[***1.声明*** 1](#_Toc264993506)

[***2.引入XSL或CSS*** 1](#_Toc264993507)

[***3.注释*** 1](#_Toc264993508)

[***4.属性*** 1](#_Toc264993509)

[***5.CDATA区*** 2](#_Toc264993510)

[***6.空元素*** 2](#_Toc264993511)

[***7.命名规则*** 2](#_Toc264993512)

[***8.转义字符*** 2](#_Toc264993513)

[***9.命名空间*** 2](#_Toc264993514)

[**第一种写法: 缺省前缀 和 前缀** 3](#_Toc264993515)

[**第二种写法: 每个节点都写前缀** 3](#_Toc264993516)

[**第三种写法: 命名空间全部写入父节点,没有前缀** 3](#_Toc264993517)

[***设计成元素的原则:*** 4](#_Toc264993518)

[***设计成属性的原则:*** 4](#_Toc264993519)

[***XSL(显示格式用的)*** 4](#_Toc264993520)

[***DTD(约束XML用的, 没有确定的数据类型)*** 4](#_Toc264993521)

[***Schema(约束XML用的)*** 4](#_Toc264993522)

[XML解析 4](#_Toc264993523)

[DOM 5](#_Toc264993524)

[***DOM4J(复杂, 高效)*** 5](#_Toc264993525)

[***JDOM(简单, 效率低)*** 5](#_Toc264993526)

[SAX 7](#_Toc264993527)

[SAXReader 7](#_Toc264993528)

[XML编辑 9](#_Toc264993529)

[用DOM创建XML 9](#_Toc264993530)

[用JDOM修改XML 9](#_Toc264993531)

[XML转换 9](#_Toc264993532)

[XSTL 9](#_Toc264993533)

[客户端转换 9](#_Toc264993534)

[服务端转换 9](#_Toc264993535)

[XML校验 9](#_Toc264993536)

***1.声明***

<?xml version=”1.0” encoding=”utf-8” standalone=”no” ?>

***2.引入XSL或CSS***

<?xml-stylesheet href=”typical.xsl” style=”text/xsl” ?>

<?xml-stylesheet href=”typical.css” style=”text/css” ?>

***3.注释***

<!-- 这就是注释 -->

***4.属性***

属性值必须用双引号引起来.

***5.CDATA区***

<![CDATA[

……………

]]>

//删除xml中的CDATA

// xmlStr = Pattern.compile("<!\\[CDATA\\[").matcher(xmlStr).replaceAll("");

// xmlStr = Pattern.compile("\\]\\]>").matcher(xmlStr).replaceAll("");

***6.空元素***

<disc></disc>

<disc />

***7.命名规则***

全小写, 短横线分割, 元素不能以xml开头, 且不能有空格.

***8.转义字符***

|  |  |
| --- | --- |
| > | &gl; |
| < | &lt; |
| & | &amp; |
| “ | &quot; |
| ‘ | &apos; |
| 空格 | &nbsp; |

***9.命名空间***

|  |
| --- |
| 第一种写法:  <school>  xmlns=”student”  xmlns:C=”class”  <name>bbs</name>  <X:name>主页</X:name>  </school> |
| 第二种写法:  <school>  xmlns:O=”http://bbs.swu.edu.cn/”  xmlns:X=”http://www.swu.edu.cn/”  <O:name>bbs</name>  <X:name>主页</X:name>  </school> |
| 第三种写法:  <school xmlns =”http://bbs.swu.edu.cn/”>  <name>bbs</name>  <name xmlns:X=”http://www.swu.edu.cn/”>  主页  </name>  </school> |

**第一种写法: 缺省前缀 和 前缀**

<school xmlns=”class” xmlns:S=”student”>

<class>

<name>

03级2班

</name>

<S:student >

<S:name>

罗利

</S:name>

</S:student>

</class>

</school>

**第二种写法: 每个节点都写前缀**

<school xmlns:C=”class” xmlns:S=”student”>

<class>

<C:name>

03级2班

</C:name>

<S:student>

<S:name>

罗利

</S:name>

</S:student>

</class>

</school>

**第三种写法: 命名空间全部写入父节点,没有前缀**

<school xmlns=”class”>

<class>

<name>

03级2班

</name>

<student xmlns=”student”>

<name>

罗利

</name>

</student>

</class>

</school>

***设计成元素的原则:***

1. 有子元素
2. 有多行
3. 多次发生
4. 值经常改变
5. 比较重要的
6. 用DTD严格限制的

***设计成属性的原则:***

说明性 或 辅助性的

***XSL(显示格式用的)***

|  |  |
| --- | --- |
| 元素 | element |
| 属性 | property |
| 值 | text |

***DTD(约束XML用的, 没有确定的数据类型)***

|  |  |
| --- | --- |
| ? | 0个或1个 |
| + | 至少有1个 |
| \* | 0个或多个 |
| 缺省 | 有且只有1个 |

|  |  |
| --- | --- |
| 逗号分隔 | 必须按严格的顺序 |
| 空格分隔 | 没有严格的顺序 |

***Schema(约束XML用的)***

# XML解析

|  |  |
| --- | --- |
| DOM | 一次把所有数据读入内存 |
| SAX | 不必一次把所有数据读入内存, 可分析大型XML文件, 常用于Server-side的XML—xhtml转换. |
| StAX | 基于一种"拉分析"模型. |

## DOM解析

|  |
| --- |
| 使用DOM API解析XML文档.  解析器读入整个文档数据, 生成一个常驻内存的树结构, 然后我们的程序代码就可以使用DOM接口来操作这个数结构. 既可以遍历树, 也可以删除树的几个部分, 还可以重新排列树和添加新的分支等. |

***DOM4J***解析 ***(复杂, 高效)***

|  |
| --- |
| **DOM4J是一套第三方类库, 工作原理基于DOM. 相对于DOM的一个优势是支持按照相对路径和绝对路径来检索节点.** |

### 写xml

|  |
| --- |
| **import** java.io.FileWriter;  **import** java.io.IOException;  **import** org.dom4j.Document;  **import** org.dom4j.DocumentHelper;  **import** org.dom4j.Element;  **import** org.dom4j.io.OutputFormat;  **import** org.dom4j.io.XMLWriter;  **public** **class** WriteXml {  **public** Document createDocument() {  Document document = DocumentHelper.*createDocument*();  Element root = document.addElement("root");    Element header = root.addElement("head").addText("head");  Element body = root.addElement("body").addText("body");  Element foot = root.addElement("foot").addText("foot");    **return** document;  }  /\*\*  \* 以一种紧凑的格式写入System.out对象  \*  \* **@param** document  \* **@throws** IOException  \*/  **public** **void** writeXmlCompaction(Document document, String xmlFileName) **throws** IOException {  OutputFormat format = OutputFormat.*createCompactFormat*();  format.setEncoding("UTF-8"); // 指定XML编码  XMLWriter writer = **new** XMLWriter(**new** FileWriter(xmlFileName), format);  writer.write(document);  writer.close();  }    /\*\*  \* 以一种优雅的格式写入System.out对象  \*  \* **@param** document  \* **@throws** IOException  \*/  **public** **void** writeXmlFormat(Document document, String xmlFileName) **throws** IOException {  OutputFormat format = OutputFormat.*createPrettyPrint*();  format.setEncoding("UTF-8"); // 指定XML编码  XMLWriter writer = **new** XMLWriter(**new** FileWriter(xmlFileName), format);  writer.write(document);  writer.close();  }      **public** **static** **void** main(String[] args) {  WriteXml xml = **new** WriteXml();  Document document = xml.createDocument();  **try** {  xml.writeXmlCompaction(document, "Compaction.xml");  xml.writeXmlFormat(document, "Format.xml");  } **catch** (IOException e) {  e.printStackTrace();  }  }  } |

***JDOM***解析***(简单, 效率低)***

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <sys-configure>  <jdbc-info>  <driver-class-name>driverClassName</driver-class-name>  <url>url</url>  <user-name>username</user-name>  <password>password</password>  </jdbc-info>  <beans>  <bean id=*"id\_1"* class=*"class\_1"* />  <bean id=*"id\_2"* class=*"class\_2"* />  </beans>  </sys-configure> |
| **package** readConfig;  **import** java.io.IOException;  **import** java.util.Iterator;  **import** java.util.List;  **import** org.jdom.Document;  **import** org.jdom.Element;  **import** org.jdom.JDOMException;  **import** org.jdom.input.SAXBuilder;  **import** org.jdom.xpath.XPath;  /\*\*  \* jdom读取xml配置文件  \* **@author** Administrator  \*  \*/  **public** **class** GlobalConfigure {    **private** **static** GlobalConfigure *instance* = **new** GlobalConfigure();    **private** **static** **final** String *CONFIG\_FILE\_NAME* = "gobal-configure.xml";    **private** Element rootElt;//根节点    /\*\*  \* 读取xml  \*/  **private** GlobalConfigure(){  SAXBuilder sb = **new** SAXBuilder();  **try** {  Document doc = sb.build(Thread.*currentThread*().getContextClassLoader().getResourceAsStream(*CONFIG\_FILE\_NAME*));  **this**.rootElt = doc.getRootElement();  initJdbcInfo();  initBeans();  } **catch** (JDOMException e) {  e.printStackTrace();  } **catch** (IOException e) {  e.printStackTrace();  }  }    **public** **static** GlobalConfigure getInstance(){  **return** *instance*;  }    /\*\*  \* 直接读取xml文件中的某个节点  \*/  **private** **void** initJdbcInfo(){  **try** {  Element url = (Element) XPath.*selectSingleNode*(rootElt, "//sys-configure/jdbc-info/url");  System.*out*.println("url=" + url.getText());  } **catch** (JDOMException e) {  e.printStackTrace();  }  }    /\*\*  \* 直接读取xml中某节点的属性  \* **@param** args  \*/  **private** **void** initBeans(){  **try** {  List beansList = XPath.*selectNodes*(rootElt, "//sys-configure/beans/bean");  **for**(Iterator iter=beansList.iterator(); iter.hasNext(); ){  Element beanElt = (Element)iter.next();  String id = beanElt.getAttributeValue("id");  String className = beanElt.getAttributeValue("class");  System.*out*.println("id=" + id );  System.*out*.println("class=" + className);  }  } **catch** (JDOMException e) {  e.printStackTrace();  }  }    **public** **static** **void** main(String[] args) {  GlobalConfigure.*getInstance*();  }  } |

## SAX解析

|  |
| --- |
| 与DOM不同, SAX基于事件机制来解析XML. |

### SAXReader

|  |
| --- |
| SAXReader reader = new SAXReader();  try {  InputStream is = servletContext.getResourceAsStream("/WEB-INF/web.xml");  Document document = (Document) reader.read(is); //传入一个输入流  String rootName = document.getRootElement().getName();  System.*out*.println("rootName: " + rootName);  } catch (Exception e) {  e.printStackTrace();  } |
| String xmlRealPath = servletContext.getRealPath("/WEB-INF/web.xml");  SAXReader reader1 = new SAXReader();  try {  File xmlFile = new File(xmlRealPath); //传入一个File文件  Document document = (Document) reader1.read(xmlFile);  String rootName = document.getRootElement().getName();  System.*out*.println("rootName: " + rootName);  } catch (Exception e) {  e.printStackTrace();  } |

## AXIOM解析

maven依赖的pom.xml配置

|  |
| --- |
| <groupId>org.apache.axis2</groupId>  <artifactId>axis2</artifactId>  <version>1.5.6</version>  </dependency>    <dependency>  <groupId>axiom-api</groupId>  <artifactId>axiom-api</artifactId>  <version>1.2.12</version>  </dependency>    <dependency> |

# XML编辑

## 用DOM创建XML

## 用JDOM修改XML

# XML转换

## XSTL

## 客户端转换

## 服务端转换

# XML校验

DTD简介

用XDK校验XML