

# 课堂笔记

## 1. XML

### 1.1 xml 简介

- a) xml, eXtensible Markup Language, 可扩展标记语言. 是一种标记语言.
- b) xml 是一种非常灵活的语言, 没有固定的标签, 所有的标签都可以自定义.
- c) 通常, xml 被用于信息的记录和传递. 因此, xml 经常被用于充当配置文件.

### 1.2 格式良好的 xml

- a) 声明信息, 用于描述 xml 的版本和编码方式

```
<?xml version="1.0" encoding="UTF-8"?>
```

- b) xml 有且仅有一个根元素
- c) xml 是大小写敏感的
- d) 标签是成对的, 而且要正确嵌套
- e) 属性值要使用双引号

例如:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!-- 这里是注释 -->
```

```
<books>
```

```
<book id="b01">时间顺流而下，编程逆水行舟
```

```
<name>java高级编程</name>
```

```
<author>张三</author>
```

```
<price>50.5</price>
</book>
<book id="b02">
<name>java中级编程</name>
<author>李四</author>
<price>30.5</price>
</book>
</books>
```

## 2. DTD

### 2.1 DTD 简介

- a) DTD, Document Type Definition, 文档类型定义
- b) DTD 用于约束 xml 的文档格式, 保证 xml 是一个有效的 xml.
- c) DTD 可以分为两种, 内部 DTD, 外部 DTD

### 2.2 使用 DTD

#### 2.2.1 内部 DTD 的定义

- a) 语法如下:

```
<!DOCTYPE 根元素 [元素声明]>
```

- b) 元素声明语法:

```
<!ELEMENT 元素名 (子元素[, 子元素...])>
```

- c) 数量词

- > +: 表示出现 1 次或多次, 至少一次
- > ?: 表示出现 0 次或 1 次 时间顺流而下, 编程逆水行舟
- > \*: 表示出现任意次

- d) 属性声明语法:

＞ 属性类型：CDATA，表示字符数据(character data)

＞ 默认值：

– #REQUIRED，表示必须出现

– #IMPLIED，表示不是必须的

<!ATTLIST 元素名称 属性名称 属性类型 默认值>

e) 带 DTD 的完整 xml 代码：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE scores [
<!ELEMENT scores (student+)>
<!ELEMENT student (name, course, score)>
<!ATTLIST student id CDATA #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT course (#PCDATA)>
<!ELEMENT score (#PCDATA)>
]>
<scores>
<student id="1">
<name>王同</name>
<course>java</course>
<score>89</score>
</student>
<student id="2">
<name>李佳</name>
<course>sql</course>
<score>58</score>
</student>
</scores>
```

## 2.2.2 外部 DTD 的定义

a) 创建一个独立的 dtd 文件

<?xml version="1.0" encoding="UTF-8"?>时间顺流而下，编程逆水行舟

```
<!ELEMENT scores (student+)>
```

```
<!ELEMENT student (name, course, score)>
<!ATTLIST student id CDATA #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT course (#PCDATA)>
<!ELEMENT score (#PCDATA)>
```

b) 在 xml 中引入外部 DTD 文件

```
<!-- 引入外部DTD文件 -->
<!DOCTYPE scores SYSTEM "scores.dtd">
```

## 3. XML 的解析

对 xml 文件进行操作，包括创建 xml，对 xml 文件进行增删

改查操作.

### 3.1 常见的 xml 解析技术

#### 3.1.1 DOM 解析

是官方提供的解析方式，基于 xml 树解析的

#### 3.1.2 SAX 解析

是民间的解析方式，基于事件的解析

#### 3.1.3 JDOM 解析

第三方提供，开源免费的解析方式，比 DOM 解析快

#### 3.1.4 DOM4J

第三方提供，开源免费，是 JDOM 的升级版时间顺流而下，编程逆水行舟

### 3.2 DOM4J 解析 XML

需要导入 dom4j 的 jar 包，解析 xml 的入口，是需要先拿到

## 一个 Document 对象

### 3.2.1 读取 xml 文件中的信息

```
public class TestXml {
    public static void main(String[] args) throws Exception {
        // [1] 创建SAXReader对象, 用于读取xml文件
        SAXReader reader = new SAXReader();
        // [2] 读取xml文件, 得到Document对象
        Document doc = reader.read(new File("src/scores2.xml"));
        // [3] 获取根元素
        Element root = doc.getRootElement();
        // [4] 获取根元素下所有子元素
        Iterator<?> it = root.elementIterator();
        while(it.hasNext()) {
            // 取出元素
            Element e = (Element) it.next();
            System.out.println(e.getName());
            // 获取id属性
            Attribute id = e.attribute("id");
            System.out.println(id.getName() + "=" + id.getValue());
            // 获取student的子元素
            Element name = e.element("name");
            Element course = e.element("course");
            Element score = e.element("score");
            // 打印
            System.out.println(name.getName() + "=" + name.getStringValue());
            System.out.println(course.getName() + "=" + course.getText());
            System.out.println(score.getName() + "=" + score.getText());
            System.out.println("-----");
        }
    }
}
```

### 3.2.2 生成 xml 文件

```
public class TestXml2 {时间顺流而下, 编程逆水行舟
    public static void main(String[] args) throws Exception {
```

```
// [1] 通过DocumentHelper生成一个Document对象
Document doc = DocumentHelper.createDocument();
// [2] 添加并得到根元素
Element root = doc.addElement("books");
// [3] 为根元素添加子元素
Element book = root.addElement("book");
// [4] 为book元素添加属性
book.addAttribute("id", "b01");
// [5] 为book添加子元素
Element name = book.addElement("name");
Element author = book.addElement("author");
Element price = book.addElement("price");
// [6] 为子元素添加文本
name.setText("Thinking in Java");
author.setText("小伟");
price.setText("88");
// [7] 格式良好的输出
OutputFormat format = OutputFormat.createPrettyPrint();
XMLWriter writer = new XMLWriter(new FileWriter(new
File("src/book2.xml")), format);
writer.write(doc);
// [8] 关闭资源
writer.close();
}
}
```