

05-xml

讲师：王振国

今日任务

- 1.XML 语法
- 2.Dom4j 解析

课堂笔记

今天的内容

1.XML 简介

什么是 xml?

xml 是可扩展的标记性语言。

xml 的作用?

xml 的主要作用有：

- 1、用来保存数据，而且这些数据具有自我描述性
- 2、它还可以做为项目或者模块的配置文件
- 3、还可以做为网络传输数据的格式（现在 JSON 为主）。

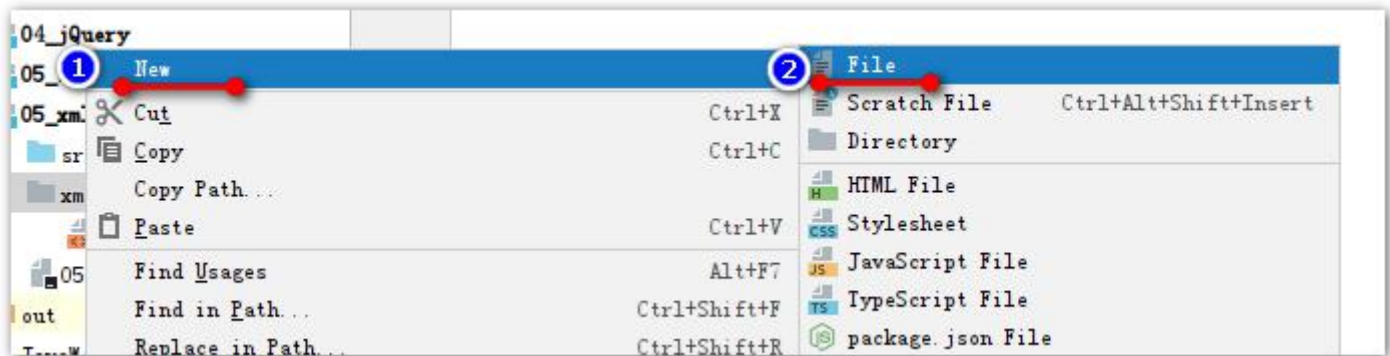
3、xml 语法

1. 文档声明。
2. 元素（标签）
3. xml 属性
4. xml 注释
5. 文本区域（CDATA 区）

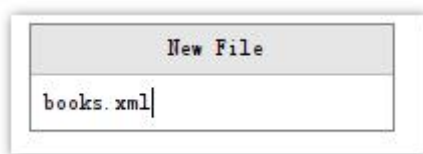
3.1、文档声明

我们先创建一个简单 XML 文件，用来描述图书信息。

1) 创建一个 xml 文件



文件名：



`<?xml version="1.0" encoding="UTF-8"?>` xml 声明。
`<!-- xml 声明 version 是版本的意思 encoding 是编码 -->`
 而且这个`<?xml` 要连在一起写，否则会有报错

属性

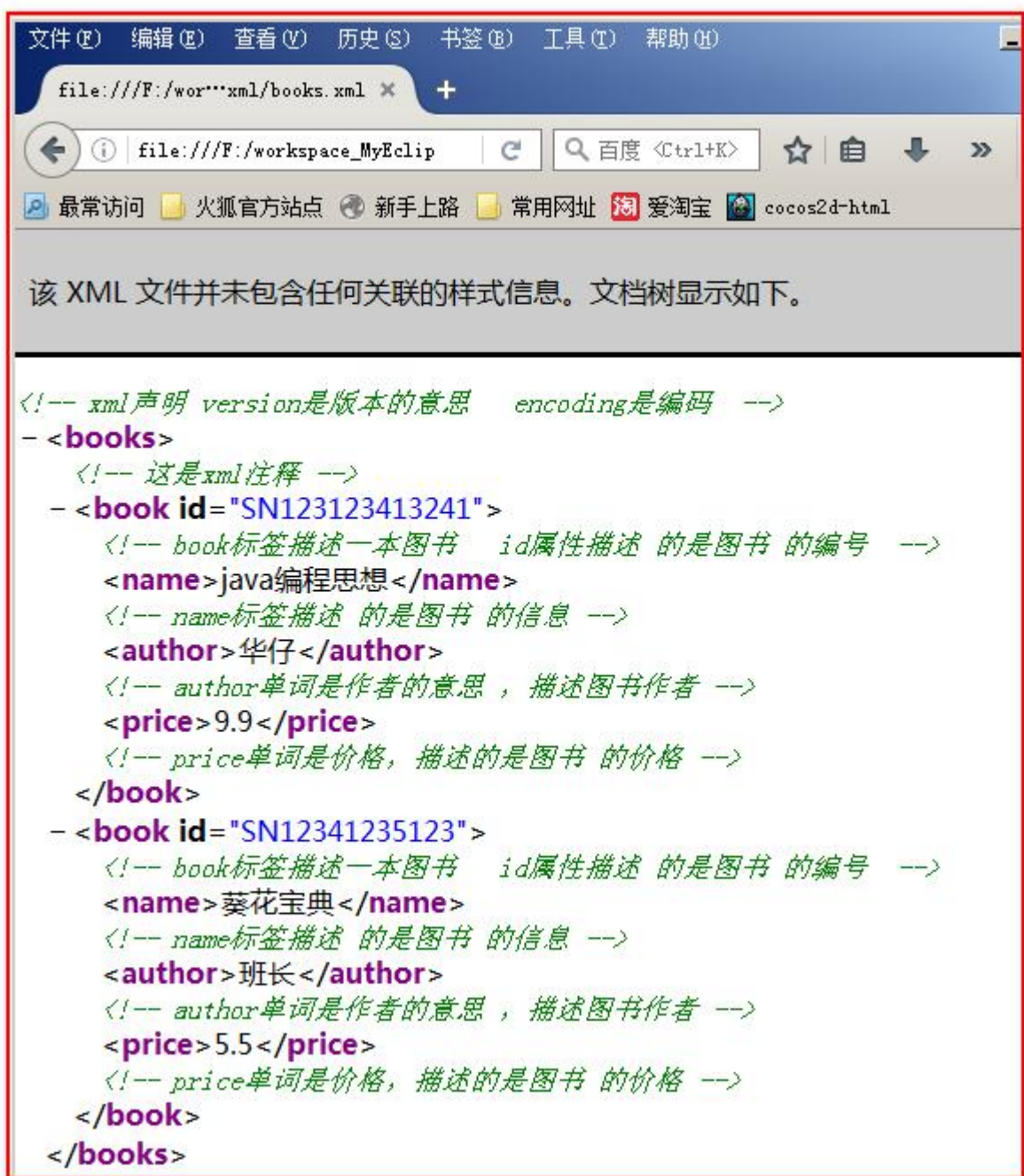
<code>version</code>	是版本号
<code>encoding</code>	是 xml 的文件编码
<code>standalone="yes/no"</code>	表示这个 xml 文件是否是独立的 xml 文件

2) 图书有 id 属性 表示唯一 标识，书名，有作者，价格的信息

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- xml 声明 version 是版本的意思 encoding 是编码 -->
<books> <!-- 这是 xml 注释 -->
```

```
<book id="SN123123413241"> <!-- book 标签描述一本图书 id 属性描述 的是图书 的编号 -->
  <name>java 编程思想</name> <!-- name 标签描述 的是图书 的信息 -->
  <author>华仔</author> <!-- author 单词是作者的意思 ， 描述图书作者 -->
  <price>9.9</price> <!-- price 单词是价格，描述的是图书 的价格 -->
</book>
<book id="SN12341235123"> <!-- book 标签描述一本图书 id 属性描述 的是图书 的编号 -->
  <name>葵花宝典</name> <!-- name 标签描述 的是图书 的信息 -->
  <author>班长</author> <!-- author 单词是作者的意思 ， 描述图书作者 -->
  <price>5.5</price><!-- price 单词是价格，描述的是图书 的价格 -->
</book>
</books>
```

在浏览器中可以查看到文档



3.2、xml 注释

html 和 XML 注释 一样 :<!-- html 注释 -->

3.3、元素（标签）

咱们先回忆一下:

html 标签:

格式: <标签名>封装的数据</标签名>

单标签: <标签名 />
 换行 <hr />水平线

双标签 <标签名>封装的数据</标签名>

标签名大小写不敏感

标签有属性, 有基本属性和事件属性

标签要闭合 (不闭合, html 中不报错。但我们要养成良好的书写习惯。闭合)

1) 什么是 xml 元素

什么是 XML 元素?

XML 元素指的是从 (且包括) 开始标签直到 (且包括) 结束标签的部分。

元素可包含其他元素、文本或者两者的混合物。元素也可以拥有属性。

```
<bookstore>
<book category="CHILDREN">
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
<book category="WEB">
  <title>Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>
</bookstore>
```

在上例中, <bookstore> 和 <book> 都拥有**元素内容**, 因为它们包含了其他元素。<author> 只有**文本内容**, 因为它仅包含文本。

元素是指从开始标签到结束标签的内容。

例如: <title>java 编程思想</title>

元素 我们可以简单的理解为是 标签。

Element 翻译 元素

2) XML 命名规则

XML 元素必须遵循以下命名规则：

2.1) 名称可以含字母、数字以及其他的字符

例如：

```
<book id="SN213412341"> <!-- 描述一本书 -->
    <author>班导</author> <!-- 描述书的作者信息 -->
    <name>java 编程思想</name> <!-- 书名 -->
    <price>9.9</price> <!-- 价格 -->
</book>
```

2.2) 名称不能以数字或者标点符号开始



2.3) 名称不能以字符“xml”（或者 XML、Xml）开始（它是可以的）



2.4) 名称不能包含空格



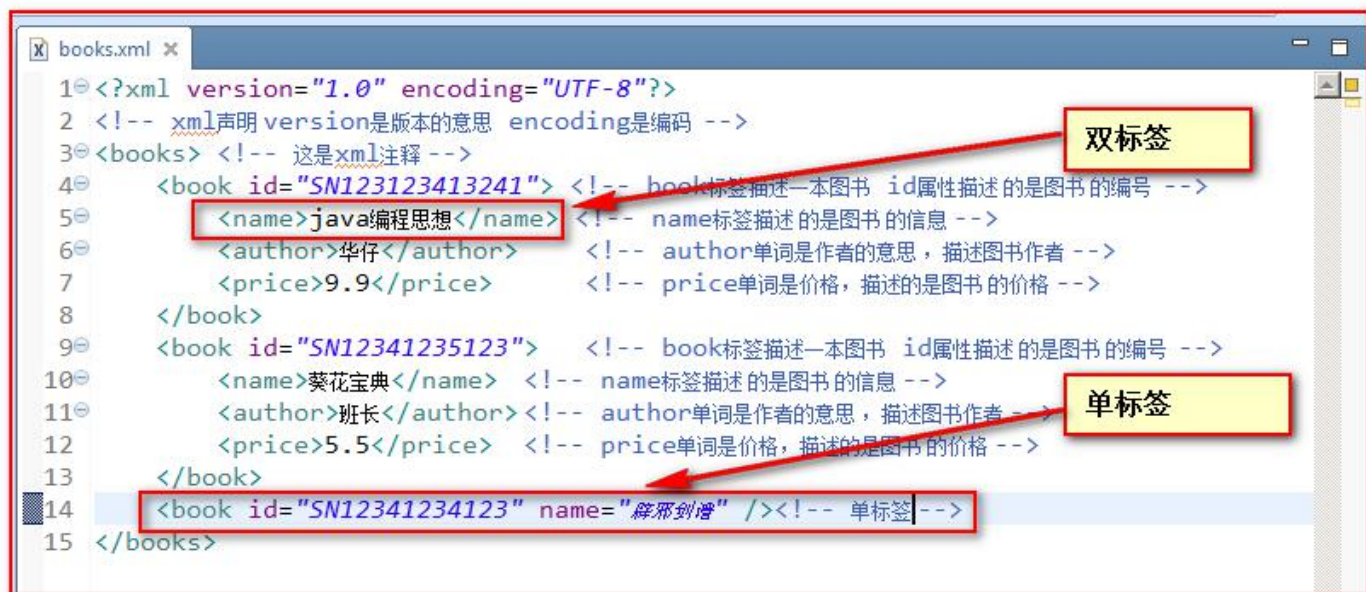
3) xml 中的元素（标签）也 分成 单标签和双标签：

单标签

格式：<标签名 属性="值" 属性="值" />

双标签

格式：< 标签名 属性="值" 属性="值">文本数据或子标签</标签名>



```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- xml声明 version是版本的意思 encoding是编码 -->
3 <books> <!-- 这是xml注释 -->
4   <book id="SN123123413241"> <!-- book标签描述一本图书 id属性描述的是图书的编号 -->
5     <name>java编程思想</name> <!-- name标签描述的是图书的信息 -->
6     <author>华仔</author> <!-- author单词是作者的意思，描述图书作者 -->
7     <price>9.9</price> <!-- price单词是价格，描述的是图书的价格 -->
8   </book>
9   <book id="SN12341235123"> <!-- book标签描述一本图书 id属性描述的是图书的编号 -->
10    <name>葵花宝典</name> <!-- name标签描述的是图书的信息 -->
11    <author>班长</author> <!-- author单词是作者的意思，描述图书作者 -->
12    <price>5.5</price> <!-- price单词是价格，描述的是图书的价格 -->
13  </book>
14  <book id="SN12341234123" name="辟邪剑谱" /> <!-- 单标签 -->
15 </books>
  
```

3.4、xml 属性

xml 的标签属性和 html 的标签属性是非常类似的，**属性可以提供元素的额外信息**

在标签上可以书写属性：

一个标签上可以书写多个属性。**每个属性的值必须使用 引号 引起来。**

的规则和标签的书写规则一致。

XML 元素可以在开始标签中包含属性，类似 HTML。

属性 (Attribute) 提供关于元素的额外信息。

XML 属性

从 HTML，你会回忆起这个：``。"src" 属性提供有关 `` 元素的额外信息。

在 HTML 中（以及在 XML 中），属性提供有关元素的额外信息：

```

<a href="demo.asp">
```

属性通常提供不属于数据组成部分的信息。在下面的例子中，文件类型与数据无关，但是对需要处理这个元素的软件来说却很重要：

```
<file type="gif">computer.gif</file>
```

XML 属性必须加引号

属性值必须被引号包围，不过单引号和双引号均可使用。比如一个人的性别，`person` 标签可以这样写：

```
<person sex="female">
```

或者这样也可以：

```
<person sex='female'>
```

1) 属性必须使用引号引起来，不引会报错示例代码

books.xml

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- xml声明 version是版本的意思 encoding是编码 -->
3 <books> <!-- 这是xml注释 -->
4 <book id=SN123123413241> <!-- book标签描述一本图书 id属性描述的是图书的编号 -->
5 <name>java编程思想</name> <!-- name标签描述的是图书的信息 -->
6 <author>华仔</author> <!-- author单词是作者的意思，描述图书作者 -->
7 <price>9.9</price> <!-- price单词是价格，描述的是图书的价格 -->
8 </book>
9 <book id="SN12341235123"> <!-- book标签描述一本图书 id属性描述的是图书的编号 -->
10 <name>葵花宝典</name> <!-- name标签描述的是图书的信息 -->
11 <author>班长</author> <!-- author单词是作者的意思，描述图书作者 -->
12 <price>5.5</price> <!-- price单词是价格，描述的是图书的价格 -->
13 </book>
14 <book id="SN12341234123" name="辟邪剑谱" /> <!-- 单标签 -->
15 </books>
    
```

xml属性必须用引号引起来

3.5、语法规则：

3.5.1) 所有 XML 元素都须有关闭标签（也就是闭合）



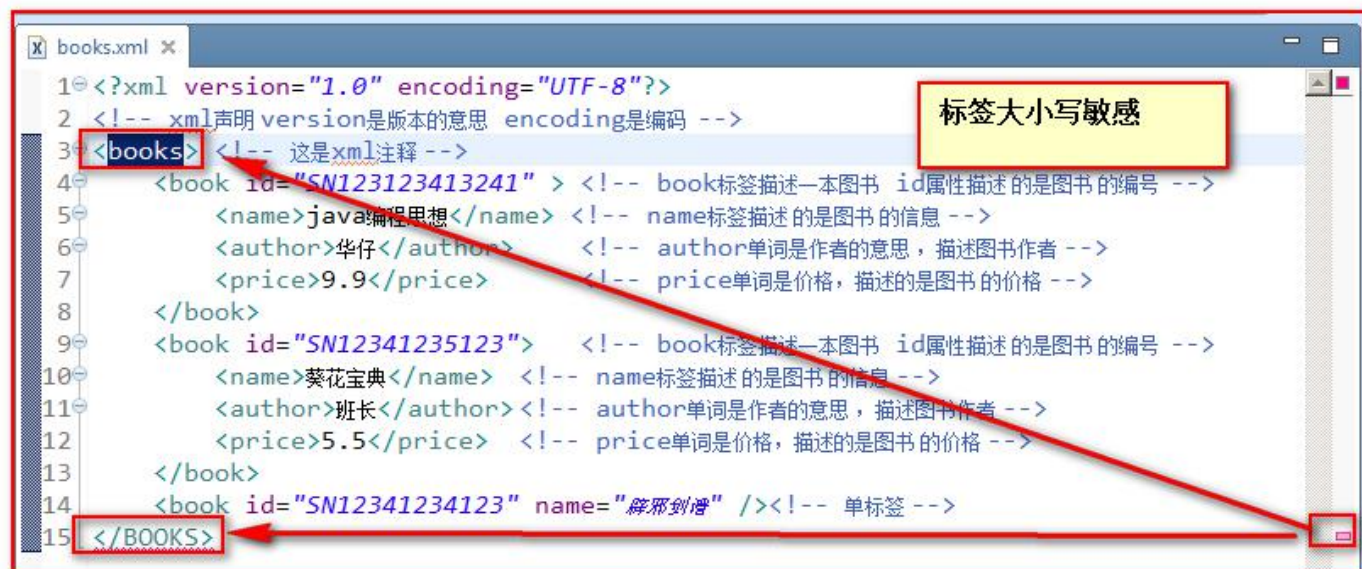
```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- xml声明 version是版本的意思 encoding是编码 -->
3 <books> <!-- 这是xml注释 -->
4   <book id="SN123123413241" > <!-- book标签描述一本书 id属性描述的是图书的编号 -->
5     <name>java编程思想</name> <!-- name标签描述的是图书的信息 -->
6     <author>华仔</author> <!-- author单词是作者的意思，描述图书作者 -->
7     <price>9.9</price> <!-- price单词是价格，描述的是图书的价格 -->
8   </book>
9   <book id="SN12341235123"> <!-- book标签描述一本书 id属性描述的是图书的编号 -->
10    <name>葵花宝典</name> <!-- name标签描述的是图书的信息 -->
11    <author>班长</author> <!-- author单词是作者的意思，描述图书作者 -->
12    <price>5.5</price> <!-- price单词是价格，描述的是图书的价格 -->
13  </book>
14  <book id="SN12341234123" name="辟邪剑谱" /><!-- 单标签 -->
15

```

xml标签一定要闭合。不闭合就会报错

3.5.2) XML 标签对大小写敏感



```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- xml声明 version是版本的意思 encoding是编码 -->
3 <books> <!-- 这是xml注释 -->
4   <book id="SN123123413241" > <!-- book标签描述一本书 id属性描述的是图书的编号 -->
5     <name>java编程思想</name> <!-- name标签描述的是图书的信息 -->
6     <author>华仔</author> <!-- author单词是作者的意思，描述图书作者 -->
7     <price>9.9</price> <!-- price单词是价格，描述的是图书的价格 -->
8   </book>
9   <book id="SN12341235123"> <!-- book标签描述一本书 id属性描述的是图书的编号 -->
10    <name>葵花宝典</name> <!-- name标签描述的是图书的信息 -->
11    <author>班长</author> <!-- author单词是作者的意思，描述图书作者 -->
12    <price>5.5</price> <!-- price单词是价格，描述的是图书的价格 -->
13  </book>
14  <book id="SN12341234123" name="辟邪剑谱" /><!-- 单标签 -->
15 </BOOKS>

```

标签大小写敏感

3.5.3) XML 必须正确地嵌套



```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- xml声明 version是版本的意思 encoding是编码 -->
3 <books> <!-- 这是xml注释 -->
4   <book id="SN123123413241" > <!-- book标签描述一本图书 id属性描述的是图书的编号 -->
5     <name>java编程思想</name> <!-- name标签描述的是图书的信息 -->
6     <author>华仔</author> <!-- author单词是作者的意思，描述图书作者 -->
7     <price>9.9</price> <!-- price单词是价格，描述的是图书的价格 -->
8   </book>
9   <book id="SN12341235123"> <!-- book标签描述一本图书 id属性描述的是图书的编号 -->
10    <name>葵花宝典</name> <!-- name标签描述的是图书的信息 -->
11    <author>班长</author> <!-- author单词是作者的意思，描述图书作者 -->
12    <price>5.5 <!-- price单词是价格，描述的是图书的价格 -->
13  </book>
14  </price>
15  <book id="SN12341234123" name="辟邪剑谱" /> <!-- 单标签 -->
16 </books>
  
```

price标签闭合位置不对

3.5.4) XML 文档必须有根元素

根元素就是顶级元素，

没有父标签的元素，叫顶级元素。

根元素是没有父标签的顶级元素，而且是唯一一个才行。



```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- xml声明 version是版本的意思 encoding是编码 -->
3 <books></books> <!-- 这是xml注释 -->
4   <book id="SN123123413241" > <!-- book标签描述一本图书 id属性描述的是图书的编号 -->
5     <name>java编程思想</name> <!-- name标签描述的是图书的信息 -->
6     <author>华仔</author> <!-- author单词是作者的意思，描述图书作者 -->
7     <price>9.9</price> <!-- price单词是价格，描述的是图书的价格 -->
8   </book>
9   <book id="SN12341235123"> <!-- book标签描述一本图书 id属性描述的是图书的编号 -->
10    <name>葵花宝典</name> <!-- name标签描述的是图书的信息 -->
11    <author>班长</author> <!-- author单词是作者的意思，描述图书作者 -->
12    <price>5.5</price> <!-- price单词是价格，描述的是图书的价格 -->
13  </book>
14  <book id="SN12341234123" name="辟邪剑谱" /> <!-- 单标签 -->
15
  
```

books和book标签同级，整个xml文档，没有根元素，报错

3.5.5) XML 的属性值须加引号



```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- xml声明 version是版本的意思 encoding是编码 -->
3 <books> <!-- 这是xml注释 -->
4   <book id=SN123123413241> <!-- book标签描述一本图书 id属性描述的是图书的编号 -->
5     <name>java编程思想</name> <!-- name标签描述的是图书的信息 -->
6     <author>华仔</author> <!-- author单词是作者的意思，描述图书作者 -->
7     <price>9.9</price> <!-- price单词是价格，描述的是图书的价格 -->
8   </book>
9   <book id="SN12341235123"> <!-- book标签描述一本图书 id属性描述的是图书的编号 -->
10    <name>葵花宝典</name> <!-- name标签描述的是图书的信息 -->
11    <author>班长</author> <!-- author单词是作者的意思，描述图书作者 -->
12    <price>5.5</price> <!-- price单词是价格，描述的是图书的价格 -->
13  </book>
14  <book id="SN12341234123" name="辟邪剑谱" /> <!-- 单标签 -->
15 </books>

```

3.5.6) XML 中的特殊字符



```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- xml声明 version是版本的意思 encoding是编码 -->
3 <books> <!-- 这是xml注释 -->
4   <book id="SN123123413241"> <!-- book标签描述一本图书 id属性描述的是图书的编号 -->
5     <name>java编程思想</name> <!-- name标签描述的是图书的信息 -->
6     <author>华仔</author> <!-- author单词是作者的意思，描述图书作者 -->
7     <price>9.9</price> <!-- price单词是价格，描述的是图书的价格 -->
8   </book>
9   <book id="SN12341235123"> <!-- book标签描述一本图书 id属性描述的是图书的编号 -->
10    <name>葵花宝典</name> <!-- name标签描述的是图书的信息 -->
11    <author>&lt;班长&gt;</author> <!-- author单词是作者的意思，描述图书作者 -->
12    <price>5.5</price> <!-- price单词是价格，描述的是图书的价格 -->
13  </book>
14  <book id="SN12341234123" name="辟邪剑谱" /> <!-- 单标签 -->
15 </books>

```

3.5.7) 文本区域 (CDATA 区)

CDATA 语法可以告诉 xml 解析器，我 CDATA 里的文本内容，只是纯文本，不需要 xml 语法解析

CDATA 格式:

<![CDATA[这里可以把你输入的字符原样显示，不会解析 xml]]>


```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- xml声明 version是版本的意思 encoding是编码 -->
3 <books> <!-- 这是xml注释 -->
4   <book id="SN123123413241" > <!-- book标签描述一本图书 id属性描述的是图书的编号 -->
5     <name>java编程思想</name> <!-- name标签描述的是图书的信息 -->
6     <author>
7       <![CDATA[
8         <<<<<华仔>>>>>
9       ]]>
10    </author> <!-- author单词是作者的意思，描述图书作者 -->
11    <price>9.9</price> <!-- price单词是价格，描述的是图书的价格 -->
12  </book>
13  <book id="SN12341235123"> <!-- book标签描述一本图书 id属性描述的是图书的编号 -->
14    <name>葵花宝典</name> <!-- name标签描述的是图书的信息 -->
15    <author>&lt;班长&gt;</author> <!-- author单词是作者的意思，描述图书作者 -->
16    <price>5.5</price> <!-- price单词是价格，描述的是图书的价格 -->
17  </book>
18  <book id="SN12341234123" name="辟邪剑谱" /> <!-- 单标签 -->
19 </books>
  
```

CDATA文本区，里面的内容不会被解析。只会把它们当成纯文本

2、xml 解析技术介绍

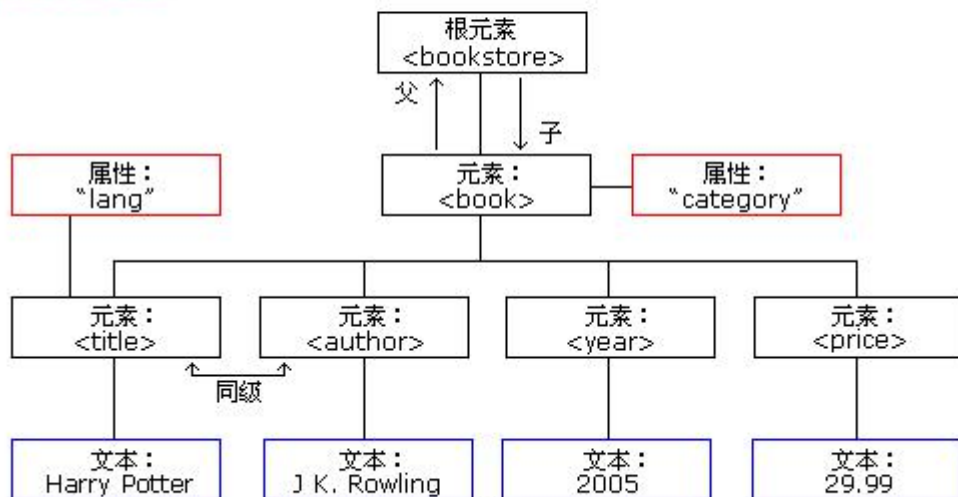
xml 可扩展的标记语言。

不管是 html 文件还是 xml 文件它们都是标记型文档，都可以使用 w3c 组织制定的 dom 技术来解析。

XML 文档对象模型定义访问和操作XML文档的标准方法。

DOM 将 XML 文档作为一个树形结构，而树叶被定义为节点。

开始学习 DOM!



document 对象表示的是整个文档（可以是 **html** 文档，也可以是 **xml** 文档）

早期 **JDK** 为我们提供了两种 **xml** 解析技术 **DOM** 和 **Sax** 简介（**已经过时，但我们需要知道这两种技术**）

dom 解析技术是 **W3C** 组织制定的，而所有的编程语言都对这个解析技术使用了自己语言的特点进行实现。
Java 对 **dom** 技术解析标记也做了实现。

sun 公司在 **JDK5** 版本对 **dom** 解析技术进行升级：**SAX**（**Simple API for XML**）

SAX 解析，它跟 **W3C** 制定的解析不太一样。它是以类似事件机制通过回调告诉用户当前正在解析的内容。
它是一行一行的读取 **xml** 文件进行解析的。不会创建大量的 **dom** 对象。
所以它在解析 **xml** 的时候，在内存的使用上。和性能上。都优于 **Dom** 解析。

第三方的解析：

jdom 在 **dom** 基础上进行了封装、

dom4j 又对 **jdom** 进行了封装。

pull 主要用在 **Android** 手机开发，是在跟 **sax** 非常类似都是事件机制解析 **xml** 文件。

这个 **Dom4j** 它是第三方的解析技术。我们需要使用第三方给我们提供好的类库才可以解析 **xml** 文件。

3、dom4j 解析技术（重点****）

由于 **dom4j** 它不是 **sun** 公司的技术，而属于第三方公司的技术，我们需要使用 **dom4j** 就需要到 **dom4j** 官网下载 **dom4j** 的 **jar** 包。

3.1、Dom4j 类库的使用

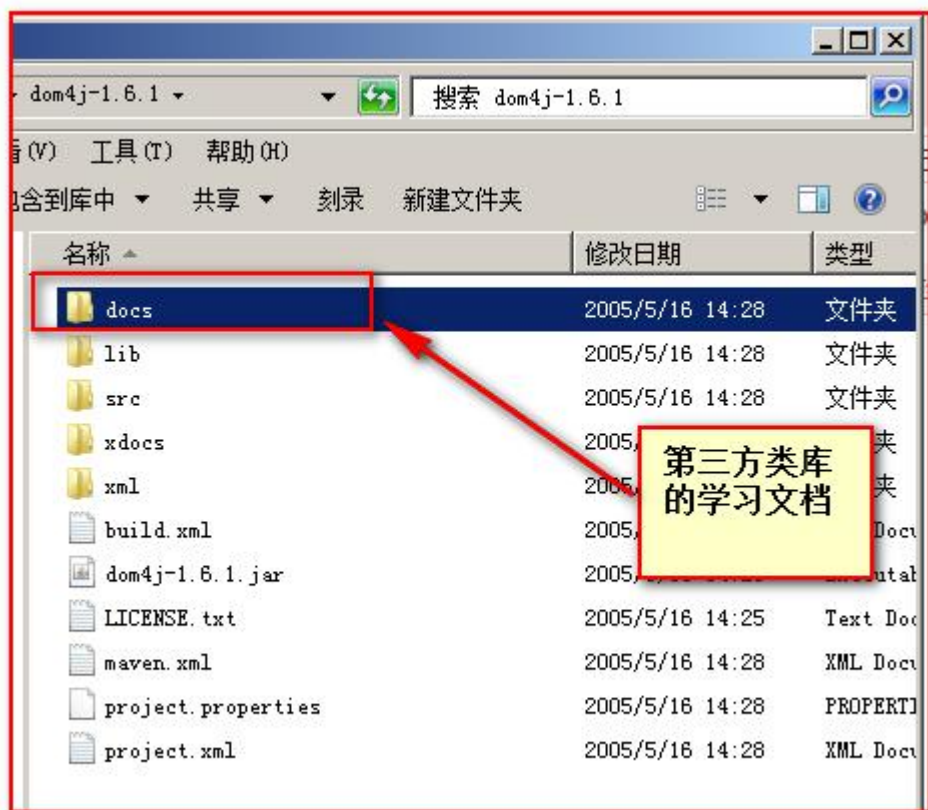


解压后：

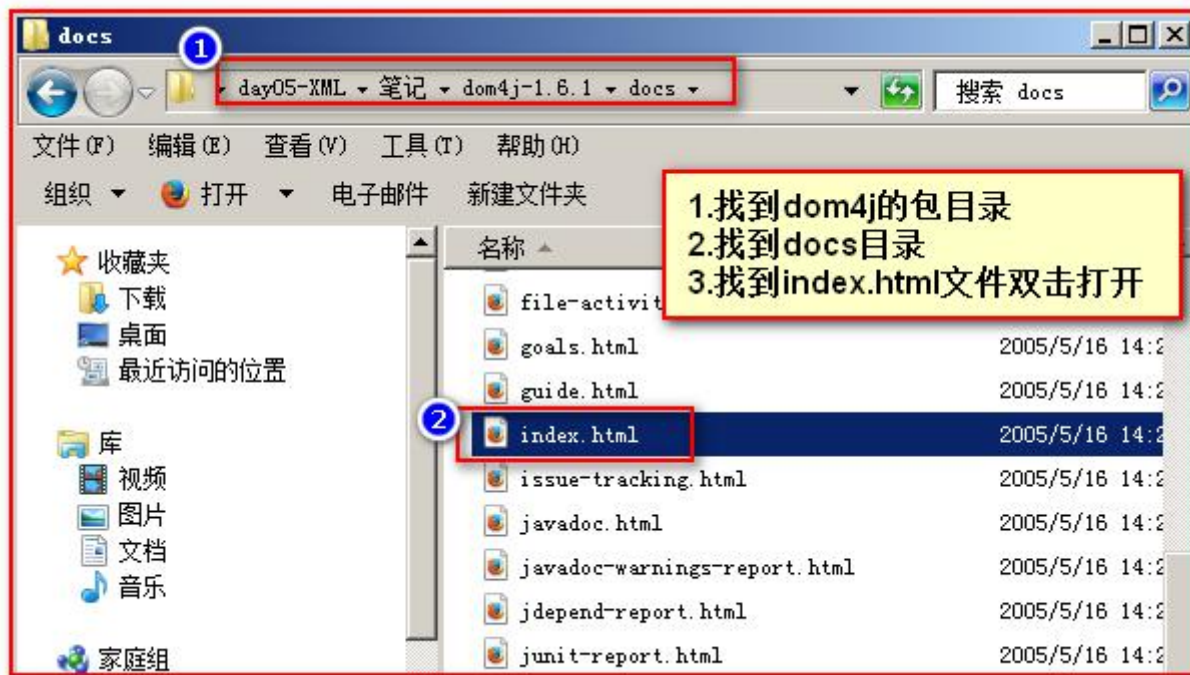


3.2、dom4j 目录的介绍：

1) docs 是文档目录



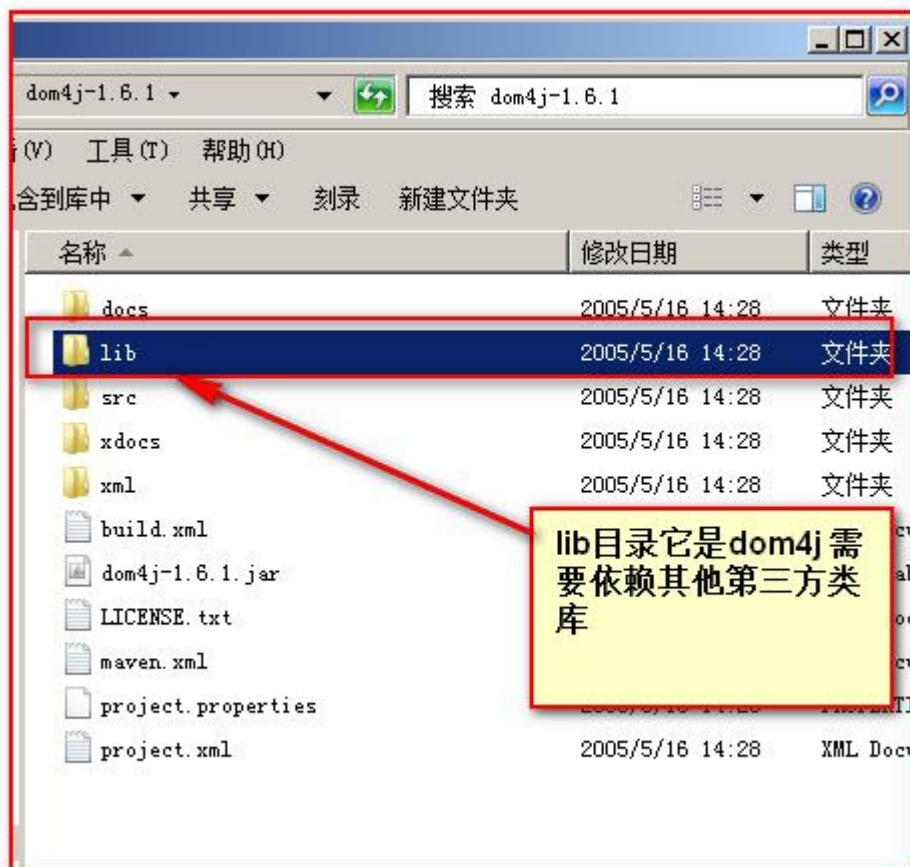
2) 如何查 Dom4j 的文档



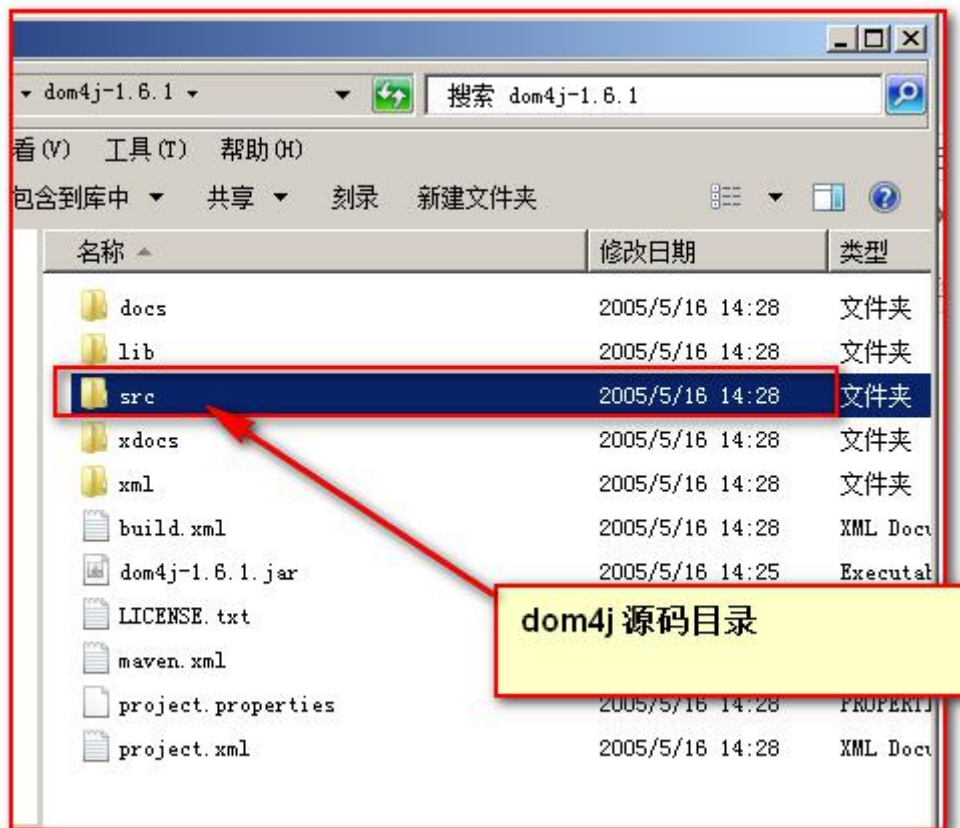
3) Dom4j 快速入门



2) lib 目录



3) src 目录是第三方类库的源码目录:



3.3、dom4j 编程步骤：

第一步：先加载 xml 文件创建 Document 对象

第二步：通过 Document 对象拿到根元素对象

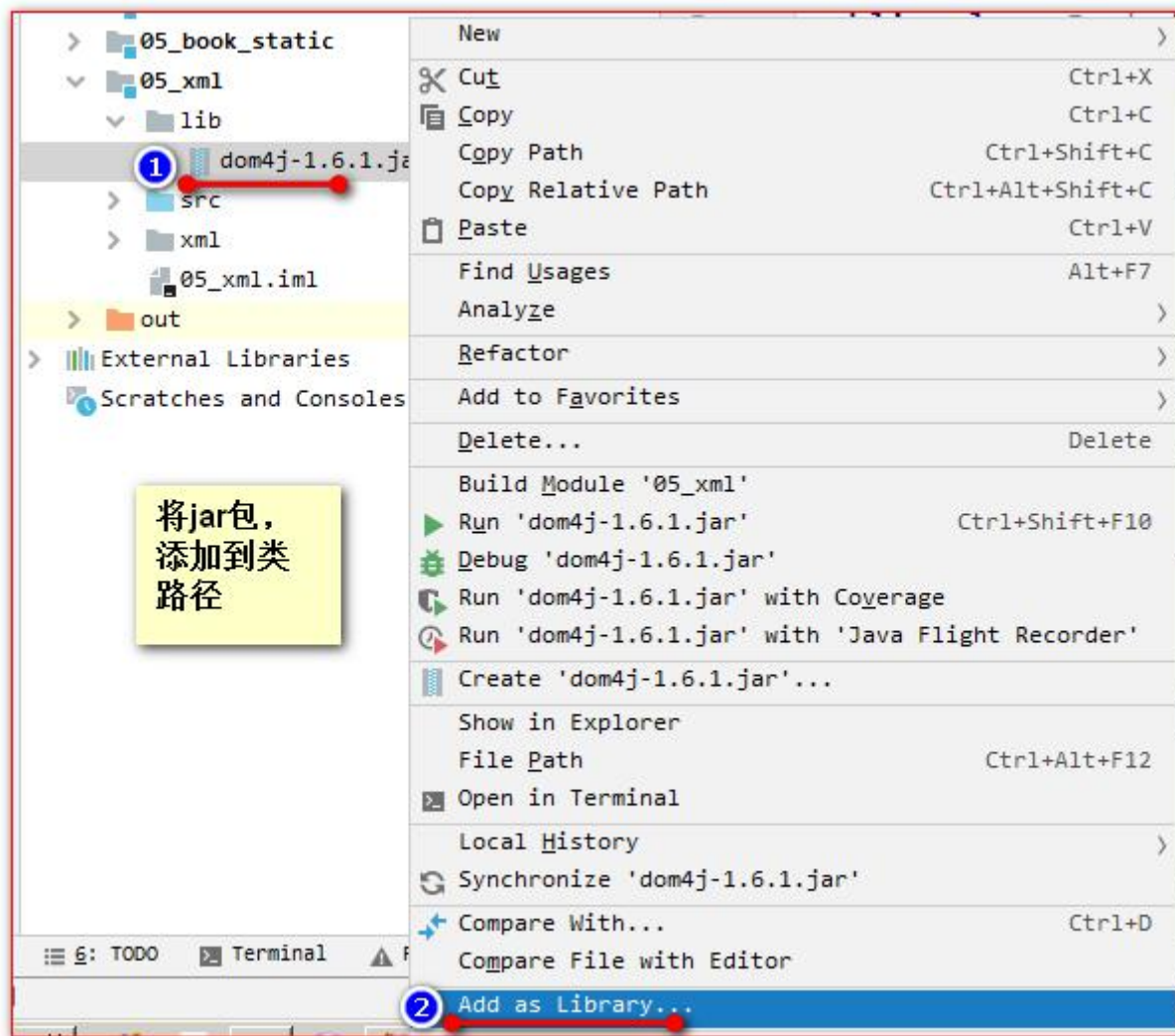
第三步：通过根元素.getElementsByTagName(标签名); 可以返回一个集合，这个集合里放着。所有你指定的标签名的元素对象

第四步：找到你想要修改、删除的子元素，进行相应的操作

第五步：保存到硬盘上

3.4、获取 document 对象

创建一个 lib 目录，并添加 dom4j 的 jar 包。并添加到类路径。



需要解析的 books.xml 文件内容

```
<?xml version="1.0" encoding="UTF-8"?>
<books>
  <book sn="SN12341232">
    <name>辟邪剑谱</name>
    <price>9.9</price>
```

```
<author>班主任</author>
</book>
<book sn="SN12341231">
  <name>葵花宝典</name>
  <price>99.99</price>
  <author>班长</author>
</book>
</books>
```

解析获取 Document 对象的代码

第一步，先创建 SaxReader 对象。这个对象，用于读取 xml 文件，并创建 Document

```
/*
 * dom4j 获取 Document 对象
 */
@Test
public void getDocument() throws DocumentException {
    // 要创建一个 Document 对象，需要我们先创建一个 SAXReader 对象
    SAXReader reader = new SAXReader();
    // 这个对象用于读取 xml 文件，然后返回一个 Document。
    Document document = reader.read("src/books.xml");
    // 打印到控制台，看看是否创建成功
    System.out.println(document);
}
```

3.5、遍历 标签 获取所有标签中的内容（*****重点）

需要分四步操作：

第一步，通过创建 SAXReader 对象。来读取 xml 文件，获取 Document 对象

第二步，通过 Document 对象。拿到 XML 的根元素对象

第三步，通过根元素对象。获取所有的 book 标签对象

第四小，遍历每个 book 标签对象。然后获取到 book 标签对象内的每一个元素，再通过 getText() 方法拿到起始标签和结束标签之间的文本内容

```
/*
 * 读取 xml 文件中的内容
 */
@Test
public void readXML() throws DocumentException {
    // 需要分四步操作：
    // 第一步，通过创建 SAXReader 对象。来读取 xml 文件，获取 Document 对象
    // 第二步，通过 Document 对象。拿到 XML 的根元素对象
    // 第三步，通过根元素对象。获取所有的 book 标签对象
    // 第四小，遍历每个 book 标签对象。然后获取到 book 标签对象内的每一个元素，再通过 getText() 方法拿到
```


起始标签和结束标签之间的文本内容

```
// 第一步，通过创建 SAXReader 对象。来读取 xml 文件，获取 Document 对象
SAXReader reader = new SAXReader();
Document document = reader.read("src/books.xml");
// 第二步，通过 Document 对象。拿到 XML 的根元素对象
Element root = document.getRootElement();
// 打印测试
// Element.asXML() 它将当前元素转换成为 String 对象
// System.out.println( root.asXML() );
// 第三步，通过根元素对象。获取所有的 book 标签对象
// Element.elements(标签名)它可以拿到当前元素下的指定的子元素的集合
List<Element> books = root.elements("book");
// 第四小，遍历每个 book 标签对象。然后获取到 book 标签对象内的每一个元素，
for (Element book : books) {
    // 测试
    // System.out.println(book.asXML());
    // 拿到 book 下面的 name 元素对象
    Element nameElement = book.element("name");
    // 拿到 book 下面的 price 元素对象
    Element priceElement = book.element("price");
    // 拿到 book 下面的 author 元素对象
    Element authorElement = book.element("author");
    // 再通过 getText() 方法拿到起始标签和结束标签之间的文本内容
    System.out.println("书名" + nameElement.getText() + "，价格："
        + priceElement.getText() + "，作者：" + authorElement.getText());
}
}
```

打印内容：

✓ Tests passed: 1 of 1 test - 70ms

C:\Java\jdk1.8.0_102\bin\java.exe ...

Book{sn='SN12341232', name='辟邪剑谱', price=9.9, author='班主任'}

Book{sn='SN12341231', name='葵花宝典', price=99.99, author='班长'}

Process finished with exit code 0

