

知识点列表

编号	名称	描述	级别
1	XML 语法规则	XML 标记文件必须遵循的规则	**
2	DTD 和 Schema	XML 文件的标记约束	**
3	解析 XML 文件	用于解析 XML 文件的 Java API	**

注： **"理解级别 ***"掌握级别 ****"应用级别

目录

1. 标记语言.....	3
2. xml 语法规则.....	6
2.1. xml 的声明	6
2.2. 标记.....	8
2.3. 元素 (Element)	8
2.4. 实体引用 (转义字符)	9
2.5. 属性 (定义在开始标记中的键值对)	10
2.6. CDATA 类型的数据 : 特殊标签.....	11
2.7. 注释 (xml 和 html 相同)	12
2.8. 规则小结	13
2.9. 使用 XML 文件描述数据的例子.....	14
3. DTD/Schema.....	14
3.1. DTD/Schema 的由来.....	14
3.2. 文档类型定义 DTD (Document Type Definition)	15
3.3. Schema.....	24
4. java API 解析 XML 文件.....	26

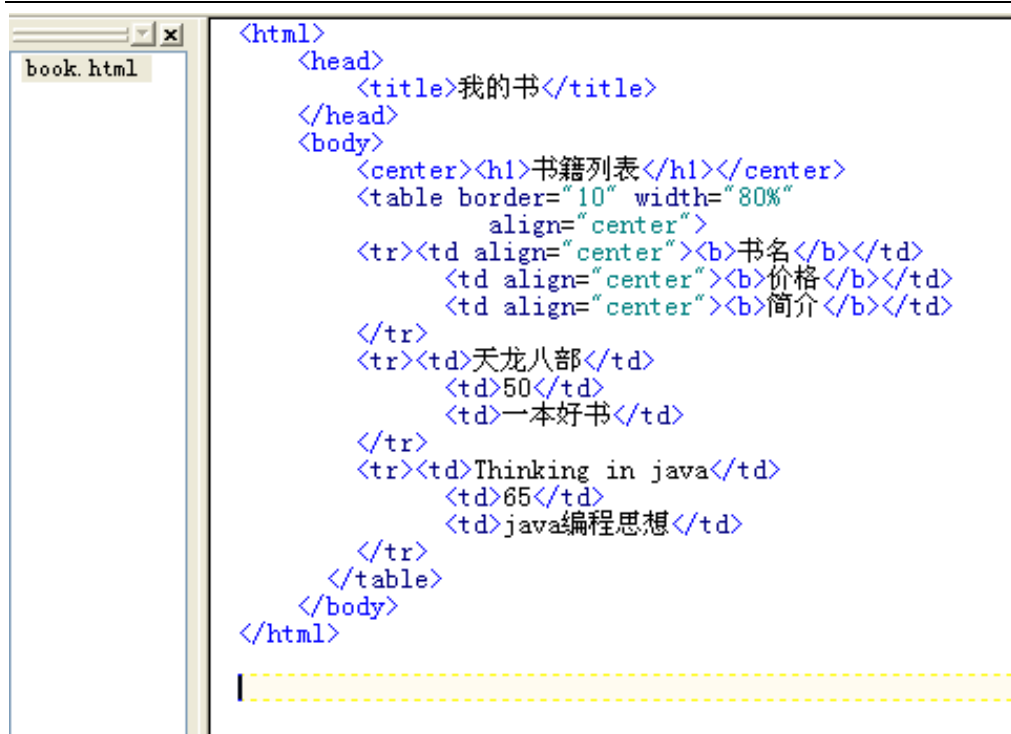
1. 标记语言

标记语言，是一种将文本（Text）以及文本相关的其他信息结合起来，展现出关于文档结构和数据处理细节的电脑文字编码。当今广泛使用的标记语言是超文本标记语言（HyperText Markup Language，HTML）和可扩展标记语言（eXtensible Markup Language，XML）。标记语言广泛应用于网页和网络应用程序。

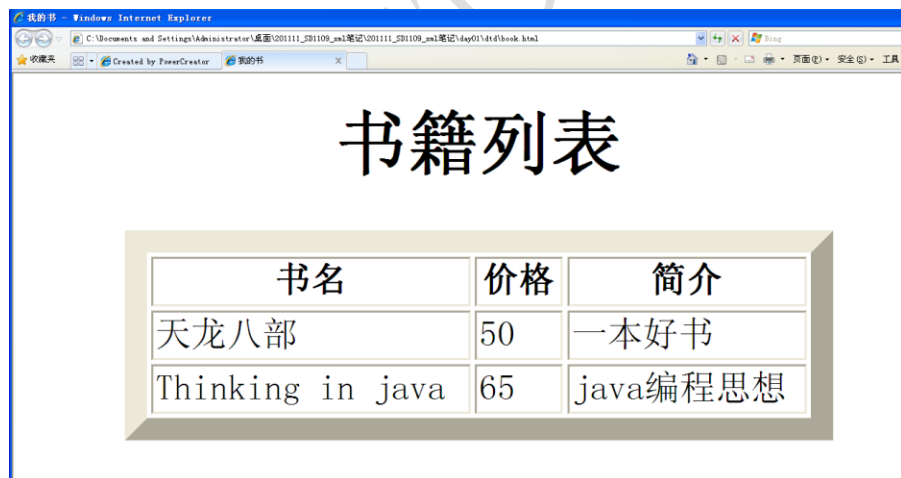
- 1) 超文本标记语言 **HTML**（Hyper Text Markup Language）
 - ✓ 写法格式：`link`
 - ✓ 关注数据的展示与用户体验
 - ✓ 标记是固定的，不可扩展（如 `<a>` 表示超链接）
- 2) 可扩展的标记语言 **XML**（eXtensible Markup Language）
 - ✓ 写法格式：同 html 样式 `<a>link`
 - ✓ 仅关注数据本身
 - ✓ 标记可扩展，可自定义
- 3) Xml 和 Html 语言由同一种父语言 SGML(Standard Generalized Markup Language,标准通用标记语言)发展出来的两种语言。
- 4) 解析器
 - ✓ 专用解析器（比如：XML SPY 专用于解析 XML 文件）
 - ✓ 浏览器
 - ✓ MyEclipse
- 5) W3C（World Wide Web Consortium）
 - ✓ W3C：开源的语言协会，万维网联盟(World Wide Web Consortium)
 - ✓ HTML 和 XML 都是 W3C 制定的语言规则
 - ✓ 官网：www.w3.org
 - ✓ 学习网站：<http://www.w3school.com.cn/>

【案例 1】html 文件演示：book.html

- 使用记事本打开

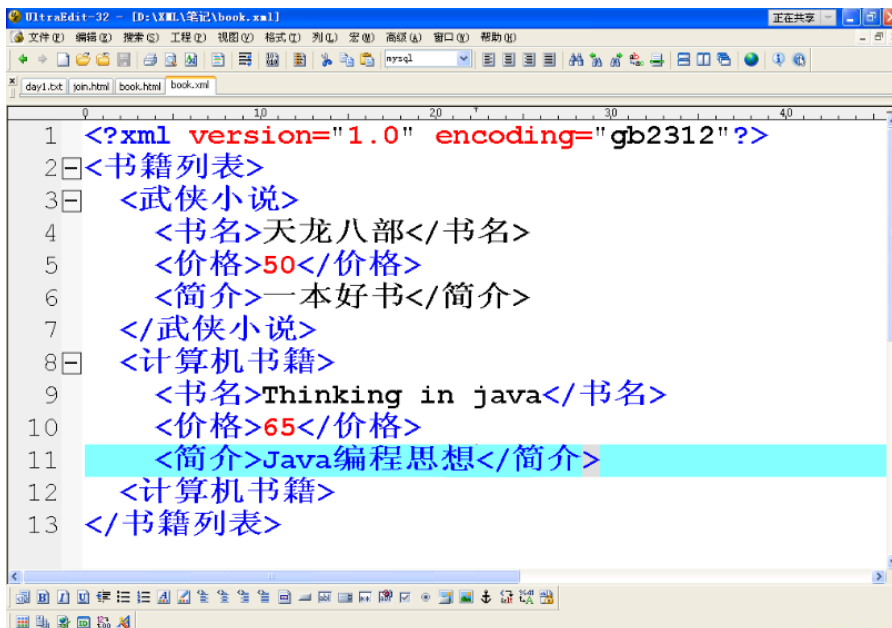


● 使用 IE 浏览器打开



【案例 2】xml 文件演示：book.xml

- 使用文本编辑器 UE 打开



```

1  <?xml version="1.0" encoding="gb2312" ?>
2  <书籍列表>
3  <武侠小说>
4    <书名>天龙八部</书名>
5    <价格>50</价格>
6    <简介>一本好书</简介>
7  </武侠小说>
8  <计算机书籍>
9    <书名>Thinking in java</书名>
10   <价格>65</价格>
11   <简介>Java编程思想</简介>
12 </计算机书籍>
13 </书籍列表>

```

- 使用 MyEclipse 内置浏览器打开



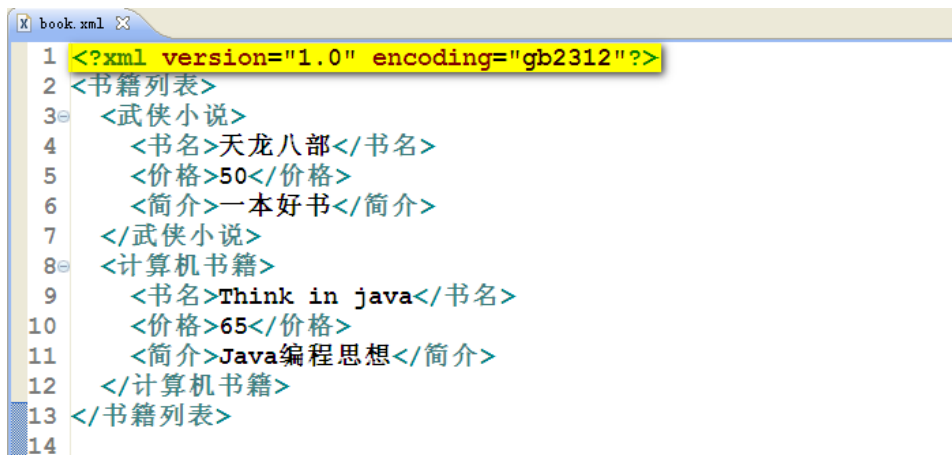
```

<?xml version="1.0" encoding="gb2312" ?>
- <书籍列表>
- <武侠小说>
  <书名>天龙八部</书名>
  <价格>50</价格>
  <简介>一本好书</简介>
</武侠小说>
- <计算机书籍>
  <书名>Think in java</书名>
  <价格>65</价格>
  <简介>Java编程思想</简介>
</计算机书籍>
</书籍列表>

```

2. xml 语法规则

2.1.xml 的声明



```

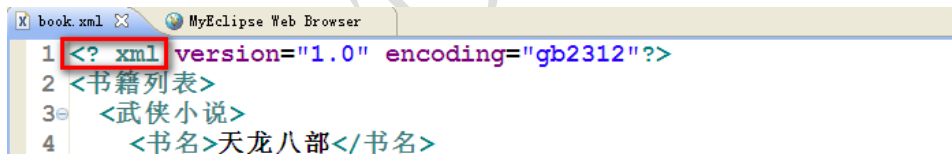
1 <?xml version="1.0" encoding="gb2312"?>
2 <书籍列表>
3   <武侠小说>
4     <书名>天龙八部</书名>
5     <价格>50</价格>
6     <简介>一本好书</简介>
7   </武侠小说>
8   <计算机书籍>
9     <书名>Think in java</书名>
10    <价格>65</价格>
11    <简介>Java编程思想</简介>
12  </计算机书籍>
13 </书籍列表>
14

```

- ✓ xml 的声明必须写在文件第 1 行
- ✓ Encoding (字符集) 属性可以省略，默认的字符集是 utf-8

● 常见错误写法

1) "?"和 xml 之间不能有空格



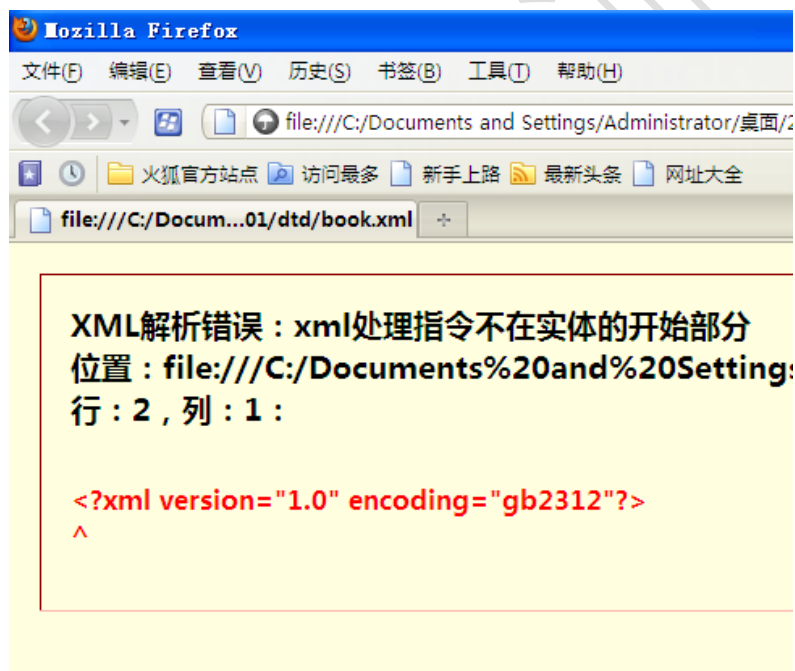
```

1 <? xml version="1.0" encoding="gb2312"?>
2 <书籍列表>
3 <武侠小说>
4 <书名>天龙八部</书名>

```



2) 声明必须顶头写，不能有空行（用 Firefox 浏览器打开）



3) 不要多写空格（Java 程序员的习惯）

浏览器不会报错，但是在 xml 解析时会出问题

```

1 <?xml version = "1.0" encoding="gb2312"?>
2 <书籍列表>
3   <武侠小说>
4     <书名>天龙八部</书名>
5     <价格>50</价格>
6     <简介>一本好书</简介>
7   </武侠小说>
8   <计算机书籍>
9     <书名>Think in java</书名>
10    <价格>65</价格>
11    <简介>Java编程思想</简介>
12  </计算机书籍>
13 </书籍列表>
14

```

2.2. 标记

- 1) 诸如 <书名></书名> 这样格式的被称作标记，标记成对出现
- 2) 标记包括开始标记和结束标记

```

1 <?xml version = "1.0" encoding="gb2312"?>
2 <书籍列表>
3   <武侠小说>
4     <书名>天龙八部</书名>
5     <价格>50</价格>
6     <简介>一本好书</简介>
7   </武侠小说>
8   <计算机书籍>
9     <书名>Think in java</书名>
10    <价格>65</价格>
11    <简介>Java编程思想</简介>
12  </计算机书籍>
13 </书籍列表>
14

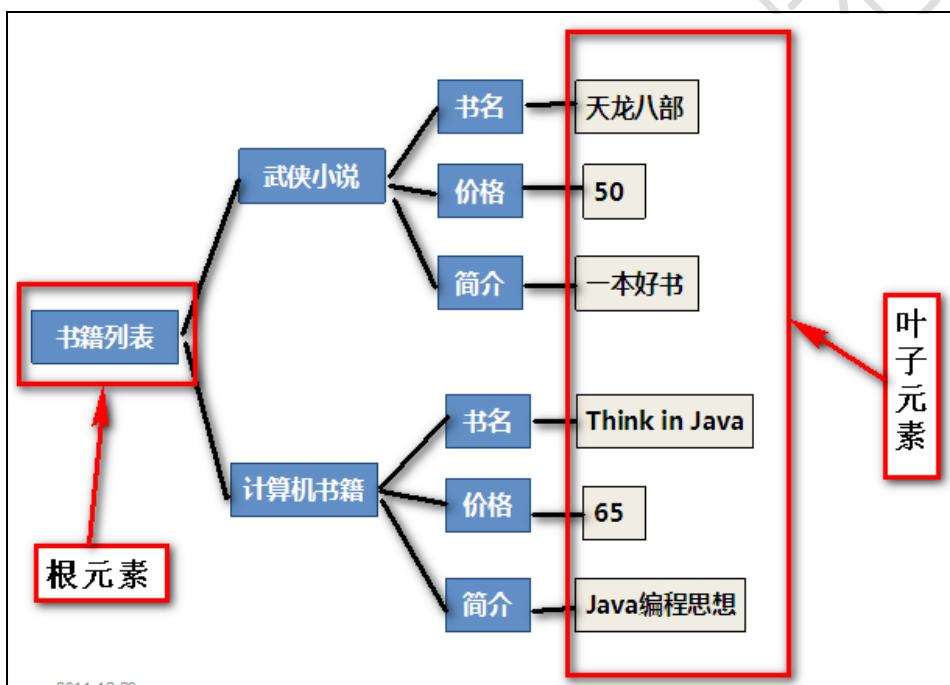
```

2.3. 元素 (Element)

- 1) **元素**：元素包括标记和其中的内容
- 2) **根元素**：最外层的元素叫根元素
- 3) **叶子元素**：最里层的（没有子元素的）元素叫叶子元素

- 4) **空元素**：没有内容的元素叫空元素，比如<a>，可以简写为：<a />
- 5) 元素必须遵循的**语法规则**
 - ✓ 所有的标记都必须有结束
 - ✓ 开始标记和结束标记必须成对出现
 - ✓ 元素必须正确嵌套
 - <a>c (正确)
 - <a>c (错误)
 - ✓ 标记的大小写敏感 Hello 和 hello 不是同一个标记
 - ✓ 有且只能有一个根元素

● 根元素和叶子元素

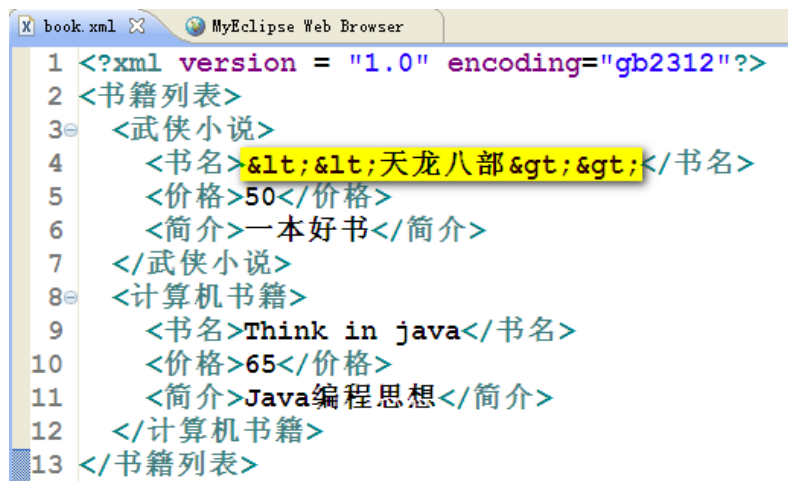


2.4. 实体引用 (转义字符)

- 1) 小于号 (<) : less than --> **<**;
- 2) 大于号 (>) : great than --> **>**;
- 3) And 符号 (&) : **&**;
- 4) 双引号 (") : **"**;
- 5) 单引号 (') : **'**;

注意这些转义字符都是以&开头，以;结尾的

● MyEclipse 内置 xml 编辑器打开



```

1 <?xml version = "1.0" encoding="gb2312"?>
2 <书籍列表>
3   <武侠小说>
4     <书名>&lt;&lt;天龙八部&gt;&gt;</书名>
5     <价格>50</价格>
6     <简介>一本好书</简介>
7   </武侠小说>
8   <计算机书籍>
9     <书名>Think in java</书名>
10    <价格>65</价格>
11    <简介>Java编程思想</简介>
12  </计算机书籍>
13 </书籍列表>
  
```

● MyEclipse 内置浏览器打开



```

<?xml version="1.0" encoding="gb2312" ?>
- <书籍列表>
-   <武侠小说>
      <书名><<天龙八部>></书名>
      <价格>50</价格>
      <简介>一本好书</简介>
    </武侠小说>
-   <计算机书籍>
      <书名>Think in java</书名>
      <价格>65</价格>
      <简介>Java编程思想</简介>
    </计算机书籍>
  </书籍列表>
  
```

2.5. 属性（定义在开始标记中的键值对）

- 1) 格式：属性="属性值"
- 2) 要求：
 - ✓ 属性必须有值
 - ✓ 属性值必须用引号引起来，单引号或双引号都可以，但必须一致

```

1 <?xml version = "1.0" encoding="gb2312"?>
2 <书籍列表>
3   <武侠小说 isbn="1234" hot="true">
4     <书名>&lt;&lt;天龙八部&gt;&gt;</书名>
5     <价格>50</价格>
6     <简介>一本好书</简介>
7   </武侠小说>
8   <计算机书籍 isbn="4567">
9     <书名>Think in java</书名>
10    <价格>65</价格>
11    <简介>Java编程思想</简介>
12  </计算机书籍>
13 </书籍列表>
14

```

- ✓ 武侠小说的属性有 2 组：isbn 号码、是否热销标记
- ✓ 计算机书籍的属性有 2 组：isbn 号码、是否热销

2.6. CDATA 类型的数据：特殊标签

- 1) 格式： <![CDATA [文本内容]]>
- 2) 特殊标签中的实体引用都被忽略，所有内容被当成一整块文本数据对待

```

1 <?xml version="1.0" encoding="gb2312"?>
2 <书籍列表>
3   <武侠小说 isbn="1234" hot="true">
4     <书名>&lt;&lt;天龙八部&gt;&gt;</书名>
5     <简介>
6       <![CDATA[
7         一本好书，
8         没<<笑傲江湖>>好看
9         比<<Think in Java>>好看
10      ]]>
11     </简介>
12   </武侠小说>

```

```

13
14 <computer_book isbn="5678">
15   <书名>Thinking in java</书名>
16   <价格>65</价格>
17   <简介>Java编程思想</简介>
18 </computer_book>
19
20 </书籍列表>

```

注：CDATA 表示是一整块，其中包括的特殊字符，比如<<不是全角的《

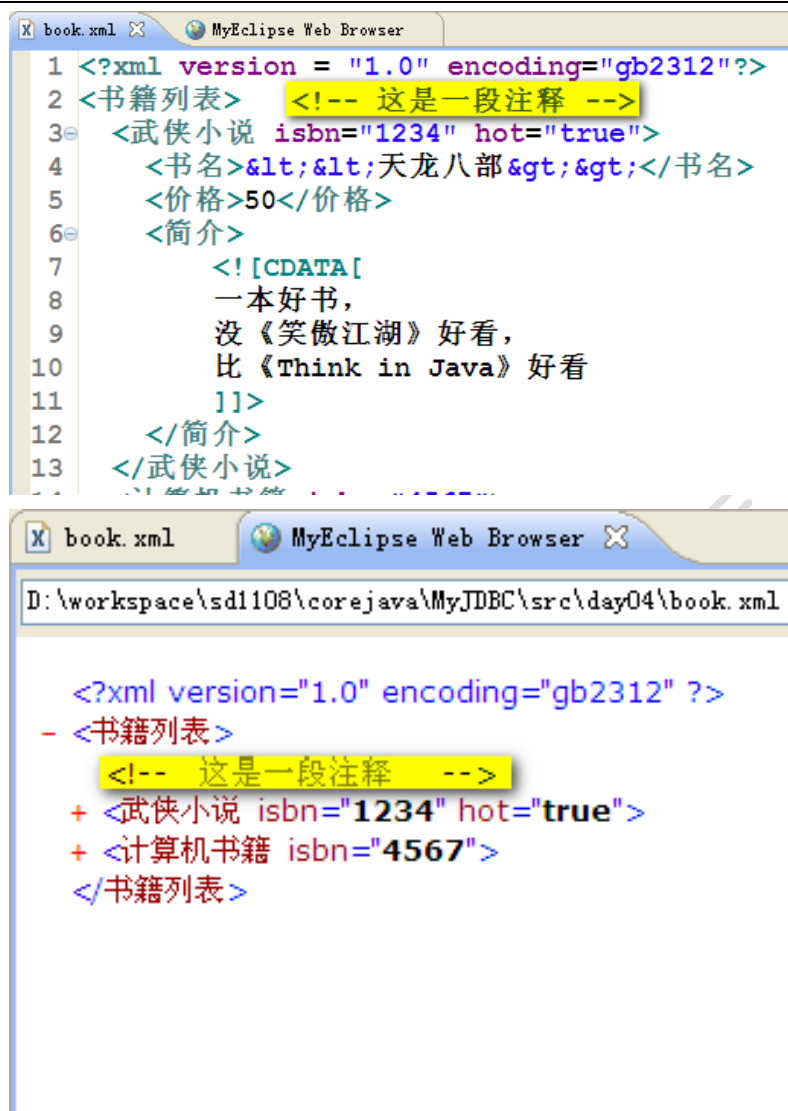
```

<?xml version="1.0" encoding="gb2312" ?>
- <书籍列表>
- <武侠小说 isbn="1234" hot="true">
  <书名><<天龙八部>></书名>
  <简介>
  - <![CDATA[
    一本好书,
    没<<笑傲江湖>>好看
    比<<Think in Java>>好看
  ]]>
  </简介>
</武侠小说>
- <computer_book isbn="5678">
  <书名>Thinking in java</书名>
  <价格>65</价格>
  <简介>Java编程思想</简介>
</computer_book>
</书籍列表>

```

2.7. 注释 (xml 和 html 相同)

- 1) <!-- 这是一段注释 -->
- 2) 编译器将忽略注释
- 3) Html 和 xml 注释方式相同



2.8. 规则小结

- 1) 必须有唯一的根元素
- 2) xml 标记大小写敏感
- 3) 标记必须配对出现，有开始有结束
- 4) 元素必须被正确嵌套
- 5) 属性必须有值，值必须用引号引起来
- 6) 如果遵循所有的规则，称作格式良好的 xml 文件（**well-formed**）

2.9. 使用 XML 文件描述数据的例子

- 1) 早期属性文件描述数据的方式
 url = jdbc:oracle:thin@192.168.0.26:1521:tarena
 dbUser = openlab
 dbPwd = open123
- 2) 现在使用 xml 表示方式

```
<datasource id="db_oracle">
  <property name="url">jdbc:thin@192.168.0.26:1521:tarena</property>
  <property name="dbUser">openlab</property>
  <property name="dbPwd">open123</property>
</datasource>
```

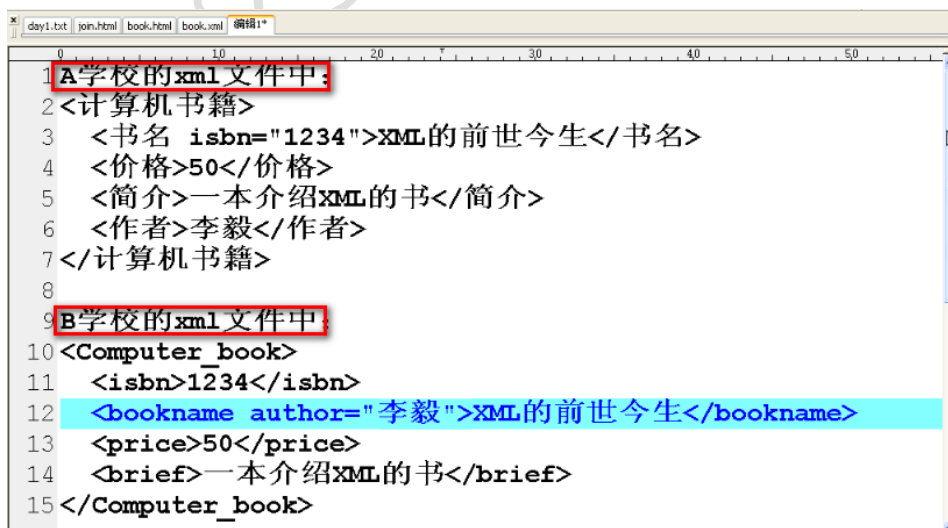
3. DTD/Schema

- 1) DTD / Schema : 用来规范 XML 的标记规则
- 2) 有效的 xml 文件(valid xml file) = 格式良好的 xml 文件 + 有 DTD 或 Schema 规则 + 遵循 DTD 或 Schema 的规则

3.1. DTD/Schema 的由来

行业交换数据时要求 xml 文件格式相同，所以需要大家遵守规范的 xml 文件格式，比如两份 xml 文件要有相同的元素嵌套关系、相同的属性定义，相同的元素顺序，元素出现相同的次数等。

如下为两份相同数据量，但是结构不同的 xml 文件，如图所示：



这两个文件数据相同，但结构不同，无法交换数据。

3.2. 文档类型定义 DTD (Document Type Definition)

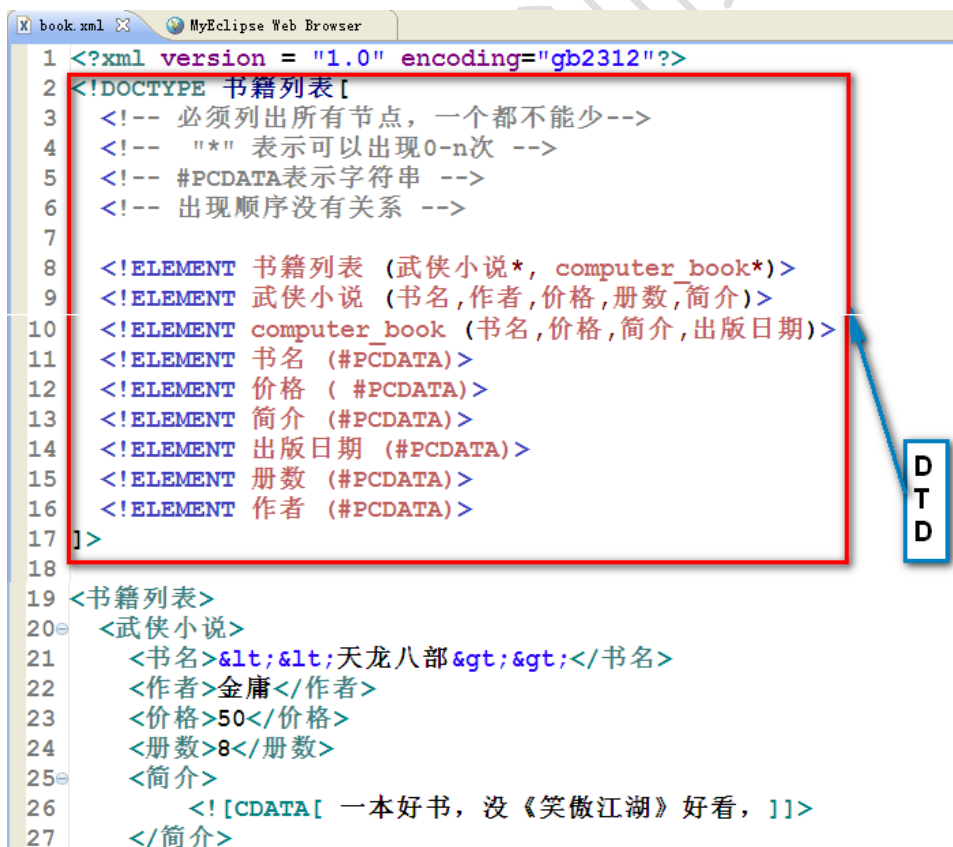
- 1) DTD 文档用来定义 XML 文件的格式，约束 XML 文件中的标记规则
- 2) DTD 类型
 - ✓ PUBLIC (行业共用的)
 - ✓ SYSTEM (小范围自定义的)

3.2.1. DTD 中的定义规则

【案例 3】dtd 规则_元素 (Element)

● 演示 01

在 xml 文件中加入 dtd 规则



```

1 <?xml version = "1.0" encoding="gb2312"?>
2 <!DOCTYPE 书籍列表[
3   <!-- 必须列出所有节点，一个都不能少-->
4   <!-- "*" 表示可以出现0-n次 -->
5   <!-- #PCDATA表示字符串 -->
6   <!-- 出现顺序没有关系 -->
7
8   <!ELEMENT 书籍列表 (武侠小说*, computer_book*)>
9   <!ELEMENT 武侠小说 (书名,作者,价格,册数,简介)>
10  <!ELEMENT computer_book (书名,价格,简介,出版日期)>
11  <!ELEMENT 书名 (#PCDATA)>
12  <!ELEMENT 价格 (#PCDATA)>
13  <!ELEMENT 简介 (#PCDATA)>
14  <!ELEMENT 出版日期 (#PCDATA)>
15  <!ELEMENT 册数 (#PCDATA)>
16  <!ELEMENT 作者 (#PCDATA)>
17 ]>
18
19 <书籍列表>
20 <武侠小说>
21   <书名>&lt;&lt;天龙八部&gt;&gt;</书名>
22   <作者>金庸</作者>
23   <价格>50</价格>
24   <册数>8</册数>
25   <简介>
26     <![CDATA[ 一本好书，没《笑傲江湖》好看，]]>
27   </简介>
    
```

```

28    </武侠小说>
29    <computer_book>
30      <书名>Think in java</书名>
31      <价格>65</价格>
32      <简介>Java编程思想</简介>
33      <出版日期>2000.1.1</出版日期>
34    </computer_book>
35  </书籍列表>

```

● 浏览器显示 (dtd 内容不显示)



● 常见错误

- 1) 书籍列表和" ("间少空格

book.xml MyEclipse Web Browser

```

1 <?xml version = "1.0" encoding="gb2312"?>
2 <!DOCTYPE 书籍列表[
3   <!-- 必须列出所有节点，一个都不能少-->
4   <!-- "*" 表示可以出现0-n次 -->
5   <!-- #PCDATA表示字符串 -->
6   <!-- 出现顺序没有关系 -->
7
8   <!ELEMENT 书籍列表 (武侠小说*, computer_book*)>
9   <!ELEMENT 武侠小说 (书名,作者,价格,册数,简介)>
10  <!ELEMENT computer_book (书名,价格,简介,出版日期)>
11  <!ELEMENT 书名 (#PCDATA)>
12  <!ELEMENT 价格 (#PCDATA)>

```

无法显示 XML 页。

使用 样式表无法查看 XML 输入。请更正错误然后单击 [刷新](#)按钮，或以后重试。

缺少所需的空白区。处理资源
' file:///D:/workspace/sd1108/corejava/MyJDBC/src/day04/book.xml'
时出错。第 8 行，位置：17

<!ELEMENT 书籍列表 (武侠小说*, computer_book*)>
-----^

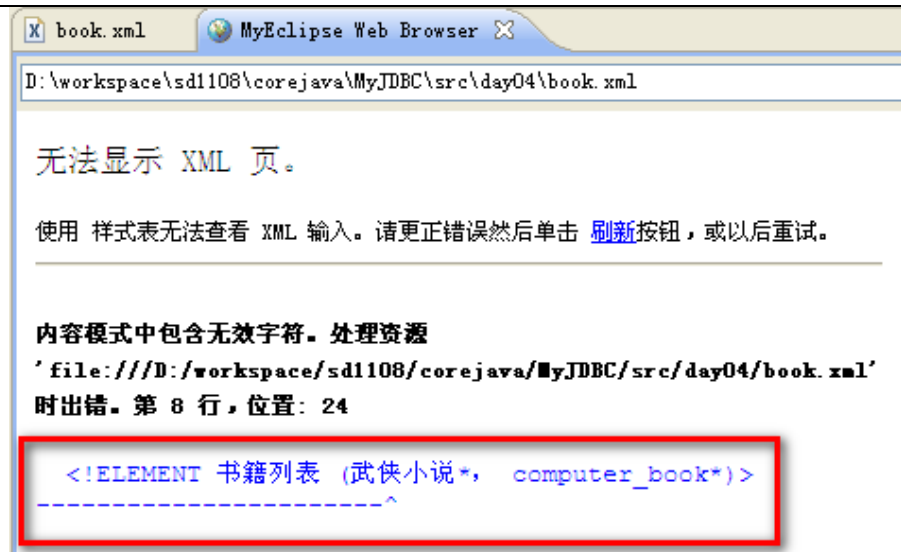
2) 全角的逗号“，”

book.xml MyEclipse Web Browser

```

1 <?xml version = "1.0" encoding="gb2312"?>
2 <!DOCTYPE 书籍列表[
3   <!-- 必须列出所有节点，一个都不能少-->
4   <!-- "*" 表示可以出现0-n次 -->
5   <!-- #PCDATA表示字符串 -->
6   <!-- 出现顺序没有关系 -->
7
8   <!ELEMENT 书籍列表 (武侠小说*, computer_book*)>
9   <!ELEMENT 武侠小说 (书名,作者,价格,册数,简介)>
10  <!ELEMENT computer_book (书名,价格,简介,出版日期)>
11  <!ELEMENT 书名 (#PCDATA)>
12  <!ELEMENT 价格 (#PCDATA)>

```



3) dtd 定义的“作者”和“书名”的顺序和 xml 文件中出现的顺序不一致



● 演示 02

- 1) 增加第 2 本武侠小说 (必须遵循 dtd 中“武侠小说”定义的规范)
- 2) "*" 星号表示该元素可出现 0-n 次

```

8  <!ELEMENT 书籍列表 (武侠小说*, computer_book*)>
9  <!ELEMENT 武侠小说 (书名,作者,价格,册数,简介)>
10 <!ELEMENT computer_book (书名,价格,简介,出版日期)>
11 <!ELEMENT 书名 (#PCDATA)>
12 <!ELEMENT 价格 (#PCDATA)>
13 <!ELEMENT 简介 (#PCDATA)>
14 <!ELEMENT 出版日期 (#PCDATA)>
15 <!ELEMENT 册数 (#PCDATA)>
16 <!ELEMENT 作者 (#PCDATA)>
17 ]>
18
19 <书籍列表>
20 <武侠小说>
21 <书名>&lt;&lt;天龙八部&gt;&gt;</书名>
22 <作者>金庸</作者>
23 <价格>50</价格>
24 <册数>8</册数>
25 <简介>
26 <![CDATA[ 一本好书，没《笑傲江湖》好看，]]>
27 </简介>
28 </武侠小说>
29 <武侠小说>
30 <书名>&lt;&lt;笑傲江湖&gt;&gt;</书名>
31 <作者>金庸</作者>
32 <价格>30</价格>
33 <册数>4</册数>
34 <简介>一本好书</简介>
35 </武侠小说>
36 <computer_book>
42 </书籍列表>

```

● 演示 03

- 1) 至少有 1 个作者，多则不限
- 2) "+"加号表示出现 1-n 次

```

8  <!ELEMENT 书籍列表 (武侠小说*, computer_book*)>
9  <!ELEMENT 武侠小说 (书名,作者+, 价格,册数,简介)>
10 <!ELEMENT computer_book (书名,价格,简介,出版日期)>
11 <!ELEMENT 书名 (#PCDATA)>
12 <!ELEMENT 价格 (#PCDATA)>
13 <!ELEMENT 简介 (#PCDATA)>
14 <!ELEMENT 出版日期 (#PCDATA)>
15 <!ELEMENT 册数 (#PCDATA)>
16 <!ELEMENT 作者 (#PCDATA)>
17 ]>
18
19 <书籍列表>
20 <武侠小说>
21 <书名>&lt;&lt;天龙八部&gt;&gt;</书名>
22 <作者>金庸</作者>
23 <作者>金庸新</作者>
24 <价格>50</价格>

```

● 演示 04

- 1) 固话或者手机，也可以两个都留，还可以多留几个
- 2) "|" 表示或（只能出现一个）

```

1 <?xml version = "1.0" encoding="gb2312"?>
2 <!DOCTYPE 书籍列表[
3
4 <!ELEMENT 书籍列表 (武侠小说*, computer_book*)>
5 <!ELEMENT 武侠小说
6 (书名,作者+, (phone|mobile)*, 价格,册数,简介)>
7 <!ELEMENT computer_book (书名,价格,简介,出版日期)>
8 <!ELEMENT 书名 (#PCDATA)>
9 <!ELEMENT 价格 (#PCDATA)>
10 <!ELEMENT 简介 (#PCDATA)>
11 <!ELEMENT 出版日期 (#PCDATA)>
12 <!ELEMENT 册数 (#PCDATA)>
13 <!ELEMENT 作者 (#PCDATA)>
14 <!ELEMENT phone (#PCDATA)>
15 <!ELEMENT mobile (#PCDATA)>
16 ]>
17
18 <书籍列表>
19 <武侠小说>
20 <书名>&lt;&lt;天龙八部&gt;&gt;</书名>
21 <作者>金庸</作者>
22 <作者>金庸新</作者>
23 <phone>13811111111</phone>
24 <价格>50</价格>

```

● 演示 05

- 1) 简介有没有都可以
- 2) "?" 问号表示出现 0 或 1 次

```

1 <?xml version = "1.0" encoding="gb2312"?>
2 <!DOCTYPE 书籍列表[
3
4 <!ELEMENT 书籍列表 (武侠小说*, computer_book*)>
5 <!ELEMENT 武侠小说
6 (书名,作者+, (phone|mobile)*, 价格,册数,简介)>
7 <!ELEMENT computer_book (书名,价格,简介?, 出版日期)>
8 <!ELEMENT 书名 (#PCDATA)>
9 <!ELEMENT 价格 (#PCDATA)>

```

```

19  <武侠小说>
37  <computer_book>
38    <书名>Think in java</书名>
39    <价格>65</价格>
40    <简介>Java编程思想</简介>
41    <出版日期>2000.1.1</出版日期>
42  </computer_book>
43  <computer_book>
44    <书名>Java核心技术 卷I</书名>
45    <价格>98.00</价格>
46    <出版日期>2001.2.2</出版日期>
47  </computer_book>
48 </书籍列表>

```

● 演示 06

"武侠小说"和"computer_book"可以交叉着添加无数本

```

book.xml  MyEclipse Web Browser
1  <?xml version = "1.0" encoding="gb2312"?>
2  <!DOCTYPE 书籍列表 [
3
4    <!ELEMENT 书籍列表 (武侠小说*, computer_book*) * >
5    <!ELEMENT 武侠小说
6      (书名,作者+, (phone|mobile)*, 价格, 册数, 简介)>
7    <!ELEMENT computer_book (书名, 价格, 简介?, 出版日期)>
8    <!ELEMENT 书名 (#PCDATA)>
9    <!ELEMENT 价格 (#PCDATA)>
10   <!ELEMENT 简介 (#PCDATA)>
11   <!ELEMENT 出版日期 (#PCDATA)>
12   <!ELEMENT 册数 (#PCDATA)>
13   <!ELEMENT 作者 (#PCDATA)>
14   <!ELEMENT phone (#PCDATA)>
15   <!ELEMENT mobile (#PCDATA)>
16 ]>
17
18 <书籍列表>
19 <武侠小说>
20 <书名>&lt;&lt;天龙八部&gt;&gt;</书名>
21 <作者>金庸</作者>
22 <作者>金庸新</作者>
23 <phone>13811111111</phone>
24 <价格>50</价格>

```

```

25     <册数>8</册数>
26     <简介>
27         <![CDATA[ 一本好书, 没《笑傲江湖》好看, ]]>
28     </简介>
29 </武侠小说>
30 <computer_book>
31     <书名>Think in java</书名>
32     <价格>65</价格>
33     <简介>Java编程思想</简介>
34     <出版日期>2000.1.1</出版日期>
35 </computer_book>
36 <武侠小说>
37     <书名>&lt;&lt;笑傲江湖&gt;&gt;</书名>
38     <作者>金庸</作者>
39     <价格>30</价格>
40     <册数>4</册数>
41     <简介>一本好书</简介>
42 </武侠小说>
43 <computer_book>
44     <书名>Java核心技术 卷I</书名>
45     <价格>98.00</价格>
46     <出版日期>2001.2.2</出版日期>
47 </computer_book>
48 </书籍列表>

```

● 知识点小结

1) 元素

出现一次而且有顺序的元素：书名, 作者, 价格, 册数, 简介

- ✓ "*"星号：表示出现 0-n 次的元素
- ✓ "+"加号：表示出现 1-n 次的元素
- ✓ "|"：表示或（只能出现一个）
比如(phone | mobile)表示固话或手机二选一
- ✓ (phone | mobile)*：表示 phone 或 mobile 可出现任意多次
- ✓ "?"问号：表示出现 0 或 1 次

2) 属性 (Attribute)

定义在开始标记中的键值对

【案例 4】dtd 规则_属性 (Attribute)

- 1) <!ATTLIST 标记名称 属性名称 属性类型>
- 2) isbn CDATA #REQUIRED：表示 isbn 属性是必须的
- 3) isbn CDATA #IMPLIED：表示 isbn 属性不是必须的
- 4) hot CDATA "false"：表示 hot 的默认值是 false，
如果属性 hot 定义了值，取此定义好的值，
如果没定义，取值 true

● 演示

```

1 <?xml version = "1.0" encoding="gb2312"?>
2 <!DOCTYPE 书籍列表[
3
4 <!ELEMENT 书籍列表 (武侠小说*, computer_book*)* >
5 <!ELEMENT 武侠小说
6 (书名,作者+, (phone|mobile)*, 价格, 册数, 简介)>
7 <!ELEMENT computer_book (书名, 价格, 简介?, 出版日期)>
8 <!ELEMENT 书名 (#PCDATA)>
9 <!ELEMENT 价格 (#PCDATA)>
10 <!ELEMENT 简介 (#PCDATA)>
11 <!ELEMENT 出版日期 (#PCDATA)>
12 <!ELEMENT 册数 (#PCDATA)>
13 <!ELEMENT 作者 (#PCDATA)>
14 <!ELEMENT phone (#PCDATA)>
15 <!ELEMENT mobile (#PCDATA)>
16 <!--ATTLIST 武侠小说 isbn CDATA #REQUIRED
17 hot CDATA "false"-->
18 <!--ATTLIST computer_book isbn CDATA #REQUIRED
19 ]>
20
21 <书籍列表>
22 <武侠小说 isbn="1234" hot="true">
23 <书名>&lt;&lt;天龙八部&gt;&gt;</书名>
24 <作者>金庸</作者>
25 <作者>金庸新</作者>
26 <phone>13811111111</phone>
27 <价格>50</价格>
28 <册数>8</册数>
29 <简介>
30 <![CDATA[ 一本好书, 没《笑傲江湖》好看, ]]>
31 </简介>
32 </武侠小说>
33 <computer_book isbn="2222">
34 <书名>Think in java</书名>
35 <价格>65</价格>
36 <简介>Java编程思想</简介>
37 <出版日期>2000.1.1</出版日期>
38 </computer_book>
39 <武侠小说 isbn="1111">
40 <书名>&lt;&lt;笑傲江湖&gt;&gt;</书名>
41 <作者>金庸</作者>
42 <价格>30</价格>

```

【案例 5】把 xml 文件和 DTD 分离

- 步骤 1: 新建 book.dtd, 把 DTD 部分保存在此文件中

```

1 <?xml version="1.0" encoding="gb2312"?>
2 <!--ELEMENT 书籍列表
3      (武侠小说*, computer_book*)>
4 <!--ELEMENT 武侠小说
5      (书名, 作者+, (phone|mobile)*, 价格, 册数, 简介)>
6 <!--ELEMENT computer_book
7      (书名, 价格, 简介?, 出版日期)>
8 <!--ELEMENT 书名 (#PCDATA)>
9 <!--ELEMENT 价格 (#PCDATA)>
10 <!--ELEMENT 简介 (#PCDATA)>
11 <!--ELEMENT 出版日期 (#PCDATA)>
12 <!--ELEMENT 册数 (#PCDATA)>
13 <!--ELEMENT 作者 (#PCDATA)>
14 <!--ELEMENT phone (#PCDATA)>
15 <!--ELEMENT mobile (#PCDATA)>
16 <!--ATTLIST 武侠小说 isbn CDATA #REQUIRED
17      hot CDATA "false">
18 <!--ATTLIST computer_book isbn CDATA #REQUIRED
19

```

第一行设置字符集是因为文档中有中文。

● 步骤 2 : xml 文件改为指定 DTD 文件为 book.dtd

```

1 <?xml version = "1.0" encoding="gb2312"?>
2 <!DOCTYPE 书籍列表 SYSTEM "book.dtd">
3
4 <书籍列表>
5   <武侠小说 isbn="1234" hot="true">
6     <书名>&lt;&lt;天龙八部&gt;&gt;</书名>
7     <作者>金庸</作者>
8     <作者>金庸新</作者>
9     <phone>13811111111</phone>
10    <价格>50</价格>
11    <册数>8</册数>
12    <简介>

```

注意：DTD 类型

- ✓ PUBLIC (行业共用的)
- ✓ SYSTEM (小范围自定义的)

3.3. Schema

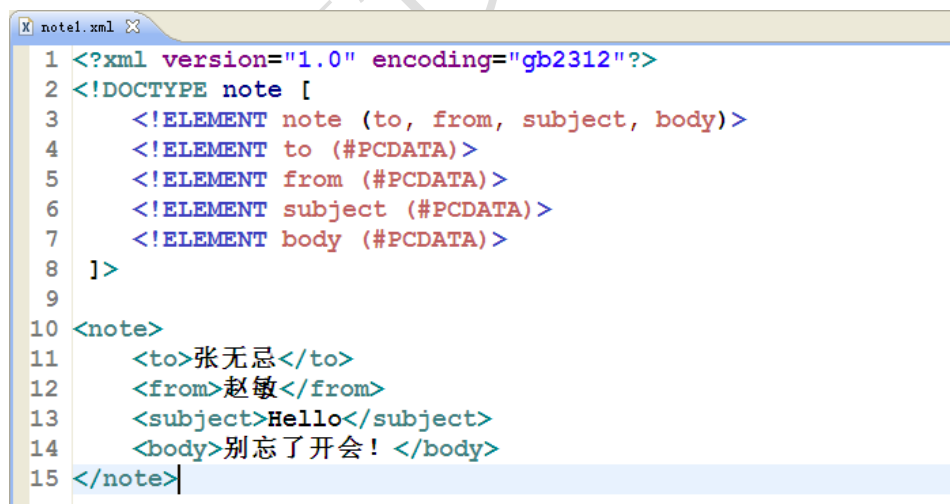
- 1) **命名空间 (Namespace)** XML 文件允许自定义标记，所以可能出现来自不同源 DTD 或 Schema 文件的同名标记，为了区分这些标记，就需要使用命名空间。
 - ✓ 命名空间的目的是有效的区分来自不同 DTD 的相同标记

- ✓ 比如如下 xml 文件中使用了命名空间区分开“表格”和“桌子”：


```
<html:table>
    <line><column>这是一个表格</column></line>
</html:table>
<product:table>
  <type>coffee table</type>
  <meterial>wood</meterial>
</product:table>
```
- 2) 因为 DTD 无法解决命名冲突，所以出现 Schema，它是 DTD 的替代者
dtd 和 schema 的功能都是用于描述 XML 结构的
- 3) Schema 支持命名空间，使用 xml 语法实现（Schema 本身就是 xml 文件）
因为用于规范和描述 xml 文件的定义文件（schema）本身也是 xml 文件，所以 xml 也被称作是**自描述的语言**
- 4) Schema 文件的扩展名 xsd：XML Schema Difinition（简称 **XSD**，遵循 W3C 标准）
- 5) Schema 中的名词：
 - ✓ 复杂元素(有子元素的元素)
 - ✓ 简单元素(叶子元素)

【案例 6】Schema 约束 xml 演示

● note.xml 中使用 dtd 约束_演示



```
1 <?xml version="1.0" encoding="gb2312"?>
2 <!DOCTYPE note [
3   <!ELEMENT note (to, from, subject, body)>
4   <!ELEMENT to (#PCDATA)>
5   <!ELEMENT from (#PCDATA)>
6   <!ELEMENT subject (#PCDATA)>
7   <!ELEMENT body (#PCDATA)>
8 ]>
9
10 <note>
11   <to>张无忌</to>
12   <from>赵敏</from>
13   <subject>Hello</subject>
14   <body>别忘了开会! </body>
15 </note>
```

● note.xml 使用 schema 约束

1) note.xml

```

1 <?xml version="1.0" encoding="gb2312"?>
2 <note xmlns="http://www.tarena.com.cn"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://www.tarena.com.cn note.xsd"
5 >
6   <to>张无忌</to>
7   <from>赵敏</from>
8   <subject>Hello</subject>
9   <body>别忘了开会! </body>
10 </note>
11

```

- ✓ **注意：**将例中黄色高亮标出部分更改为你的 xsd 所在位置（本例中 note.xsd 和 note.xml 在同一目录下）

2) note.xsd

```

1 <?xml version="1.0"?>
2 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
3   targetNamespace="http://www.tarena.com.cn"
4   xmlns="http://www.tarena.com.cn"
5 >
6
7   <xs:element name="note">
8     <xs:complexType>
9       <xs:sequence>
10        <xs:element name="to" type="xs:string" />
11        <xs:element name="from" type="xs:string" />
12        <xs:element name="subject" type="xs:string" />
13        <xs:element name="body" type="xs:string" />
14      </xs:sequence>
15    </xs:complexType>
16  </xs:element>
17
18 </xs:schema>

```

学习建议：

一般情况下，程序员的工作是根据已有的 xsd 或 dtd 文件规则编写 xml 文件，故现在不必过分关注 dtd 或 xsd 文件细节。

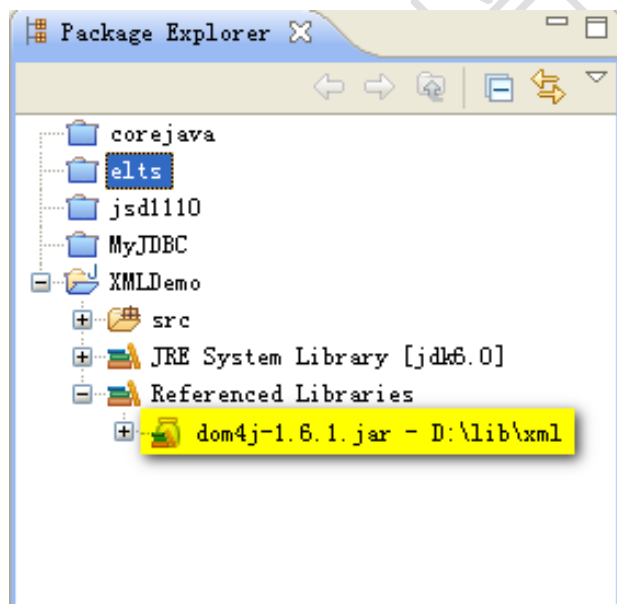
4. java API 解析 XML 文件

- 1) Java 与 xml 有很多共同点（比如跨平台、与厂商无关），目前为止 java 对 xml 的解析较其他语言更完善

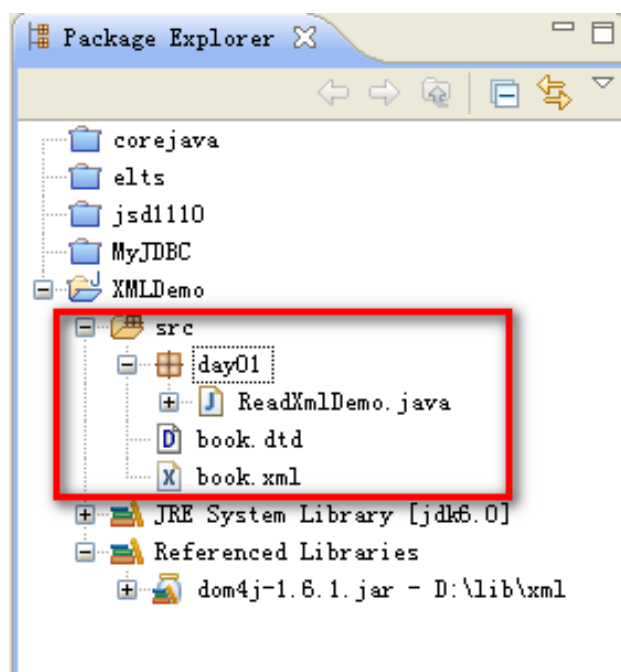
- 2) 两种解析方式：
 - ✓ **DOM** (Document Object Model 文档对象模型)
 - 关键字：树(Document)
 - 优点： 把 xml 文件在内存中构造树形结构，可以遍历和修改节点
 - 缺点： 如果文件比较大，内存有压力，解析的时间会比较长
 - ✓ **SAX** (Simple API for Xml 基于 XML 的简单 API)
 - 关键字：流(Stream)
 - 把 xml 文件作为输入流，触发标记开始，内容开始，标记结束等动作
 - 优点： 解析可以立即开始，速度快，没有内存压力
 - 缺点： 不能对节点做修改
- 3) **JDOM / DOM4J** ：目前市场上常用的 2 种解析 XML 文件的 API
 - ✓ dom4j-1.6.1.jar 结合了 DOM 和 SAX 两种解析方式的优点

【案例 7】DOM4j 解析 xml 文件演示

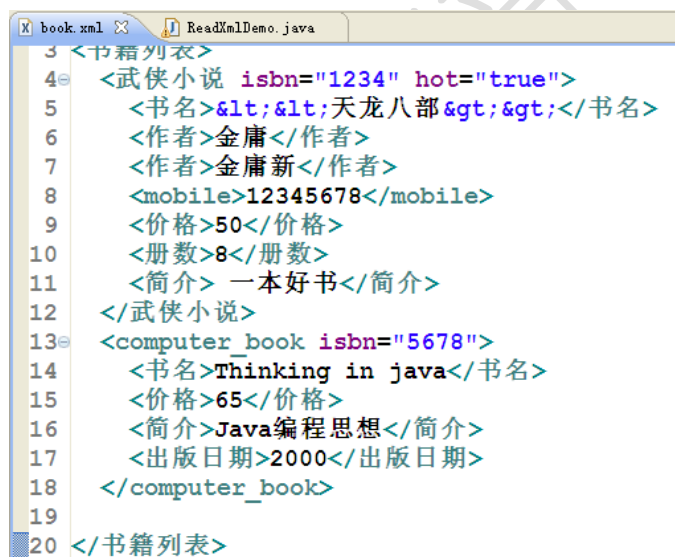
- 步骤 1：
 - 1) 创建项目 XMLDemo
 - 2) 加入 dom4j 的 jar 包 (dom4j-1.6.1.jar)



- 步骤 2：
 - 1) 准备好要解析的文件 book.xml，放入路径 src/下
 - 2) 新建 ReadXMLDemo.java



book.xml



book.dtd

```

1 <?xml version="1.0" encoding="gb2312"?>
2 <!--ELEMENT 书籍列表 (武侠小说*, computer_book*)-->
3 <!--ELEMENT 武侠小说
4 (书名, 作者+, (phone|mobile)*, 价格, 册数, 简介)>
5 <!--ELEMENT computer_book (书名, 价格, 简介?, 出版日期)>
6 <!--ELEMENT 书名 (#PCDATA)>
7 <!--ELEMENT 价格 (#PCDATA)>
8 <!--ELEMENT 简介 (#PCDATA)>
9 <!--ELEMENT 出版日期 (#PCDATA)>
10 <!--ELEMENT 册数 (#PCDATA)>
11 <!--ELEMENT 作者 (#PCDATA)>
12 <!--ELEMENT phone (#PCDATA)>
13 <!--ELEMENT mobile (#PCDATA)>
14 <!--ATTLIST 武侠小说 isbn CDATA #REQUIRED
15 hot CDATA "false">
16 <!--ATTLIST computer_book isbn CDATA #REQUIRED

```

● 步骤 3 : ReadXmlDemo.java

读取 book.xml 中的元素，打印到控制台

```

1 package day01;
2 import org.dom4j.*;
3 import org.dom4j.io.*;
4 import java.io.*;
5 import java.util.*;
6 /**
7  * 利用DOM4J解析XML文件
8  * @author teacher
9  */
10 public class ReadXmlDemo {
11     public static void main(String[] args) {
12         readBook("src/book.xml");
13     }
14
15     /**
16      * 读入指定的xml文件，分析元素，打印数据
17      * @param filename 指定的xml文件名
18      */
19     public static void readBook(String filename) {
20         // 解析器
21         SAXReader reader = new SAXReader();
22         // 指定xml文件
23         File file = new File(filename);
24     }

```

```

25     try {
26         // 开始解析，构建树形结构并返回
27         Document doc = reader.read(file);
28         // 获得根元素：书籍列表
29         Element rootElmt = doc.getRootElement();
30
31         // 获得所有武侠小说的元素集合
32         List list = rootElmt.elements("武侠小说");
33         parseNovel(list);
34
35         // 获得所有computer_book的元素集合
36         List list1 = rootElmt.elements("computer_book");
37         //parseComputerBook(list1);
38
39     } catch (DocumentException e) {
40         e.printStackTrace();
41     }
42 }
43
44 /**
45  * 解析所有武侠小说的元素集合
46  * @param list 武侠小说的元素集合
47  */
48 private static void parseNovel(List list) {
49     Iterator it = list.iterator();
50     while(it.hasNext()){
51
52         //武侠小说的元素
53         Element novelElmt = (Element)it.next();
54
55         System.out.println("书名: " +
56             novelElmt.elementText("书名"));
57         System.out.println("作者: " +
58             novelElmt.elementText("作者"));
59         System.out.println("价格: " +
60             novelElmt.elementText("价格"));
61         System.out.println("册数: " +
62             novelElmt.elementText("册数"));
63         System.out.println("简介: " +
64             novelElmt.elementText("简介"));
65     }
66 }
67 }

```

Console SQL Results Properties
<terminated> ReadXmlDemo [Java Application] C:\Program Files\Java\jdk1.6.0_06\bin\javaw.exe (Dec 22, 2011 2:07:03 PM)

书名: <<天龙八部>>

作者: 金庸

价格: 50

册数: 8

简介:

一本好书,没有<<笑傲江湖>>好看,
比<<Java编程思想>>好看。

书名: 笑傲江湖

作者: 金庸

价格: 55

册数: 4

简介: 一本好书

注1:

```

<?xml version="1.0" encoding="gb2312"?>
<书籍列表>
  <武侠小说 isbn="1234" hot="true">
    <书名>&lt;&lt;天龙八部&gt;&gt;</书名>
    <简介>一本好书</简介>
  </武侠小说>
  <武侠小说 isbn="1235" hot="true">
    <书名>&lt;&lt;笑傲江湖&gt;&gt;</书名>
    <简介>还是一本好书</简介>
  </武侠小说>
  <computer_book isbn="5678">
    <书名>Thinking in java</书名>
    <价格>65</价格>
    <简介>Java编程思想</简介>
  </computer_book>
  <computer_book isbn="5679">
    <书名>Thinking in C++</书名>
    <价格>65</价格>
    <简介>C++编程思想</简介>
  </computer_book>
</书籍列表>
  
```

```

// 获得根元素: 书籍列表
Element rootElmt = doc.getRootElement();

// 获得所有武侠小说的元素集合
List list = rootElmt.elements("武侠小说");
parseNovel(list);

// 获得所有computer_book的元素集合
List list1 = rootElmt.elements("computer_book");
//parseComputerBook(list1);
  
```

注2:

```

k.xml x ReadXmlDemo.java
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE 书籍列表 [
<书籍列表>
  <武侠小说 isbn="123456">
    <书名>&lt;&lt;&lt;
    <作者>金庸</作者>
    <作者>金庸新</作者>
    <mobile>123456</mobile>
    <价格>50</价格>
    <册数>8</册数>
    <简介>一本好书</简介>
  </武侠小说>
  <computer_book>
    <书名>Thinking</书名>
    <价格>65</价格>
    <简介>Java编程</简介>
    <出版日期>2000</出版日期>
  </computer_book>
</书籍列表>

```

//武侠小说的元素

```
Element novelElmt = (Element)it.next();
```

```

System.out.println("书名: " +
    novelElmt.elementText("书名"));
System.out.println("作者: " +
    novelElmt.elementText("作者"));
System.out.println("价格: " +
    novelElmt.elementText("价格"));
System.out.println("册数: " +
    novelElmt.elementText("册数"));
System.out.println("简介: " +
    novelElmt.elementText("简介"));

```

元素(Element)

● 步骤 4 : ReadXmlDemo.java

读取 book.xml 中元素的属性，打印到控制台

```

book.dtd ReadXmlDemo.java x
44 // **
45 * 解析所有武侠小说的元素集合
46 * @param list 武侠小说的元素集合
47 */
48 private static void parseNovel(List list) {
49     Iterator it = list.iterator();
50     while(it.hasNext()){
51
52         //武侠小说的元素
53         Element novelElmt = (Element)it.next();
54
55         //1. 取出所有元素
56         System.out.println("书名: " +
57             novelElmt.elementText("书名"));
58         System.out.println("作者: " +
59             novelElmt.elementText("作者"));
60         System.out.println("价格: " +
61             novelElmt.elementText("价格"));
62         System.out.println("册数: " +
63             novelElmt.elementText("册数"));
64         System.out.println("简介: " +
65             novelElmt.elementText("简介"));
66
67         System.out.println("#####");

```



```

68
69 //2. 取出元素的所有属性
70 //获取武侠小说元素的所有属性，并返回集合的迭代器
71 //2.1 方式1
72 List attrList = novelElmt.attributes();
73 Iterator attrIt = attrList.iterator();
74 while(attrIt.hasNext()){
75     Attribute attr = (Attribute)attrIt.next();
76     System.out.println(
77         attr.getName() + "=" + attr.getValue());
78 }
79
80 //2.2 方式2
81 Iterator atts = novelElmt.attributeIterator();
82 // 遍历所有属性
83 while(atts.hasNext()){
84     //打印属性名和属性值
85     Attribute att = (Attribute)atts.next();
86     System.out.println(
87         att.getName() + "=" + att.getValue());
88 }
89 }
90 }
91 }

```