**Java 中Properties类的使用**

**1.概述**

1.Properties 继承于 Hashtable。表示一个持久的属性集，属性列表以key-value的形式存在，key和value都是字符串。

public class Properties extends Hashtable<Object,Object> {}

**2.使用场景**

需要避免硬编码的应用场景下需要使用Properties文件来加载程序需要配置的信息，比如JDBC、MyBatis框架。

**3.常用方法**

1.向Properties放入值

public synchronized Object setProperty(String key, String value) {}

2.向Properties取值

public String getProperty(String key) {}

3.向Properties取值,如果key不存在，则使用默认值。

public String getProperty(String key, String defaultValue) {}

4.通过InputStream加载将properties文件信息加载到Properties对象中；

public synchronized void load(InputStream inStream) throws IOException {}

**3.实例**

Mybatis解析全局配置文件中的Setting标签

private void settingsElement(Properties props) {

    configuration.setAutoMappingBehavior(AutoMappingBehavior.valueOf(props.getProperty("autoMappingBehavior", "PARTIAL")));

    configuration.setAutoMappingUnknownColumnBehavior(AutoMappingUnknownColumnBehavior.valueOf(props.getProperty("autoMappingUnknownColumnBehavior", "NONE")));

    configuration.setCacheEnabled(booleanValueOf(props.getProperty("cacheEnabled"), true));

    configuration.setProxyFactory((ProxyFactory) createInstance(props.getProperty("proxyFactory")));

    configuration.setLazyLoadingEnabled(booleanValueOf(props.getProperty("lazyLoadingEnabled"), false));

    configuration.setAggressiveLazyLoading(booleanValueOf(props.getProperty("aggressiveLazyLoading"), false));

    configuration.setMultipleResultSetsEnabled(booleanValueOf(props.getProperty("multipleResultSetsEnabled"), true));

    configuration.setUseColumnLabel(booleanValueOf(props.getProperty("useColumnLabel"), true));

    configuration.setUseGeneratedKeys(booleanValueOf(props.getProperty("useGeneratedKeys"), false));

    configuration.setDefaultExecutorType(ExecutorType.valueOf(props.getProperty("defaultExecutorType", "SIMPLE")));

    configuration.setDefaultStatementTimeout(integerValueOf(props.getProperty("defaultStatementTimeout"), null));

    configuration.setDefaultFetchSize(integerValueOf(props.getProperty("defaultFetchSize"), null));

    configuration.setDefaultResultSetType(resolveResultSetType(props.getProperty("defaultResultSetType")));

    configuration.setMapUnderscoreToCamelCase(booleanValueOf(props.getProperty("mapUnderscoreToCamelCase"), false));

    configuration.setSafeRowBoundsEnabled(booleanValueOf(props.getProperty("safeRowBoundsEnabled"), false));

    configuration.setLocalCacheScope(LocalCacheScope.valueOf(props.getProperty("localCacheScope", "SESSION")));

    configuration.setJdbcTypeForNull(JdbcType.valueOf(props.getProperty("jdbcTypeForNull", "OTHER")));

    configuration.setLazyLoadTriggerMethods(stringSetValueOf(props.getProperty("lazyLoadTriggerMethods"), "equals,clone,hashCode,toString"));

    configuration.setSafeResultHandlerEnabled(booleanValueOf(props.getProperty("safeResultHandlerEnabled"), true));

    configuration.setDefaultScriptingLanguage(resolveClass(props.getProperty("defaultScriptingLanguage")));

    configuration.setDefaultEnumTypeHandler(resolveClass(props.getProperty("defaultEnumTypeHandler")));

    configuration.setCallSettersOnNulls(booleanValueOf(props.getProperty("callSettersOnNulls"), false));

    configuration.setUseActualParamName(booleanValueOf(props.getProperty("useActualParamName"), true));

    configuration.setReturnInstanceForEmptyRow(booleanValueOf(props.getProperty("returnInstanceForEmptyRow"), false));

    configuration.setLogPrefix(props.getProperty("logPrefix"));

    configuration.setConfigurationFactory(resolveClass(props.getProperty("configurationFactory")));

    configuration.setShrinkWhitespacesInSql(booleanValueOf(props.getProperty("shrinkWhitespacesInSql"), false));

    configuration.setArgNameBasedConstructorAutoMapping(booleanValueOf(props.getProperty("argNameBasedConstructorAutoMapping"), false));

    configuration.setDefaultSqlProviderType(resolveClass(props.getProperty("defaultSqlProviderType")));

    configuration.setNullableOnForEach(booleanValueOf(props.getProperty("nullableOnForEach"), false));

  }

**使用DOM生成xml文件**

**1.使用DOM生成xml步骤**

1.获取Document对象

2.通过Document创建元素

3.写入文件（生成对应xml文件）

**2.常用API**

1.XML文档对应的Document对象。

2.在DOM树中，所有节点都是Node对象。Element是Node的最常用的子接口，代表一个XML元素。

1.创建一个元素

 public Element createElement(String tagName)throws DOMException;

 2.为元素设置对应的指定属性值

  public void setAttribute(String name,String value)throws DOMException;

 3.设置元素的文本

  public void setTextContent(String textContent)throws DOMException;

 4.追加元素到指定元素下

 public Node appendChild(Node newChild)throws DOMException;

## 3.使用Dom生成XML

1.生成XML

private static void createDomXml() throws ParserConfigurationException, TransformerFactoryConfigurationError,

  TransformerConfigurationException, TransformerException {

*//1创建一个DocumentBuilderFactory的对象*

 DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

*//2创建一个DocumentBuilder对象*

 DocumentBuilder builder = factory.newDocumentBuilder();

*//3通过DocumentBuilder对象的newDocument()方法创建一个Document对象，该对象代表一个XML文件.*

 Document document = builder.newDocument();

*//4.通过Document对象的createElement()方法创建节点，并为他们赋值。*

 createEle(document);

*//5创建TransformerFactory类的对象*

 TransformerFactory tff = TransformerFactory.newInstance();

*/6.通过TransformerFactory创建Transformer对象*

 Transformer newTransformer = tff.newTransformer();

*/7.设置输出数据时换行和缩进*

 newTransformer.setOutputProperty(OutputKeys.INDENT, "yes");

 newTransformer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount","4");

*//8.使用Transformer对象的transform()方法将DOM树转换为XML文件*

 newTransformer.transform(new DOMSource(document), new StreamResult(new File("D:\\temp\\EmployeeMapper.xml")));

}

2.创建节点

private static void createEle(Document document) {

*//1.创建根节点*

 Element mapper = document.createElement("mapper");

 mapper.setAttribute("namespace", "com.itour.persist.EmployeeMapper");

*//2创建resultMap节点*

 Element resultMap = document.createElement("resultMap");

 resultMap.setAttribute("id", "BaseResultMap");

 resultMap.setAttribute("type", "com.itour.model.Employee");

*//2.创建resultMap下的id节点*

 Element id = document.createElement("id");

 id.setAttribute("column", "EMPNO");

 id.setAttribute("property", "empno");

 resultMap.appendChild(id);

*//3.创建resultMap下的result节点*

 Element ename = document.createElement("result");

 ename.setAttribute("column", "ENAME");

 ename.setAttribute("property", "ename");

 resultMap.appendChild(ename);

 Element job = document.createElement("result");

 job.setAttribute("column", "JOB");

 job.setAttribute("property", "job");

 resultMap.appendChild(job);

 Element mgr = document.createElement("result");

 mgr.setAttribute("column", "MGR");

 mgr.setAttribute("property", "mgr");

 resultMap.appendChild(mgr);

 Element hierdate = document.createElement("result");

 hierdate.setAttribute("column", "HIERDATE");

 hierdate.setAttribute("property", "hierdate");

 resultMap.appendChild(hierdate);

 Element sal = document.createElement("result");

 sal.setAttribute("column", "SAL");

 sal.setAttribute("property", "sal");

 resultMap.appendChild(sal);

 Element comm = document.createElement("result");

 comm.setAttribute("column", "COMM");

 comm.setAttribute("property", "comm");

 resultMap.appendChild(comm);

 Element deptno = document.createElement("result");

 deptno.setAttribute("column", "DEPTNO");

 deptno.setAttribute("property", "deptno");

 resultMap.appendChild(deptno);

*//4.添加resultMap节点到根节点*

 mapper.appendChild(resultMap);

*//5.添加SQL节点*

 Element sql = document.createElement("sql");

 sql.setAttribute("id", "Base\_Column\_List");

 sql.setTextContent("EMPNO,ENAME,JOB,MGR,HIERDATE,SAL,COMM,DEPTNO");

*//6.添加sql节点到根节点*

 mapper.appendChild(sql);

*//7.添加根节点到Document*

 document.appendChild(mapper);

}

**使用Dom4j生成xml**

**1.Dom4j生成xml步骤**

1.添加对应的依赖

<dependency>

 <groupId>dom4j</groupId>

 <artifactId>dom4j</artifactId>

 <version>1.6.1</version>

</dependency>

2.创建document对象

3.创建对应的节点

4.生成对应的xml

**2.常用API**

1.添加节点

Element addElement(String name);

2.为节点添加对应属性

 Element addAttribute(String name, String value);

3.为节点设置文本

 void setText(String text);

**3.生成xml**

private static void CreateXmlDom4j() throws IOException {

*//1.创建document对象*

 Document document = DocumentHelper.createDocument();

*//2.创建节点*

 CreateDom4jEle(document);

*//3.生成XML文件*

 OutputFormat format = OutputFormat.createPrettyPrint();

 format.setEncoding("UTF-8");*// 设置编码格式*

 FileWriter fw = new FileWriter(new File("D:\\temp\\EmpMapper.xml"));

 XMLWriter writer = new XMLWriter(fw, format);

 writer.write(document);

 writer.close();

}

**4.创建节点**

private static void CreateDom4jEle(Document document) {

*//1.创建根节点mapper*

 Element mapper = document.addElement("mapper");

 mapper.addAttribute("namespace", "com.itour.persist.EmployeeMapper");

*//2.在mapper节点下创建resultMap节点*

 Element resultMap = mapper.addElement("resultMap");

   resultMap.addAttribute("id", "BaseResultMap");

*//2.1.在resultMap节点下创建id节点*

 Element id = resultMap.addElement("id");

        id.addAttribute("column", "EMPNO");

        id.addAttribute("property", "empno");

*//2.2.在resultMap节点下创建result节点*

 Element ename = resultMap.addElement("result");

 ename.addAttribute("column", "ENAME");

 ename.addAttribute("property", "ename");

 Element job = resultMap.addElement("result");

 job.addAttribute("column", "JOB");

 job.addAttribute("property", "job");

 Element mgr = resultMap.addElement("result");

 mgr.addAttribute("column", "MGR");

 mgr.addAttribute("property", "mgr");

 Element hierdate = resultMap.addElement("result");

 hierdate.addAttribute("column", "HIERDATE");

 hierdate.addAttribute("property", "hierdate");

 Element sal = resultMap.addElement("result");

 sal.addAttribute("column", "SAL");

 sal.addAttribute("property", "sal");

 Element comm = resultMap.addElement("result");

 comm.addAttribute("column", "COMM");

 comm.addAttribute("property", "comm");

 Element deptno = resultMap.addElement("result");

 deptno.addAttribute("column", "DEPTNO");

 deptno.addAttribute("property", "deptno");

*//3.在mapper节点下创建sql节点*

 Element sql = mapper.addElement("sql");

 sql.addAttribute("id", "Base\_Column\_List");

 sql.setText("EMPNO,ENAME,JOB,MGR,HIERDATE,SAL,COMM,DEPTNO");

}

**使用Dom解析XML**

**1.使用DOM解析xml步骤**

**1.构建Document对象**

public Document parse(File f) throws SAXException, IOException {}

 public Document parse(InputStream is)throws SAXException, IOException {}

 public Document parse(String uri)throws SAXException, IOException {}

 public Document parse(InputStream is, String systemId)throws SAXException, IOException {}

 public abstract Document parse(InputSource is)throws SAXException, IOException;

**2.根据Document获取XML元素**

XML文档对应的Document对象，在Document中包含了一些访问XML文档中节点的方法。

public Element getDocumentElement();

 public NodeList getElementsByTagName(String tagname);

 public Element getElementById(String elementId);

**3.解析xml元素**

在DOM树中，所有节点都是Node对象,Element是Node的最常用的子接口，代表一个XML元素。

1.获取节点的名字

 public String getNodeName();

2.获取节点的文本

 public String getTextContent()throws DOMException;

3.获取节点的相关属性

 public NamedNodeMap getAttributes();

4.获取节点的子节点

 public NodeList getChildNodes();

**2.Dom解析XML**

private static void parseDomXml() throws ParserConfigurationException, SAXException, IOException {

//1.1创建一个DocumentBuilderFactory的对象

DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

//1.2创建一个DocumentBuilder对象

DocumentBuilder builder = factory.newDocumentBuilder();

//1.3通过DocunmentBuilder的parse(…)方法得到Document对象

File f = new File("D:\\temp\\EmployeeMapper.xml");

Document document = builder.parse(f);

//2.1根据Document解析XML元素

Element rootEle = document.getDocumentElement();

System.out.println(rootEle.getTagName());

//2.2获取指定标签的节点元素

NodeList mList = document.getElementsByTagName("sql");

for (int i = 0; i < mList.getLength(); i++) {

//在DOM树中，所有节点都是Node对象，每个Node对象都可能包括nodeName、nodeValue和attributes等属性。

Node item = mList.item(i);

//获取节点的名字

String nodeName = item.getNodeName();

//获取节点的文本值

String text = item.getTextContent();

//获取节点的属性值

NamedNodeMap attributes = item.getAttributes();

Node id = attributes.getNamedItem("id");

System.out.println();

System.out.println("id: "+id+" nodeName: "+nodeName +" text: "+text);

}

//2.3根据元素的id属性获取对应的元素

Element sql = document.getElementById("Base\_Column\_List");

String tagName = sql.getTagName();

System.out.println(tagName);

}

**3.Dom解析XML中遇到问题**

1.根据Document获取XML元素时，使用getElementById（）获取不到对应的元素。未找到对应解决方案。

**使用Dom4j解析xml**

**1.Dom4j解析步骤**

1.添加都有的依赖

<dependency>

 <groupId>dom4j</groupId>

 <artifactId>dom4j</artifactId>

 <version>1.6.1</version>

</dependency>

2.创建解析器对象

3.使用解析器对象读取XML文档生成Document对象

4.解析节点内容

**2.常用API**

 1.获取文档的根节点

 Element getRootElement();

 2.获取节点的名字

 String getName();

 3.获取节点中指定属性的值，如果属性不存在或为空字符串，则为空。

 String attributeValue(String name);

 4.获取元素下的元素列表

 List elements();

 5. 获取元素文本内容

 String getText();

**3.优缺点**

由于DOM解析要求解析器将整个XML文件全部加载到内存中，生成一个Document对象。

1.)元素和元素之间保留结构，关系，可以针对元素进行增删改查操作。

2.）如果XML文件过大，可能会导致内存溢出。

**4.Dom4j解析xml**

private static void dom4jReader() throws DocumentException {

*//1.创建解析器对象*

 SAXReader reader = new SAXReader();

*//2.使用解析器对象读取XML文档生成Document对象*

 Document document = reader.read(new File("D:\\temp\\EmpMapper.xml"));

*//3.1获取根节点*

 Element rootElement = document.getRootElement();

*//3.2获取节点的名字和namespace属性值*

 System.out.println(rootElement.getName());

 System.out.println(rootElement.attributeValue("namespace"));

*//3.3获取根节点下的子节点*

 List<Element> elements = rootElement.elements();

 for (Element ele : elements) {

*//获取节点的名字*

  String name = ele.getName();

*//获取节点的id属性值*

  String rid = ele.attributeValue("id");

  System.out.println(rid);

*//获取标签*

  if("resultMap".equals(name)) {*//解析resultmap标签*

*//获取resultMap节点下的子节点*

   List<Element> resultMap = ele.elements();

   for (Element resultmap : resultMap) {

*//获取子节点下指定的属性*

    System.out.println(resultmap.attributeValue("column")+"--"+resultmap.attributeValue("property"));

   }

  }

  if("sql".equals(name)) {*//解析sql标签*

*//获取sql标准的id和sql标签中的值*

   System.out.println(ele.attributeValue("id")+"---"+ele.getText());

  }

 }

}

**使用SAX解析XML**

**1.SAX解析**

SAX采用事件驱动模式，对内存耗费比较小，可以解析超大XML。

**2.SAX解析xml步骤**

1.获取SAX解析器对象

2.创建事件处理器来解析xml，创建一个类，继承 DefaultHander类，重类中的方法。

3.将被解析的xml文件和事件处理器相关联

**3.使用SAX解析XML**

public class SaxTest {

public static void main(String[] args) throws ParserConfigurationException, SAXException, IOException {

*//1.创建SAX解析器的工厂*

 SAXParserFactory factory = SAXParserFactory.newInstance();

*//2.获取SAX解析器对象*

 SAXParser parse = factory.newSAXParser();

*//3.将被解析的XML文件和事件处理器相关联,*

 parse.parse(new File("D:\\temp\\EmpMapper.xml"), new EmployeeHandler());

}

}

class EmployeeHandler extends DefaultHandler{

*/\*\**

  \* qName:节点名称

  \* attributes:节点的属性相关信息

  \*/

 @Override

 public void startElement(String uri, String localName, String qName, Attributes attributes) throws SAXException {

*// TODO Auto-generated method stub*

  System.out.println("开始解析元素"+qName);

  int length = attributes.getLength();

  for (int i=0;i<length; i++) {

   String qName2 = attributes.getQName(i);

   String value = attributes.getValue(i);

   System.out.println(qName2+"---"+value);

  }

 }

 @Override

 public void characters(char[] ch, int start, int length) throws SAXException {

*// TODO Auto-generated method stub*

  String str = new String(ch,start,length);

  System.out.println(str);

 }

}

**Dom结合XPath解析xml**

**1.基本步骤**

1.使用Dom方式构建Document

2.构建XPath对象

3.使用XPath对象解析节点

**2.代码**

public class XpathTest {

public static void main(String[] args) throws ParserConfigurationException, SAXException, IOException, XPathExpressionException {

*//1.使用Dom方式构建Document对象*

       DocumentBuilder builder = DocumentBuilderFactory.newInstance().newDocumentBuilder();

    Document document = builder.parse(new File("D:\\temp\\EmployeeMapper.xml"));

*//2.生成XPath对象*

    XPath xpath = XPathFactory.newInstance().newXPath();

*//3.获取mapper下的所有节点*

       Node node = (Node)xpath.evaluate("/mapper", document, XPathConstants.NODE);

*//4.获取mapper节点下的子节点*

       NodeList childNodes = node.getChildNodes();

       for (int i = 0; i < childNodes.getLength(); i++) {

*//4.1.获取节点信息*

      Node item = childNodes.item(i);

      if(3!= item.getNodeType()) {

       System.out.println(item.getNodeName()+"--"+item.getNodeType());

      }

      if("sql".equals(item.getNodeName())) {

       System.out.println(item.getTextContent());

      }

*//4.2.获取节点属性值*

   NamedNodeMap attributeNodes = item.getAttributes();

        if (attributeNodes != null) {

            for (int  j= 0; j < attributeNodes.getLength(); j++) {

              Node attribute = attributeNodes.item(j);

              System.out.println(attribute.getNodeName()+"---"+ attribute.getNodeValue());

            }

          }

  }

}

}

# Java解析XML方案

## 1.java解析XML主要方式

XML解析主要有两种方式：

1.基于树的结构来解析，如DOM。

2.基于事件驱动来解析，如Sax。

## 2.Dom方式解析XML

Dom是基于树形结构的 XML 解析方式，它会将整个 XML 文档读入内存并构建一个 DOM 树，基于这棵树形结构对各个节点（Node）进行操作。

### 优缺点

1.将整个Dom树加载到内存中，允许随机读取访问数据，对文档结构进行增删。

2.将整个Dom树加载到内存中且必须一次性解析完成，占内存且耗时。

3.适合文档较小，且需要修改文档内容的场景下使用。

## 3.SAX方式解析XML

SAX基于事件驱动来解析，对xml文档进行解析时,应用程序通过事件处理函数实现对xml文档的解析。

### 优缺点

1.采用事件驱动模式，对内存耗费比较小。

2.不能随机访问xml文档，不支持原地修改xml。

3.适合文档较大，只需要读取文档数据的场景下使用。

## 4.DOM基于XPath解析XML

1.XPath 是一门在 XML 文档中查找信息的语言。XPath 用于在 XML 文档中通过元素和属性进行导航，并对元素和属性进行遍历。

2.使用Dom+XPaht解析时，使用Dom解析XML，使用XPath来查找XML元素信息。

3.Mybatis底层使用了DOM基于XPath来解析Mybatis的配置文件的。