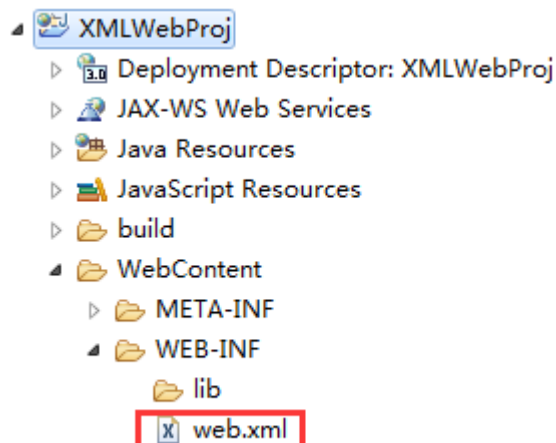


# XML详解

# 什么是XML



```
web.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3 xmlns="http://java.sun.com/xml/ns/javaee"
4 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
5 id="WebApp_ID"
6 version="3.0">
7   <display-name>XMLWebProj</display-name>
8   <welcome-file-list>
9     <welcome-file>index.html</welcome-file>
10    <welcome-file>index.htm</welcome-file>
11    <welcome-file>index.jsp</welcome-file>
12    <welcome-file>default.html</welcome-file>
13    <welcome-file>default.htm</welcome-file>
14    <welcome-file>default.jsp</welcome-file>
15   </welcome-file-list>
16 </web-app>
```

# 主要内容

- XML概述
- XML文档规则
- 命名空间
- Xpath语言详解
- 操作XML文档的几种方式介绍
- SAX操作XML文档

# XML概述

- XML的基本概念

- XML(Extensible Markup Language,可扩展标记语言)
- 允许开发者自由定义标签，可以将标签和内容有效分离
- XML不再侧重于数据如何展示，而是更多地关注数据如何存储和传输

# XML概述

- XML的应用场景

- 1.XML把数据从HTML分离出来

- 2.简化数据共享

- 3.简化数据的传输

- 4.简化平台的变更

# XML概述

- XML的优势

- 1.简单易用的标记语言

- 2.严格的格式

- 3.数据逻辑和显示逻辑分离

# XML概述

- 编写第一个XML文件：存储图书信息

书名	作者	价格
Java思想	小王	79.00
Spring指南	小李	89.00

# XML文档规则

- XML整体结构

- 1.有且只有一个根元素

- 2.元素必须合理结束

- 3.元素之间必须合理嵌套

- 4.元素的属性必须有值



# XML文档规则

- XML声明

- `<?xml version= "1.0" encoding= "GB2312" standalone= "yes" ?>`

- 字符集

1. 简体中文:GBK或GB2312

2. 繁体中文:BIG5

3. 西欧字符:ISO8859-1

4. 通用的国际编码:Unicode

5. 针对Unicode的可变长度字符编码:UTF8

# XML文档规则

- XML元素的基本规则

- 1. 合法标签名

- XML元素由开始标签和结束标签组成，结束标签比开始标签多一条斜线
    - XML文档区分大小写。因此，开始标签和结束标签名必须绝对相同，大小写也要完全一致

# XML文档规则

- XML元素的基本规则

- 1.合法标签名

XML文档对于标签名有如下要求:

- 标签名可以字母（包括非西欧字符）、数字、下画线（\_）、中画线（—）、冒号（:）和点号（.）组成，但不能以数字、中画线和点号开头
- 标签名不能包括<、>、,、\$等符号
- 标签名中尽量不要出现英文冒号“:”，除非是在使用名字空间
- 标签名不能以字符“xml”（或者XML、Xml等任意大小写组合开始）
- 标签名不能包含空格

# XML文档规则

- XML元素的基本规则

- 2.嵌套子元素

- XML允许无限深度嵌套子元素，只要保证元素之间合理嵌套即可
    - XML元素可以嵌套多个重名的子元素，这多个元素之间是有序的

# XML文档规则

- XML元素的基本规则

## 3.空元素

- XML允许使用空元素语法，空元素不可接受子元素，也不可接受字符串内容
- 空元素和内容空的元素并不相同
- 空元素只是不能包含子元素，也不能包含字符串内容，但完全可以接受属性，而且可以接受任意多个属性

# XML文档规则

- **字符数据**

**开始标签和结束标签之间的文本可以是任何Unicode字符，并且其间的任何字符都将忠实地传递给XML处理程序**

**如果文本字符串中包含一些特殊的字符，例如尖括号（<）或and符号（&），由于这些符号在XML文档中都有特殊的含义，因此直接在XML元素中使用该字符串将引起文档混乱**

# XML文档规则

- 字符数据

- 1.使用实体引用

为了正确处理XML文档中的特殊字符，XML允许使用实体来表示这些特殊字符。

XML预置了5个实体引用，如表：

实体引用	所工表符号	说明
&lt;	<	小于符号
&gt;	>	大于符号
&amp;	&	and符号
&apos;	'	英文单引号
&quot;	“	英文双引号

# XML文档规则

- 字符数据

- 2.使用CDATA标记

在特殊标记CDATA下，所有的特殊字符，甚至是有效的元素都将被当成简单字符处理

实体引用也会失去作用，变成纯文本

语法:

<![CDATA[文本内容]]>



# XML文档规则

- **注释**

**XML文档还可加入解释用的字符数据，这些解释用的字符串不会被XML解析器处理.这些解释用的文本称为注释**

**语法:**

`<!--注释字符串-->`

# XML文档规则

- **注释**

**XML注释需要注意的地方：**

- 1. 不要把注释放在标签之内，否则，该文档将不是一个格式良好的XML文档**
- 2. 不要把注释放在XML声明之前，XML声明应该永远处于XML文档的第一行**
- 3. 不要在注释中使用双中画线（--）**

# XML文档规则

- **W3C对于属性的使用建议**

- 属性通常提供属于数据组成部分的信息，如果属性值里包含的信息属于该实体本身，则应该使用子元素来指定该信息，因此，W3C推荐尽量使用子元素，而避免使用属性

# XML文档规则

- 换行处理

目前主流的操作系统，主要有3种换行符：

1.Windows平台:回车符（CR）和换行符（LF）的组合存储换行

2.UNIX和Linux平台：以换行符（LF）存储换行

3.Macintosh平台：以回车符（CR）存储换行

XML统一换行符（LF）存储换行

# XML文档规则

- XML文档分类

1. 格式不良的XML文档

- 完全没有遵守XML文档基本规则的XML文档

# XML文档规则

- XML文档分类

- 2. 格式良好但无效的XML文档

- 遵守了XML文档基本规则，但没有使用DTD或Schema定义语义约束的XML文档
    - 使用DTD或Schema定义了语义约束，但没有遵守DTD或Schema所定义的语义约束的XML文档

# XML文档规则

- XML文档分类

- 3. 有效的XML文档

- 遵守了XML文档基本规则，并使用DTD或Schema定义了语义约束，而且也完全遵守了DTD或Schema所定义的语义约束的XML文档

# XML命名空间

- **为什么使用命名空间**

- 在同一份XML文档中可能出现多个同名的元素和属性——这多个同名的元素和属性具有不同的含义和作用，但如果我们不从语法上提供区别，则XML处理器无法区分它们



# XML命名空间

- 使用命名空间

语法：

`xmlns[:prefix]= “命名空间字符串”`

命名空间的特征：

- 名字很长（命名空间往往是一个绝对的URL地址）
- 名字里往往包含英文冒号、斜线等特殊字符

# XML命名空间

- 属性使用命名空间

-通常情况下，由于属性是属于某个元素的，因此很自然地认为属性总是属于它所在元素所处的命名空间，一般无须专门为属性指定命名空间

# XPath语言

- XPath概念

- XPath语言是一门专门用于在XML文档中查找信息的语言，其他XML程序可利用XPath在XML文档中对元素和属性进行导航

# XPath语言

- XPath节点

节点类型	说明
XML文档根节点	XML文档的根称为文档节点或根节点
元素节点	一个元素的开始标签、结束标签，以及开始标签和结束标签之间的全部内容整体称为元素节点
属性节点	元素的每个属性都是属性节点。属性节点包括属性名和属性值两个部分。XPath认为属性节点必须依附于元素节点
XML节点	XML文档里<!--和-->包含的部分就是注释，注释对应的就是注释节点
命名空间节点	命名空间节点代表XML文档中的xmlns:prefix属性
文本节点	即XML元素中间的字符数据，包括CDATA段中的字符数据

# XPath语言

- XPath基本概念

- 基本值（或称原子值）

- 基本值专门用于表示简单的数据值，例如整数值、字符串等。我们可以基本值当成没有父节点且没有子节点的节点

- 项

- 项是XPath2.0提出的一个术语，一个项代表一个节点或基本值

# XPath语言

- XPath基本概念

- 节点集和序列

- 在某些情况下，XPath表达式可以表示多个节点，多个节点组合在一起在XPath1.0里称为节点集

- XPath2.0提出一个序列的概念，XPath2.0的序列可以代表一个普通的项，也可以代表节点集

# XPath语言

- **节点关系**

- **父节点**
- **子节点**
- **兄弟节点**
- **祖先节点**
- **后代节点**

# XPath语言

- 相对路径和绝对路径

XPath同样支持相对路径和绝对路径。对于XPath而言，绝对路径以斜线（/）开头，而相对路径则不会以斜线（/）开的

例如：

**/list/book/name:**该路径总是匹配list元素内的book元素之内的name子元素

**list/book/name:**该路径到底匹配哪个或哪些节点是不确定的



# XPath语言

- XPath基础语法

- XPath使用路径表达式来定位XML文档中的节点或节点集，每个Xpath表达式总由多个步（step）组成，多个步之间用斜线分隔

例如:

`/list/book/name`

- XPath中步的完整语法格式如下:

`轴::节点测试[限定谓语]`

# XPath语言

- XPath基础语法

- 轴

**XPath的步骤使用轴来定义所选节点与当前节点之间的结构关系**

轴名称	含义
ancestor	祖先节点
ancestor-or-self	祖先节点并包含自身
attribute	选择节点的所有属性
child	选取当前节点的所有子节点
parent	选取当前节点的父节点
descendant	当前节点的所有后代节点
descendant-or-self	当前节点的所有后代节点并包含自身
self	发前了点自身
following-sibling	选择当前节点的兄弟节点

# XPath语言

- XPath基础语法

- 节点测试

节点测试用于从指定轴所匹配的节点集中选出特定的节点

轴名称	含义	实例
nodename	从指南轴匹配的所有节点中选出具有nodename的节点	child::book
node()	选择指定轴匹配的所有类型节点	child::node()
text()	选择指定轴匹配的所有文本在型节点	child::text()
comment()	选择指定轴匹配的民有注释节点	descendant:commet()
*	通配符，不进地任何过滤	child::*

# XPath语言

- XPath基础语法

- 限定谓语句

限定谓语句是一个boolean表达式，或者可以转换为boolean值的表达式，用于进一步提炼所选的节点集。限定谓语句应该放在括号中

语法:

`child::book[1]`或`child::book[position()=1]`

# XPath语言

- XPath基础语法

- 简化写法

- 省略child轴，例如:website/muke等同于child::website/child::muke
    - 使用@符号代替attribute轴,例如:book[@isbn= "123456" ]等同于child::book[attribute::isbn= "123456" ]
    - 使用//代表后代表节点，例如://book等同于/descendant-or-self::node()/child::book
    - 使用一个点代表当前节点，例如:./book等同于self::node()/child::book
    - 使用两个点代表上一级节点,例如:../book等同于parent::node()/child::book

# XPath语言

- XPath运算符

- 算术运算符

运算符	含义
+	加
-	减
*	乘
div	除
mod	求余

# XPath语言

- XPath运算符

- 比较运算符

运算符	含义
=	等于
!=	不等于
<	小于
<=	小于或等于
>	大于
>=	大于或等于

# XPath语言

- XPath运算符

- 逻辑运算符

运算符	含义
or	或
and	与



# XPath语言

- XPath运算符

- 组合多个路径的运算符

XPath还提供了一个“|”运算符，可用于组合多个路径表达式,通过“|”运算符，可以一次选取若干个路径

例如:book[position()=1] | name[position()=last()]

# XPath语言

- 节点相关的常见函数

名称	说明
fn:position()	返回当前正在被处理的节点在父节点中的index值
fn:last()	返回当前正在被处理的节点列表中项的数目
fn:name()	返回当前节点的名称或指定节点集中第一个节点的名称
fn:root()	通常返回文档根节点
fn:node-name(node)	返回node节点名称

# DOM、SAX和JAXP解析

- XML文档解析方式

- DOM : Document Object Model:即文档对象模型，它是由W3C推荐的  
处理XML文档的规范
- SAX : Simple API for XML,它并不是W3C推荐的标准，但却是整个XML  
行业的事实规范

# DOM、SAX和JAXP解析

- XML文档解析原理



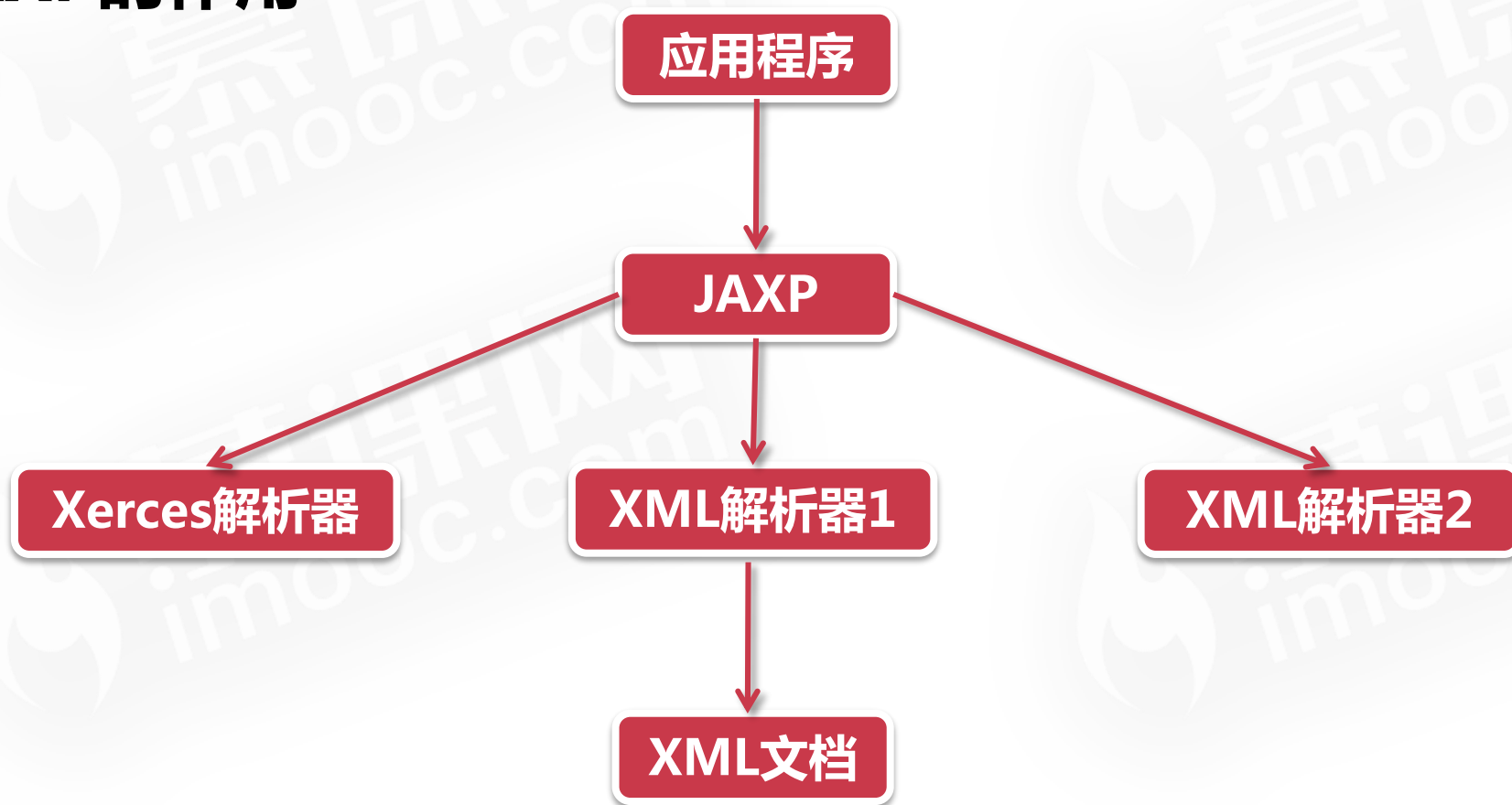
# DOM、SAX和JAXP解析

- **JAXP概述**

- Java解析XML文档的API称为JAXP，它的全称是Java API for XML。
- JAXP往往作为JDK的一部分发布，但它仅仅是一些API接口，并未提供真正的实现，因此实际使用过程中还需要具体的解析实现。
- JAXP只是允许应用程序在不同的XML解析器之间切换

# DOM、SAX和JAXP解析

- JAXP的作用



# DOM、SAX和JAXP解析

- **JAXP提供的与解析相关的类**

- **DocumentBuilderFactory** : 获取DOM解析工厂类
- **DocumentBuilder**:DOM解析器标准接口
- **SAXParserFactory** : 获取SAX解析器的工厂类
- **SAXParser**:SAX解析器的标准接口

# DOM、SAX和JAXP解析

- **JAXP的SAX支持**

- **SAX的处理机制**

- **SAX采用事件机制的方式来解析XML文档，这是一种快速读写XML数据的方式。**

- **使用SAX解析器对XML文档进行解析时，会触发一系列事件，这些事件将被相应的事件监听器监听，从而触发相应的事件处理方法，应用程序通过这些事件处理方法实现对XML文档的访问**



# DOM、SAX和JAXP解析

- JAXP的SAX支持

- SAX解析XML文档的流程



# DOM、SAX和JAXP解析

- JAXP的SAX支持

- SAX解析XML文档的流程

-使用SAX机制解析XML文档时，SAX解析器负责在XML文档中“行走”，每当遇到文档开始、元素开始、文本、元素结束和文档结束时，都将负责向外发送事件，而程序员则负责提供事件监听器来监听这些事件，并通过事件获取XML文档信息



# DOM、SAX和JAXP解析

- **JAXP的SAX支持**

- **JAXP为SAX解析器提供2组API :**
  - **XMLReader和XMLReaderFactory:XMLReaderFactory工厂类的createXMLReader()静态方法用于创建XMLReader**
  - **SAXParserSAXParserFactory : SAXParserFactory工厂类的新SAXParser()实例方法用于创建SAXParser**

# DOM、SAX和JAXP解析

- JAXP的SAX支持

- XMLReader解析XML文档的方法:

- void parse(InputSource input):解析InputSource输入源中的XML文档

- void parse(String systemId):解析系统URI（磁盘文件是URI的一种）所代表的XML文档

# DOM、SAX和JAXP解析

- JAXP的SAX支持

- SAXParser解析XML文档的方法:

- void parse(File f,DefaultHandler dh):使用指定的dh作为监听器监听SAX解析事件，解析f文件所代表的XML文档
    - void parse(InputSource is,DefaultHandler dh):使用指定的dh作为监听器监听SAX解析事件，解析is输入源中的XML文档
    - void parse(InputStream is,DefaultHandler dh):使用指定的dh作为监听器监听SAX事件，解析is输入流中的XML文档
    - void parse(String uri,DefaultHandler dh):使用指定的dh作为监听器监听SAX事件，解析URI所代表的XML文档

# DOM、SAX和JAXP解析

- **JAXP的SAX支持**

- **SAX解析事件**

- **ContentHandler**:监听XML文档内容处理事件的监听器

- **DTDHandler**:监听DTD处理事件的监听器

- **EntityResolver**:监听实体处理事件的监听器

- **ErrorHandler**:监听解析错误的监听器

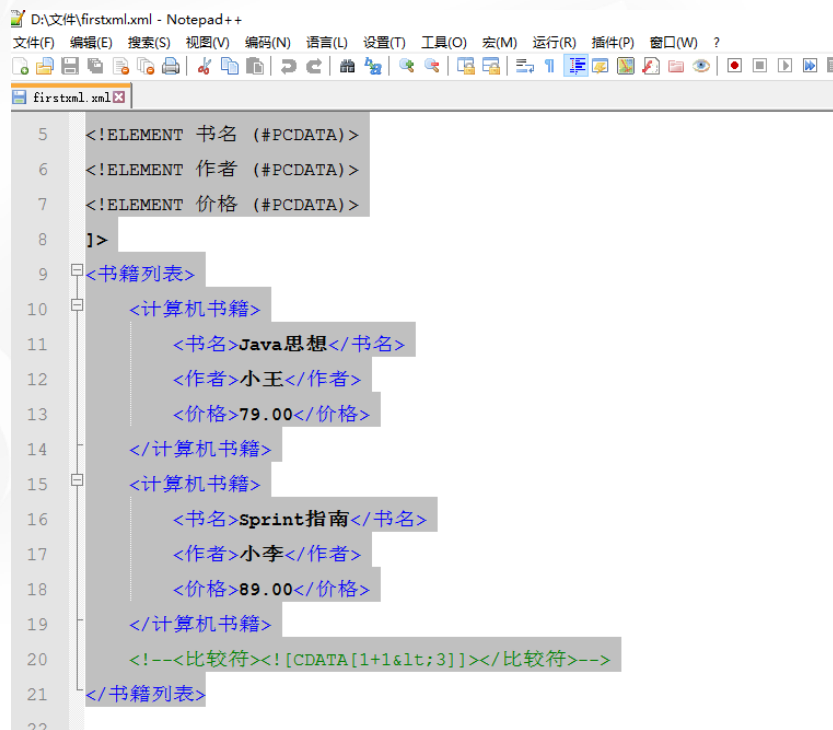
- **JAXP提供了一个DefaultHandler类，这个类实现了上述4个监听器接口，并为监听器接口中所包含的方法提供了空实现**

# DOM、SAX和JAXP解析

- JAXP的SAX支持

- SAX解析XML文档

- 需求：解析出firstxml.xml的内容



```
DA文件\firstxml.xml - Notepad++
文件(F)  编辑(E)  搜索(S)  视图(V)  编码(N)  语言(L)  设置(T)  工具(O)  宏(M)  运行(R)  插件(P)  窗口(W)  ?
firstxml.xml
5  <!ELEMENT 书名 (#PCDATA)>
6  <!ELEMENT 作者 (#PCDATA)>
7  <!ELEMENT 价格 (#PCDATA)>
8  ]>
9  <书籍列表>
10 <计算机书籍>
11   <书名>Java思想</书名>
12   <作者>小王</作者>
13   <价格>79.00</价格>
14 </计算机书籍>
15 <计算机书籍>
16   <书名>Sprint指南</书名>
17   <作者>小李</作者>
18   <价格>89.00</价格>
19 </计算机书籍>
20 <!--<比较符><![CDATA[1+1<3]]></比较符-->
21 </书籍列表>
22
```

# DOM、SAX和JAXP解析

- **JAXP的SAX支持**

- **SAX解析XML文档**

- **实现思路:**

- 1、在工程中引入Xerces-J具体解析器实现类jar包
- 2、自定义事件监听器继承自DefaultHandler
- 3、通过SAXParseFactory的newInstance()方法创建SAX解析器工厂对象
- 4、通过SAXParseFactory对象的newSAXParser()方法创建SAXParser对象
- 5、调用SAXParser对象的parse()方法解析XML文档



# DOM、SAX和JAXP解析

- JAXP的SAX支持

- SAX生成XML文档

```
<书籍列表>
  <计算机书籍>
    <书名>Java思想</书名>
    <作者>小王</作者>
    <价格>79.00</价格>
  </计算机书籍>
  <计算机书籍>
    <书名>Sprint指南</书名>
    <作者>小李</作者>
    <价格>89.00</价格>
  </计算机书籍>
</书籍列表>
```

# DOM、SAX和JAXP解析

- JAXP的SAX支持

- SAX生成XML文档

- 实现思路

- 1、创建保存xml的结果流对象
- 2、获取sax生产工厂对象实例
- 3、获取sax生产处理器对象实例
- 4、获取sax生产器
- 5、生产文档及文档中的元素