XML解析

定义：Extension Markup Language ， 可扩展标记语言，简称XML。

特点

1.XML与操作系统无关。

2.实现不同系统之间的数据交换。

3.XML文件由一系列的标签元素组成。

基本语法

<标签名 属性名="属性值" 属性名="属性值">元素内容</标签名>

注意事项

1.属性值用双引号包裹。

2.一个标签可以有多个属性。

3.属性值中不能包含特殊字符<,>,",',&。

4.XML标签区分大小写。

5.XML必须有正确的嵌套结构。

6.同级标签以缩进对齐。

7.标签名称可以包含字母，数组或者其他的字符。

8.标签名称不能以数字或者标点符号开始。

9.标签名称不能包含空格。

```
<           &lt;
>           &gt;
"           &quot;
'           &apos;
&           &amp;
```

XML案例

```xml
<?xml version="1.0" encoding="UTF-8"?>
<books>
    <book id="1">
        <name>Java高级编程</name>
        <author>张三</author>
        <price>33.5</price>
    </book>
    <book id="2">
        <name>数据结构</name>
        <author>李四</author>
        <price>43.5</price>
    </book>
</books>
```

实际开发中，我们的重点是读取XML文件，而非定义XML的结构，只要你的程序可以读取XML文件包含的数据即可，XML没有特殊的要求。

dom4j是一款优秀的Java XML API（ApplicationInterface），性能优异，功能强大，使用简单。

```java
package com.southwind.test;

import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.io.UnsupportedEncodingException;
import java.util.Iterator;

import org.dom4j.Document;
import org.dom4j.DocumentException;
import org.dom4j.Element;
import org.dom4j.io.SAXReader;
import org.dom4j.io.XMLWriter;

public class Test {
    public static void main(String[] args) {
        delete();
    }
    /*
     * 读取xml
     */
    public static void read() {
        try {
            //1.实例化SAXReader对象
            SAXReader saxReader = new SAXReader();
            //2.读取xml文件，生成Document
            Document document = saxReader.read("src/book.xml");
            //3.获取根节点元素
            Element root = document.getRootElement();
```

```
            //4.迭代根节点
            Iterator<Element> rootIter = root.elementIterator();
            while(rootIter.hasNext()) {
                Element element = rootIter.next();
                String id = element.attributeValue("id");
                System.out.println("id:"+id);
                //5.继续迭代
                Iterator<Element> elementIter = element.elementIterator();
                while(elementIter.hasNext()) {
                    Element attribute = elementIter.next();
                    String name = attribute.getName();
                    String value = attribute.getText();
                    System.out.println(name+":"+value);
                }
            }
        } catch (DocumentException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
    /*
     * 添加节点
     */
    public static void add() {
        try {
            //1.实例化SAXReader对象
            SAXReader saxReader = new SAXReader();
            //2.读取xml文件，生成Document
            Document document = saxReader.read("src/book.xml");
            //3.获取根节点元素
            Element root = document.getRootElement();
            //4.给根节点添加book节点
            Element book = root.addElement("book");
            //5.给book添加id属性
            book.addAttribute("id", "3");
            //6.给book添加name节点
            Element name = book.addElement("name");
            name.addText("MySQL数据库");
            book.addElement("author").addText("王五");
            book.addElement("price").addText("60.5");
            //7.将修改之后的document对象写入book.xml
            OutputStream outputStream = new
FileOutputStream("src/book.xml");
            XMLWriter xmlWriter = new XMLWriter(outputStream);
            xmlWriter.write(document);
            xmlWriter.close();
            outputStream.close();
        } catch (DocumentException e) {
            // TODO Auto-generated catch block
```

```
            e.printStackTrace();
        } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (UnsupportedEncodingException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
    /*
     * 修改xml
     */
    public static void update() {
        try {
            //1.实例化SAXReader对象
            SAXReader saxReader = new SAXReader();
            //2.读取xml文件，生成Document
            Document document = saxReader.read("src/book.xml");
            //3.获取根节点元素
            Element root = document.getRootElement();
            //4.迭代根节点
            Iterator<Element> rootIter = root.elementIterator();
            while(rootIter.hasNext()) {
                Element element = rootIter.next();
                element.addAttribute("type", "计算机");
            }
            //5.将修改之后的document对象写入book.xml
            OutputStream outputStream = new
FileOutputStream("src/book.xml");
            XMLWriter xmlWriter = new XMLWriter(outputStream);
            xmlWriter.write(document);
            xmlWriter.close();
            outputStream.close();
        } catch (DocumentException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (UnsupportedEncodingException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

```
    }
    /*
     * 删除节点
     */
    public static void delete() {
        try {
            //1.实例化SAXReader对象
            SAXReader saxReader = new SAXReader();
            //2.读取xml文件，生成Document
            Document document = saxReader.read("src/book.xml");
            //3.获取根节点元素
            Element root = document.getRootElement();
            //4.迭代根节点
            Iterator<Element> rootIter = root.elementIterator();
            while(rootIter.hasNext()) {
                Element element = rootIter.next();
                String id = element.attributeValue("id");
                if(id.equals("2")) {
                    element.getParent().remove(element);
                }
            }
            //5.将修改之后的document对象写入book.xml
            OutputStream outputStream = new
FileOutputStream("src/book.xml");
            XMLWriter xmlWriter = new XMLWriter(outputStream);
            xmlWriter.write(document);
            xmlWriter.close();
            outputStream.close();
        } catch (DocumentException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (UnsupportedEncodingException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

XML解析+反射的应用

```
package com.southwind.test;
```

```
import java.lang.reflect.Constructor;
import java.lang.reflect.Field;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.util.Iterator;

import org.dom4j.Document;
import org.dom4j.DocumentException;
import org.dom4j.Element;
import org.dom4j.io.SAXReader;

public class Test2 {
    public static void main(String[] args) {
        try {
            //1.读取xml
            SAXReader saxReader = new SAXReader();
            Document document = saxReader.read("src/bean.xml");
            Element root = document.getRootElement();
            String className = root.attributeValue("class");
            Class clazz = Class.forName(className);
            Constructor constructor = clazz.getConstructor(null);
            //2.创建对象
            Object object = constructor.newInstance(null);
            //3.给属性赋值
            Iterator<Element> rootIter = root.elementIterator();
            while(rootIter.hasNext()) {
                Element element = rootIter.next();
                String name = element.getName();
                Field field = clazz.getDeclaredField(name);
                name = "set"+name.substring(0,
1).toUpperCase()+name.substring(1);
                Method method = clazz.getMethod(name, field.getType());
                String value = element.getText();
                Object val = value;
                switch (field.getType().getName()) {
                    case "int":
                        val = Integer.parseInt(value);
                        break;
                    case "float":
                        val = Float.parseFloat(value);
                        break;
                }
                method.invoke(object, val);
            }
            System.out.println(object);
        } catch (DocumentException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
```

```
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (NoSuchMethodException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (SecurityException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (InstantiationException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IllegalArgumentException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (InvocationTargetException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (NoSuchFieldException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```