# XML
# 智能信息检索技术

李新叶 著

中国电力出版社
CHINA ELECTRIC POWER PRESS

# XML
# 智能信息检索技术

李新叶 著

中国电力出版社
CHINA ELECTRIC POWER PRESS

## 内 容 提 要

全书共分为6章，主要内容有XML技术、XML信息检索基础、XML语义检索、XML聚类研究、基于自然语言接口的XML语义检索，以及基于本体的XML语义检索。附录提供了一些主要的程序代码。

本书可为从事XML智能信息检索的科技人员提供参考，也可作为普通高等院校计算机科学与技术等相关专业的教学用书。

**图书在版编目(CIP)数据**

XML智能信息检索技术/李新叶著. —北京：中国电力出版社，2013.11

ISBN 978-7-5123-4938-4

Ⅰ.①X… Ⅱ.①李… Ⅲ.①智能检索系统-可扩充语言-程序设计 Ⅳ.①G354.4-39

中国版本图书馆CIP数据核字(2013)第223456号

中国电力出版社出版、发行

（北京市东城区北京站西街19号 100005 http://www.cepp.sgcc.com.cn）

北京丰源印刷厂印刷

各地新华书店经售

*

2013年11月第一版 2013年11月北京第一次印刷

787毫米×980毫米 16开本 11印张 208千字

印数0001—3000册 定价**28.00**元

# 前　言

XML 以其简单性、开放性、可扩展性和互操作性等特点，已经成为驻留在 Web 上的主要信息表示形式之一。目前，多个应用领域已经确定了与 XML 相关的行业标准，例如：SOC 设计中 IP 组件的 XML 描述，电子商务中的 ebXML、数学领域的 MathML、地理信息系统的 GML，以及 XML 在数字图书馆中的应用等。随着 XML 技术在各行各业的广泛应用，基于 XML 文档的结构化及自描述等特性进行有效的信息检索已成为国际信息检索领域研究的热点问题之一。2002 年成立的 INEX（Initiative for Evaluation of XML Retrieval）会议是著名的 XML 测评会议，是一个专门为 XML 检索的不同方法提供统一测评的平台。目前 INEX 会议是世界上最全面、最深入地研究 XML 检索的活动。

对于 Web 下的 XML 信息检索系统而言，输入界面的设计应该是对用户友好的，即输入接口应该简单化、智能化，同时检索结果应该是符合用户查询语义需求的。本书从上述角度考虑，介绍了 XML 智能信息检索相关技术，主要针对基于简单输入形式（标签-关键词形式）和基于自然语言接口的 XML 语义检索进行了研究，围绕上述核心内容进行了一些展开研究，包括标签-关键词输入形式的查询扩展、XML 聚类方法、XML 聚类分析，最后对基于本体的 XML 语义检索进行了探讨。本书的内容是在作者攻读博士学位期间的一部分研究内容基础上进行后续研究所得，在研究过程中作者发表了多篇期刊和会议论文，这些成果为本专著的写作奠定了基础。本书第 3 章中的部分内容来自于作者的博士论文。

全书共分 6 章。第 1 章简要介绍了 XML 相关技术；第 2 章主要介绍了常见的几种信息检索模型，XML 信息检索研究现状及相关技术，包括 XML 索引结构与 XML 检索方法等；第 3 章深入分析了 XML 文档结构潜藏的语义，研究了基于“标签-关键词”输入的 XML 语义检索，最后基于关联规则挖掘技术提出了适用于上述

XML 语义检索的查询扩展方法；第 4 章首先研究将聚类技术用于 XML 语义检索系统，以提高 XML 语义检索系统的性能，然后研究新的聚类技术——谱聚类并将其用于 XML 文档检索结果的自动聚类，最后研究基于相关反馈的聚类技术并用于 XML 文档检索结果的优化输出；第 5 章在第 3 章的基础上研究基于自然语言接口的 XML 语义检索，主要研究将自然语言描述的查询语句转变为“标签-关键词”形式的查询输入，以适合于 XML 语义搜索引擎；第 6 章通过对已有的 XML 搜索引擎和语义搜索相关研究的分析，提出了一种基于领域本体的 XML 语义搜索方案，本方案不仅使用了领域本体来丰富 XML 文档和查询的语义描述，同时还考虑了 XML 文档的结构。附录提供了与 XML 检索有关的主要的程序代码。

在本书的撰写过程中，参考了大量国内外文献资料，这些资料为作者的深入研究提供了基础，并为本书的撰写提供了借鉴，对这些主要参考文献的作者表示感谢。另外，作者指导的研究生黄涵娟、杨林慧分别对本书的第 5 章、第 6 章内容的撰写有所贡献，在此表示感谢。

本书的出版受到河北省教育厅指导性计划项目的资助（Z2012038）。

XML 智能信息检索是 XML 信息检索研究领域的发展趋势，涉及的技术很广泛，本书仅对部分内容作了一定的研究。由于作者的学术水平和科研能力有限，有一些问题研究得不够深入，此外，作者在撰写本书的过程中难免会有疏漏的地方，望广大专家、读者提出宝贵意见，以不断完善我们的研究成果。

2013 年 8 月

# 目 录

前言

**第 1 章 XML 技术** …… 1

1.1 标记语言的发展 …… 1

1.2 可扩展标记语言 XML …… 3

1.3 DTD 简介 …… 5

1.4 XML Schema …… 8

1.5 XPath …… 9

1.6 XML 文档解析器 …… 10

1.7 XML 特点及其应用 …… 16

1.8 章节组织结构 …… 17

**第 2 章 XML 信息检索基础** …… 18

2.1 信息检索系统 …… 18

2.2 信息检索模型 …… 19

2.3 信息检索评价 …… 23

2.4 XML 信息检索 …… 24

**第 3 章 XML 语义检索** …… 32

3.1 引言 …… 33

3.2 XML 文档及其结构语义 …… 34

3.3 XML 语义检索 …… 37

3.4 XML 语义搜索理论基础——多节点间语义相关的判断规律 …… 40

3.5 XML 语义检索的索引结构 …… 42

3.6 XML 快速语义搜索算法 …… 44

3.7 实验分析 …… 48

3.8 检索结果排序 …… 50
3.9 XML 语义检索系统的查询扩展技术 …… 54
3.10 小结 …… 68

**第 4 章 XML 聚类研究** …… 69
4.1 概述 …… 69
4.2 聚类分析基础 …… 69
4.3 XML 文档聚类 …… 71
4.4 用聚类技术改进 XML 语义检索 …… 73
4.5 基于谱分析的 XML 文档聚类方法 …… 81
4.6 改进的多路谱聚类算法 …… 87
4.7 基于改进的多路谱聚类算法实现对 XML 文档的聚类 …… 94
4.8 用相关反馈信息指导 XML 聚类 …… 106
4.9 一种基于用户查询日志的高效 XML 聚类算法 …… 109

**第 5 章 基于自然语言接口的 XML 语义检索** …… 117
5.1 基于自然语言接口的信息检索概述 …… 117
5.2 自然语言接口关键技术 …… 119
5.3 基于英语接口的 XML 信息检索 …… 122
5.4 中文 XML 文档的自然语言接口分析 …… 124

**第 6 章 基于本体的 XML 语义检索** …… 131
6.1 本体相关知识 …… 131
6.2 基于本体的语义检索模型 …… 133
6.3 基于领域本体的 XML 智能搜索 …… 134

附录 A XML 语义检索索引 Java 程序 …… 144
附录 B XML 语义检索程序 …… 149
附录 C XML 谱聚类程序 …… 159

**参考文献** …… 162

# 第1章 XML 技术

## 1.1 标记语言的发展[1-3]

对一篇文档标记是指在文档中添加一些额外的注释或代码。文档标记提供了文档的布局或结构化信息，应用这些信息可以很容易地区分文档的不同部分，如果没有这些信息，就要把文档当成一个整体来处理。标准通用置标语言（Standard Generalized Markup Language，SGML）作为国际标准（ISO8879）于1986年首次公布，它是一种通用的文档结构描述标记语言，为语法标记提供了非常强大的工具，同时具有很好的扩展性，因此在数据分类和索引中非常有用。但SGML复杂度太高，不适合网络的日常应用，而且开发成本高、不被主流浏览器所支持，使得SGML在Web上的推广受到阻碍。

因此，1989年Tim Berners-lee依据SGML开发出一种非常简单的超文本置标语言（HyperText Markup Language，HTML），随后又对其进行修改、升级，HTML从1.0版本到5.0版本，扩展了许多命令以提供更强大的功能。HTML是一种格式化标签语言，由一些格式标签和资源引用构成，通过标签来设置文本的格式，浏览器对HTML文件中的格式标签进行解释并以指定的方式显示出来。HTML文件除了包含信息的内容外，还包含有信息的格式，如文本的字体、文本标题、段落、列表等，此外还有被称为超链接的资源引用标签。

HTML包含标记头、标记尾和属性定义。HTML标签指令写在尖括弧“<>”内部，指令执行完毕，末尾需要利用反斜线关闭此标签指令。标签内可以含有多个属性，属性之间用空格分隔。一组标记以及其包含的内容称为组件，HTML文件就是由许多组件构成的。一个HTML文件的结构基本上可以分为两部分，一部分称为标题区（Head Section），另一部分称为主体区（Body Section）。文件结构标记用来标示何处属于标题，何处属于主体。HTML文件中最基本的结构如下：

<HTML>

<HEAD>
<TITLE>文件标题</TITLE>
</HEAD>
<BODY>
文件主体
</BODY>
</HTML>

其中，<HTML>：标示整篇 HTML 文件。一个标准的 HTML 文件是一篇以<HTML>开头，而以</HTML>结束的文件。

<HEAD>：标示文件标题区。HTML 文件中，由<HEAD>和</HEAD>所包含的区域称为文件的标题区。通常 HEAD 标识的段落部分都包含在 HTML 区域之中。

<TITLE>：网页标题。<TITLE>…</TITLE>是 HTML 文件标题区最重要也是最常用的标记。其他标题区的标记还包括<base>、<isindex>、<link>、<nextid>、<meta>等。<TITLE>标记的用途是设置网页标题，这个标题会显示在浏览器窗口的标题栏上，且不出现在浏览器的页面（page）文字中。而大部分浏览器的收藏（My Favorites）、书签（BookMark）或历史记录列表（History List）功能，也都是以这个文件的标题作为名称的。由<TITLE>…</TITLE>所标注的文字没有长度限制，但是太长的标题，有可能会被截掉，并且也不容易记忆。

<BODY>：标示文件主体区。在 HTML 文件中，由<BODY>…</BODY>所包含的区域称为文件的主体区，通常是在<HEAD>区段之后。<BODY>标记的属性为 background 属性。这个属性可以指定一个图形文件（一般为 Gif 或 Jpeg），作为背景图案，整个网页会用该图案填充。

HTML 标签指令和指令类型都较少，因此简单易用，HTML 广泛用于 Web 应用中，编写针对这种标记语言的软件也非常容易，因此，HTML 使 Internet 获得巨大成功。但是，HTML 的缺点也日益显现出来。首先，作为一种简单的表示性语言，HTML 把数据和显示格式一起存放，如果只想使用数据而不需要格式，则很难分离哪些是数据，哪些是格式，从而无法表达数据内容。而这一点恰恰是电子商务、智能搜索引擎所必需的。另外，HTML 语言不能描述矢量图形、数学公式、化学符号等特殊对象，在数据显示方面的描述能力也不尽如人意。最重要的是，HTML 只是 SGML 的一个实例化的子集，可扩展性差，用户根本不能自定义有意义的置标供他人使用。这一切都成为 Web 技术进一步发展的障碍。

在这种情况下，开发一种兼具 SGML 的强大功能、可扩展性以及 HTML 的简单性

的语言势在必行。由此诞生了可扩展标记语言（eXtended Markup Language，XML）。而随着 Internet 的发展，1998 年 2 月 W3C 公布了 XML1.0 版本，立即成为计算机史上一个重要的里程碑。XML 同样是 SGML 的一个简化子集，它将 SGML 的丰富功能与 HTML 的易用性结合到 Web 应用中，以一种开放的自我描述方式定义了数据结构，在描述数据内容的同时能突出对结构的描述，从而体现出数据之间的关系。这样所组织的数据对于应用程序和用户都是友好的、可操作的。

## 1.2 可扩展标记语言 XML[1]

与 HTML 相同，XML 使用尖括弧“<>”描述标签，标签也是成对出现的，但是与 HTML 不同的是：①XML 标签不是预定义的，可根据需要自行定义；②XML标签及其内容称为元素，XML 元素可以由任意多层嵌套组成；③XML 文档只描述数据而不包括显示格式，其显示可以由另一文件描述。XML 采用简单的格式——文本格式，因此可以跨平台使用。

下面结合一个 XML 文档例子说明 XML 的语法格式。

**例 1-1** XML 文档例子。

```
<! --XML 文件示例-->
<? xml version ="1.0" encoding ="gb2312"?>
  <books>
    <book year="2000">
      <title>数据库系统及应用</title>
      <author>
        <name>李丽</name>
      </author>
    </book>
    <book year="2010">
      <title>C++语言</title>
      <author>
        <name>王平</name>
      </author>
    </book>
    </books>
```

XML 文档一般由四部分组成：XML 声明、处理指令、XML 元素和注释。其中 XML 声明和 XML 元素是必需的，而处理指令和注释则是可选内容。

XML 声明用于声明该文档是一个 XML 文档；它的起始标识是“<?”，结束标识是“?>”。还可以有其他的用途，比如定义文档的编码方式是 GB 码还是 Unicode 编码方式，或是把一个样式单文件应用到 XML 文档上用以显示。上例中的 XML 声明语句如下：

<? xml Version ="1.0" encoding ="gb2312"?>

处理指令是包含在 XML 文档中的一些命令性语句，目的是告诉 XML 处理一些信息或执行一定的动作。

XML 元素是 XML 文件内容的基本单元，元素是由起始标记、元素内容和结束标记组成。用户把要描述的数据对象放在起始标记和结束标记之间。例如：

<name>王平</name>

XML 元素中还可以再嵌套别的元素，这样使相关信息构成等级结构。最外层的元素称为根元素，一个 XML 文档中有且仅有一个根元素，其他所有的元素都是它的子元素，在例 1-1 中，<books>就是根元素。在<books>的元素中包括了所有书的信息，每本书都由<book>元素来描述，而<book>元素中又嵌套了<title>和< author >元素。

XML 的元素是拥有属性的，而且可以有许多属性。属性给元素提供进一步的说明信息，它必须出现在起始标记中。属性以“名称/取值对”出现，属性名不能重复，名称与取值之间用“=”分隔，并用引号把取值引起来。例如：

<salary currency="US$"> 25000 </salary>

上面的属性说明了薪水的货币单位是美元。

注释是 XML 文件中用作解释的字符数据，XML 处理器不对它们进行任何处理。注释是用“<! ——”和“ ——>”引起来的，可以出现在 XML 元素间的任何地方，但是不可以嵌套，例 1-1 中的——XML 文件示例——>就是注释。

XML 是一种语法要求比较严格的标记语言。如果一个 XML 文档满足以下要求，则称其为一个结构良好的 XML 文档：

(1) 文档的开始必须是 XML 声明。

(2) 含有数据的元素必须有起始标记和结束标记，起始标记和结束标记应当匹配，且大小写一致。XML 对字母的大小写是敏感的。

(3) 不含数据并且仅使用一个标记的元素必须以/>结束。

(4) 文档只能有一个能够包含全部其他元素的元素，即根元素必须唯一。

(5) 元素只能嵌套不能重叠。

(6) 属性值必须加引号，属性是不允许重复的。

(7) 字符<和 & 只能用于起始标记和实体引用。

(8) 出现的实体引用只有 &amp、&lt、&gt、&apos 和 &quoto。

(9) 文档必须包含一个或多个元素。

(10) 元素必须正确关闭。

例 1-1 就是一个结构良好的 XML 文档例子。

对于一个结构良好的 XML 文档，还要遵守 DTD 或 XML Schema 的语法规定，只有这样才能保证 XML 文档的易读性，同时还能充分地体现数据信息之间的关系，从而更好地描述数据。

## 1.3 DTD 简介[4]

文档类型定义英文全称为 Document Type Definition，简称 DTD，是一套关于标记的语法规则，它定义了 XML 文档的逻辑结构，规定了 XML 文档中所使用的元素、实体、元素的属性、元素与实体之间的关系。

DTD 可以定义 XML 文档的词汇和语法。利用正则表达式，DTD 除了可以说明 XML 文件中哪些元素是必需的、哪些是可选的、元素所能包含的属性等元素本身信息外，还可以描绘元素之间的结构信息。比如，某个元素可以嵌套哪些子元素、子元素的个数，以及出现次序、是否可选等。

DTD 可以存在于 XML 文档内部，也可以在外部单独定义。在外部定义 DTD 的好处是，可以方便地被多个 XML 文档共享，只需要定义一份 DTD 文档，即可为多个 XML 文档定义语义约束。

典型的 DTD 格式如下：

(1) 以 DOCTYPE 声明为起始标志，告诉解析器以下内容属于 DTD。

(2) 位于 DOCTYPE 后的 DTD 名称，必须与 XML 文档中的根元素完全一致，后面是一个“[”号，接下来是 DTD 正文。DTD 正文包括以下结构：

1) 零到多个注释部分，DTD 注释与 XML 注释的语法完全相同。

2) 零到多个<! ELEMENT…>定义，每个<! ELEMENT…>定义一个 XML 元素。

(3) 零到多个<! ATTLIST…>定义，每个<! ATTLIST…>定义一个属性。

(4) 零到多个<! ENTITY…>定义，每个<! ENTITY…>定义一个实体。

(5) 零到多个<! NOTATION…>定义，每个<! NOTATION…>定义一个符号。

使用 ELEMENT 声明 XML 元素的语法格式如下：

<! ELEMENT 元素名 元素内容>

其中，元素名为当前元素指定的元素名称，元素内容用来指定元素的内容类型，它可以分为 EMPTY（空）、子元素类型、混合型、ANY（任意）和 #PCDATA 共 5 种类型。

如果元素内容是 EMPTY，则该元素只可能有属性，不能有元素内容。其中声明空元素的语法如下：

<! ELEMENT 元素名 EMPTY>

定义只能是字符串元素的语句格式如下：

<! ELEMENT 元素名 #PCDATA>

某个元素既有字符串内容，又包含子元素，被称为混合内容的元素。一般来说，XML 文档不推荐使用混合内容的元素。定义混合内容元素的语句格式如下：

<! ELEMENT 父元素名（#PCDATA | 子元素 1 | 子元素 2 | 子元素 3……）*>

元素包含子元素是 XML 文档中最常见的情形，DTD 可以有效地定义各元素之间的父子关系，从而有效地描述整个文档结构。定义元素包含的子元素时，DTD 规定了对有序的子元素及互斥的子元素的定义。定义有序的子元素时用英文逗号（,）作为子元素之间的分隔符，则子元素之间必须遵守所定义的顺序；互斥的子元素表明一系列子元素之间只能出现其中一次。互斥子元素使用竖线（|）分隔，以竖线（|）分隔的多个元素只能出现其中之一。子元素的出现频率通过在元素声明后紧跟一个表示频率的特殊标记来表示，DTD 中表示频率的特殊标记有 3 个：

+　表明子元素可以出现 1 次或多次；

*　表明子元素可以出现 0 次或多次；

?　表明子元素可以出现 0 次或 1 次。

如果在定义子元素时，没有在子元素后指定任何表示频率的特殊标记，则表明这些子元素只能出现一次，且必须出现一次。

DTD 对属性声明的语法格式如下：

<! ATTLIST 元素名 属性名 属性类型 [属性限定条件] [默认值] >

其中，“属性限定条件”和“默认值”两部分是可选的，会有下面几种情况：

在没有指定“元素对属性的约束”时，必须为该属性指定“默认值”；

当“元素对属性的约束”是#REQUIRED时，不能为该属性指定“默认值”；

当“元素对属性的约束”是“IMPLIED”时，不能为该属性指定“默认值”；

当“元素对属性的约束”是“FIXED”时，必须为该属性指定“默认值”。

对属性的限定条件包括：

#REQUIRED：必须的属性，意味着必须为该元素提供该属性；

#IMPLIED：该属性是可有可无的；

#FIXED：该属性的值是固定的，定义时必须指定固定值。使用该元素时无需为其分配该属性，XML处理器会自动为属性增加固定值。

字符类型：CDATA是简单的纯文本字符类型，也是最常用的类型，将简单的文本用做属性值。可以包括任何字符串，但不允许使用“<”、“>”、“&”、“”” 和“ ‘”。如果需要使用，必须使用实体引用。属性值和元素内容都可以是文本类型，但是定义的方法不同。

枚举类型：声明了属性的备选值列表，属性必须从该列表中选择一个值作为属性值。

DTD对实体的声明分为以下3种情况：

(1) 定义普通实体。

普通实体是在XML文档里使用的实体，语法如下：

<! ENTITY 实体名“实体值” >

使用实体的语法如下：

& 实体名；

定义了可在XML文档中使用的实体后，该实体既可以作为某个XML元素的字符串使用，也可以作为某个XML元素的属性值使用。

(2) 外部普通实体。

外部实体的值不在DTD中直接指定，而是专门提供一个文件为该实体指定值——“实体值所在的文件的URL”处的文件来指定。定义外部实体的语法格式如下：

<! ENTITY 实体名 SYSTEM“实体值所在文件的URI” >

<! ENTITY 实体名 PUBLIC“公用实体标识名”“实体值所在文件的URI” >

(3) 外部参数实体。

类似于定义内部参数实体，语法结构如下：

<! ENTITY %实体名 SYSTEM | PUBLIC [“公用实体标识名”]“实体值所在文件的URI” >

目前，DTD是W3C推荐的验证XML文档有效性的唯一正式规范，但它也有许多

不足之处[1]：

（1）DTD 过于复杂，要熟悉它的语法、标记集合需要一定的时间和精力，而且 DTD 采用的是非 XML 语法规则，不能用 XML 工具进行操作处理。

（2）DTD 对数据类型定义支持不够，所定义的数据类型有限，并且都是针对属性而设立的，无法满足电子商务等 Web 应用所需要的丰富数据类型。

（3）扩展机制复杂，也很脆弱，最大的弊病在于不能表达元素之间的相互关系。

（4）DTD 不支持名称空间的机制。

以上种种缺陷，促使 W3C 组织致力于寻求一种新的机制来取代 DTD。在众多的标准之中，Microsoft 公司提出的 XML Schema 较为引人注目。它具有完全符合 XML 语法、丰富的数据类型、良好的可扩展性，以及易被 DOM 等 XML 解析器处理等优点。

## 1.4 XML Schema

XML Schema（XML 模式）用来描述 XML 元素和属性。它包括属性和元素类型说明，是可以为 XML 元素和属性提供数据的类型校验模块。Schema 与 DTD 不同，DTD 本身有自己的语法和要求，而 Schema 是完全符合 XML 语法的，与 DTD 相比，XML Schema 具有以下几个明显的优势[5]：

（1）XML Schema 使用 XML 语法。

（2）XML Schema 支持名称空间。

（3）XML Schema 支持多种数据类型。

（4）XML Schema 具有更为强大和灵活的定义能力。

下面是一个简单的例子[6]：

```
<?xml version="1.0" encoding="GB2312"? >
<Schema xmlns="urn: schemas-microsoft-com: xml-data"
xmlns: dt="urn: schemas-microsoft-com: datatypes">
<ElementType name="name"/>
<ElementType name="sex"/>
<ElementType name="age"/>
<ElementType name="birthday" content="eltOnly">
<element type="year"/>
<element type="month"/>
<element type="day"/>
```

</ElementType>

</Schema>

其中，第1行是一个XML序言，符合XML基本语法要求，第2行和第3行是对Schema域名的一个定义，第4、5、6行分别是对name、sex、age元素的定义，第7行是对birthday元素的一个定义，但birthday元素又包含了year、month和day三个元素，content属性是对元素中的数据进行说明的。根据content中的数据eltOnly可以知道birthday元素的数据只能够由元素构成。

从上面的例子可以知道如何对一个简单的元素进行定义，也可以知道一个由其他元素组成的元素如何定义。通过由元素组成的元素的这种定义，就可以定义一个嵌套结构的XML树的基本结构。

## 1.5 XPath

XPath是查找XML文档中的部分内容的一种语言，目前W3C已推出功能强大的XPath 2.0版本。

XPath使用路径表达式确定XML文档的节点，下面以例1-1所示的XML文档为例简要介绍XPath的语法。XPath使用模式表达式识别XML文档的节点。一个XPath的模式是使用反斜杠“/”分开子元素名称描述路径。

下面的XPath表达式选择元素books下元素book中的所有title元素：

/ books / book/ title

其中，用“/”路径开始代表元素的绝对路径，不用“/”路径开始则代表元素的相对路径，例如：

book/ title

而用“//”路径开始代表整个文档满足条件的所有元素，例如下面的XPath表达式选择文档中所有的book元素：

// book

1. 选择未知元素

通配符“*”可用于选择未知XML元素。例如下面的XPath表达式选择元素books中的所有book元素所属的子元素：

/ books / book/*

下面的XPath表达式选择元素books下所有孙子辈的title元素：

/ books /* / title

2. 选择分支

使用方括号 [] 可以指定特定的元素。例如下面的 XPath 表达式选择元素 books 中的第一个 book 的子元素：

/ books / book [1]

下面的 XPath 表达式选择元素 books 中的最后一个 book 的子元素：

/ books / book [last ()]

3. 选择几个路径

在 XPath 表达式中，使用“|”运算符可以选择几个路径 。例如下面的 XPath 表达式从隶属于 books 的 book 元素中选择所有 title 和 author 元素 ：

/ books / book / title | / books / book / author

4. 选择属性

在 XPath 中，所有属性使用@前缀。例如下面的 XPath 表达式选取所有名为 year 的属性：

//@year

下面的 XPath 表达式，选取所有具有 year 属性的 book 元素：

//book [@year]

下面的 XPath 表达式，选取所有具有 year 等于“2010”属性的 book 元素：

//book [@year= “2010”]

## 1.6 XML 文档解析器

对于一个 XML 文档，应用程序是不能直接使用或进行处理的，必须通过 XML 解析器把文档解析成可操作的文档。XML 文档解析器实际上是一些代码，用来读取文档并分析文档的结构。解析 XML 文档是处理 XML 文档的第一步。一般来说，要对 XML 文档中的数据进行处理、创建 XML 文档、对文档进行添加、删除、修改等操作都要通过 XML 解析器。解析器取得 XML 文档并检查文档是结构良好的或有效的。在大多数情况下，会生成一个解析树，该树含有 XML 文档中的所有对象。

目前，对于 XML 文档的解析主要有两种方法：一个是 DOM（Document Object Model），一个是 SAX（Simple API for XML）。这两种解析方法都可以解析 XML，但是在解析 XML 的方式上有很大的不同，DOM 是一次性地把 XML 文件读入到内存里解析，处理速度很快，而 SAX 则是一部分一部分地解析，所以资源占用较少。目前 Microsoft 和 Java 都支持这两种解析器。