使用和维护数据库，简称DBMS。（Oracle就是数据库管理系统）

* 最新数据库流行度排名：

  * 查看链接： http://db-engines.com/en/ranking

352 systems in ranking, September 2019

Rank			DBMS	Database Model	Score		
Sep 2019	Aug 2019	Sep 2018			Sep 2019	Aug 2019	Sep 2018
1.	1.	1.	Oracle	Relational, Multi-model	1346.66	+7.18	+37.54
2.	2.	2.	MySQL	Relational, Multi-model	1279.07	+25.39	+98.60
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1085.06	-8.12	+33.78
4.	4.	4.	PostgreSQL	Relational, Multi-model	482.25	+0.91	+75.82
5.	5.	5.	MongoDB	Document	410.06	+5.50	+51.27
6.	6.	6.	IBM Db2	Relational, Multi-model	171.56	-1.39	-9.50
7.	7.	7.	Elasticsearch	Search engine, Multi-model	149.27	+0.19	+6.67
8.	8.	8.	Redis	Key-value, Multi-model	141.90	-2.18	+0.96
9.	9.	9.	Microsoft Access	Relational	132.71	-2.63	-0.69
10.	10.	10.	Cassandra	Wide column	123.40	-1.81	+3.85
11.	11.	11.	SQLite	Relational	123.36	+0.65	+7.91

* 数据库分类：

  * 关系数据库

    * oracle，mysql，MS sql server，db2，SQLite等

  * 非关系数据库

    * Redis，Solr，ElasticSearch

* 数据库常见服务介绍

```
* OracleOraDb11g_home1TNSListener
    * 监听器服务，服务只有在数据库需要远程访问或使用SQL Developer等工具的时候才需要，
    * 此服务被默认的设置为开机启动(非必须启动))

* OracleService*
   * 数据库服务，是Oracle核心服务该服务，是数据库启动的基础，
   * 只有该服务启动，Oracle数据库才能正常操作。此服务被默认的设置为开机启动。(必须启动)

```

* 能够理解SQL语言概述

```
* SQL语言1974年由Boyce和Chamberlin提出
* SQL（结构化查询语言(Structured Query Language))
* SQL 是一种特殊目的的编程语言，是一种数据库查询和程序设计语言.
* SQL用于存取数据以及查询、更新和管理关系数据库系统。
* SQL语言分类:
 * 数据定义语言(DDL Data Definition Language):
    create创建、alter更改、truncate截断、drop删除
 * 数据操纵语言(DML Data Manipulation Language):
    insert插入、delete删除、update更新
 * 数据查询语言（DQL:Data Query Language）:
    select选择
 * 事务控制语言(TCL Transaction Control Language):
    commit提交、savepoint保存点、rollback回滚
 * 数据控制语言(DCL Data Control Language):
    grant授予、revoke回收
 *指针控制语言（CCL CURSOR Control Language）:
    DECLARE CURSOR(定义游标之类)
```

* 能够掌握常用的Oracle数据类型

```
* VARCHAR2(size):
 * 可变长度的字符串,其最大长度为size个字节;
 * size的最大值是4000,而最小值是1;
 * 必须指定一个VARCHAR2的size;
```

```
 * 英文字母认为是1个字节，中文2个字节

* NVARCHAR2(size)
  * 可变长度的字符串,依据所选的国家字符集,其最大长度为size个字符或字节;
  * size的最大值取决于储存每个字符所需的字节数,其上限为4000;
  * 必须指定一个NVARCHAR2的size;
  * 英文字母认为是2个字节，中文2个字节

* CHAR(size)
    * 固定长度的字符数据,其长度为size个字节;
    * size的最大值是2000字节,而最小值和默认值是1;

* NCHAR(size)
    * 也是固定长度。根据Unicode标准定义
* 建议（char，nchar，varchar，nvarchar）：
 * 如果你肯定存储的数据长度，而且不包中文的，可以选择char类型
 * 如果肯定存储的数据长度，但可能包括中文，可以选择nchar类型。
 * 如果不确定存储的数据长度，存储只有英文、数字的最好用varchar
 * 如果不确定存储的数据长度，也有可能有中文，可以选择nvarchar类型

* NUMBER(p,s)
  * 精度为p并且数值范围为s的数值;
  * 精度p的范围从1到38;
  * 数值范围s的范围是从-84到127;
      * NUMBER(5,2) 表示整数部分最大3位，小数部分为2位;
      * NUMBER(5,-2) 等同number(7)
      * NUMBER 表示使用默认值,即等同于NUMBER(5);
  * NUMBER(p):表示是一个整数，等同NUMBER(p,0)
    * 说明：p多大就是表多少位，如p=10,这个类型可以，存储一个10位整数
  * NUMBER(p,正数):表示是一个小数
    * 说明 p多大就是表多少位,s表示小数点从右到左多少位
    * Number(5,2):999.99
  * NUMBER(p,s负数):表示是一个整数
    * p多大就是表多少位，s表示这个整数增加多少位
    * Number(5,-2) 等同number(7)
  * 整数位是不能超出设置的大小的，小数位如果超出设置的大小会四舍五入
    * NUMBER(5,-3):99999000
    * Number(7,3) :9999.999
    * Number(8,2) :999999.99
```

```
* DATE
  * 有效日期范围从公元前4712年1月1日到公元后9999年12月31日
* timestamp
  * 支持日期加时间，如:‘2019-09-05 12:02:01’

* CLOB
  * 一个字符大型对象,可容纳单字节的字符
  * 最大为4G字节(text)

* NCLOB
  * 一个字符大型对象,可容纳单字节或多字节字符;
  * 最大为4G字节
  * 储存国家字符集

* BLOB
  * 一个二进制大型对象;
  * 最大4G字节

```