

## DOM写入

**public** **class** CreateDOMXML {

**public** **static** **void** main(String[] args) {

**try** {

//创建DocumentBuilderFactery对象

DocumentBuilderFactory dbf = DocumentBuilderFactory.*newInstance*();

//创建DocumentBuilder对象

DocumentBuilder db = dbf.newDocumentBuilder();

//创建根节点

Document document = db.newDocument();

//创建bookstore节点

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

//创建book节点

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

//设置book的属性

book.setAttribute("id", "1");

//将book节点添加到bookstore节点中

bookstore.appendChild(book);

//将bookstore添加到根节点中

document.appendChild(bookstore);

//创建TransformerFactory对象

TransformerFactory transformerFactory = TransformerFactory.*newInstance*();

**try** {

//创建Transformer对象

Transformer transformer = transformerFactory.newTransformer();

//格式化xml

transformer.setOutputProperty(OutputKeys.***INDENT***, "yes");

transformer.transform(**new** DOMSource(document), **new** StreamResult(**new** File("xml/book1.xml")));

} **catch** (TransformerConfigurationException e) {

e.printStackTrace();

} **catch** (TransformerException e) {

e.printStackTrace();

}

} **catch** (ParserConfigurationException e) {

e.printStackTrace();

}

}

}

## Sax写入

/\*\*

\* SAX创建XML

\* **@author** qinbo

\*

\*/

**public** **class** SAXCreateXmlTest {

/\*\*

\* 创建SAX XML

\*/

**public** **static** **void** createSAXXMl() {

//获取xml内容

List<Book> lsBook = SAXTest.*paseXML*();

//生成xml

//1.创建SAXTransformerFactory对象

SAXTransformerFactory saxTransformerFactory = (SAXTransformerFactory) SAXTransformerFactory.*newInstance*();

**try** {

//2.创建TransformerHandler对象

TransformerHandler transformerHandler = saxTransformerFactory.newTransformerHandler();

//3.创建Transformer对象

Transformer transformer = transformerHandler.getTransformer();

//4.通过Transformer对象设置生成的xml的属性

//设置编码

transformer.setOutputProperty(OutputKeys.***ENCODING***, "UTF-8");

//设置换行

transformer.setOutputProperty(OutputKeys.***INDENT***, "yes");

File file = **new** File("xml/books1.xml");

**if**(!file.exists()) {

file.createNewFile();

}

//5.创建Result对象，并使其与Handler关联

Result result = **new** StreamResult(**new** FileOutputStream(file));

transformerHandler.setResult(result);

//6.利用Handler对象进行xml文件的编写

//打开document

transformerHandler.startDocument();

//创建AttributesImpl对象

AttributesImpl attr = **new** AttributesImpl();

//创建bookstore节点

transformerHandler.startElement("", "", "bookstore", attr);

//遍历lsBook生成xml内容

**for** (Book book : lsBook) {

//创建book节点

attr.clear();

attr.addAttribute("", "", "id", "", book.getId());

transformerHandler.startElement("", "", "book", attr);

attr.clear();

//name节点

**if**(**null** != book.getName() && !"".equals(book.getName().trim())) {

transformerHandler.startElement("", "", "name", attr);

transformerHandler.characters(book.getName().toCharArray(), 0, book.getName().length());

transformerHandler.endElement("", "", "name");

}

//author节点

**if**(**null** != book.getAuthor() && !"".equals(book.getAuthor().trim())) {

transformerHandler.startElement("", "", "author", attr);

transformerHandler.characters(book.getAuthor().toCharArray(), 0, book.getAuthor().length());

transformerHandler.endElement("", "", "author");

}

//year节点

**if**(**null** != book.getYear() && !"".equals(book.getYear().trim())) {

transformerHandler.startElement("", "", "year", attr);

transformerHandler.characters(book.getYear().toCharArray(), 0, book.getYear().length());

transformerHandler.endElement("", "", "year");

}

//price节点

**if**(**null** != book.getPrice() && !"".equals(book.getPrice().trim())) {

transformerHandler.startElement("", "", "price", attr);

transformerHandler.characters(book.getPrice().toCharArray(), 0, book.getPrice().length());

transformerHandler.endElement("", "", "price");

}

//关闭book节点

transformerHandler.endElement("", "", "book");

}

//关闭bookstore节点

transformerHandler.endElement("", "", "bookstore");

//关闭document

transformerHandler.endDocument();

System.***out***.println("生成xml成功！");

} **catch** (TransformerConfigurationException e) {

e.printStackTrace();

} **catch** (FileNotFoundException e) {

e.printStackTrace();

} **catch** (IOException e) {

e.printStackTrace();

} **catch** (SAXException e) {

e.printStackTrace();

}

}

**public** **static** **void** main(String[] args) {

SAXCreateXmlTest.*createSAXXMl*();

}

}

## Dom4J写入

**public** **class** CreateDom4jTest {

**public** **static** **void** main(String[] args) {

//1.创建Document对象

Document document = DocumentHelper.*createDocument*();

//2.添加根节点

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

//向rss中添加属性

rss.addAttribute("version", "2.0");

//3.生成子节点及节点内容

Element channel = rss.addElement("channel");

Element title = channel.addElement("title");

title.addText("国内最新新闻");

//4.设置生成xml的格式

OutputFormat format = OutputFormat.*createPrettyPrint*();

//5.生成xml文件

File file = **new** File("xml/dom4j.xml");

**try** {

XMLWriter writer = **new** XMLWriter(**new** FileOutputStream(file), format);

//设置转义是否生效，默认是true生效

writer.setEscapeText(**false**);

writer.write(document);

writer.close();

} **catch** (UnsupportedEncodingException e) {

e.printStackTrace();

} **catch** (FileNotFoundException e) {

e.printStackTrace();

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

## Jdom写入

**public** **class** CreateJDomTest {

**public** **static** **void** main(String[] args) {

//1.生成根节点

Element rss = **new** Element("rss");

//添加属性

rss.setAttribute("version", "2.0");

//添加子节点和节点内容

Element channel = **new** Element("channel");

Element title = **new** Element("title");

//取消转义

CDATA cdata = **new** CDATA("<国内最新新闻>");

title.addContent(cdata);

channel.addContent(title);

rss.addContent(channel);

//2.生成Document

Document document = **new** Document(rss);

//3.生成xml

**try** {

//设置xml生成格式

Format format = Format.*getPrettyFormat*();

//设置换行

//format.setIndent("");

XMLOutputter outputter = **new** XMLOutputter(format);

outputter.output(document, **new** FileOutputStream(**new** File("xml/jdom.xml")));

} **catch** (FileNotFoundException e) {

e.printStackTrace();

} **catch** (IOException e) {

e.printStackTrace();

}

}

}