

IBM软件学院系列教材

XML 及相关技术



IBM 软件学院
IBM Software Institute

第一部分:XML 基础与命名空间

Unit 1- XML 简介

HTML 与标记语言	1-1
HTML 简介	1-1
HTML 基本语法	1-2
标记语言	1-3
HTML 与 标记语言的关系	1-3
从 HTML 到 XML	1-4
HTML 的缺点	1-4
XML 的诞生	1-5
XML 的特征及目标(1 of 2)	1-6
XML 的特征及目标(2 of 2)	1-7
XML 的应用	1-9
XML 相关技术标准 (1 of 3)	1-10
XML 相关技术标准 (2 of 3)	1-11
XML 相关技术标准 (3 of 3)	1-12
XML 的优点(1 of 2)	1-13
XML 的优点(2 of 2)	1-14
本章总结	1-15

Unit 2- XML 基本语法

标记语法	2-1
字符及字符编码	2-2
XML 名称规范	2-2
创建一个简单的 XML 文档实例	2-3
定义基本元素	2-3

改进元素	2-4
在浏览器中显示定义的 XML 文档	2-5
XML 文档结构	2-6
元素(Element)	2-7
元素名称	2-7
开始标记、结束标记及空标记	2-8
元素关系	2-9
元素的嵌套	2-10
元素的内容	2-10
元素范例	2-11
属性(Attribute)	2-12
属性范例	2-13
序言(Prolog)	2-13
XML 声明	2-14
字符编码声明(Encoding Declaration)	2-14
文档独立声明(Standalone Declaration)	2-14
DOCTYPE 声明(DOCTYPE Declaration)	2-15
处理指令(Process Instruction,PI)	2-16
样式表(Stylesheet)处理指令	2-17
注释(Comment)	2-18
空白(White Space)(1 of 2)	2-19
空白(White Space)(2 of 2)	2-21
CDATA 段	2-22
字符引用及实体引用	2-23
字符引用	2-23
常用字符引用	2-24
实体引用	2-25
使用实体引用(或字符引用)还是 CDATA 段?	2-26
XML 文档类型	2-27
XML 基本语法总结	2-28
本章小结	2-28

Unit 3- XML 命名空间

为什么需要命名空间?	3-1
命名空间标识	3-4
URL、URN 及 URI	3-5
命名空间的声明	3-6
命名空间的使用	3-6
在元素上使用命名空间	3-7
在属性上使用命名空间	3-8
合法名称(Qualified Name, QName)	3-9
命名空间的范围(或作用域)	3-10
命名空间的覆盖	3-11
缺省命名空间盖	3-13
缺省命名空间的使用	3-14
命名空间综合实例	3-15
命名空间综合实例(继续)	3-16
本章小结	3-16

第二部分:XML 数据模型技术**Unit 4 - 文档类型定义(DTD)**

XML 数据模型技术	4-1
DTD 简介	4-2
DTD 类型	4-3
内部 DTD	4-3
外部 DTD	4-4
即声明内部 DTD,又引用外部 DTD 文件	4-5
使用系统链接方式链接外部 DTD	4-5
使用公共链接方式链接外部 DTD	4-6
公共标识的命名规则	4-6
使用系统还是公共链接方式?	4-7
一个 DTD 实例	4-7
元素类型声明(Element Type Declare, ETD)	4-8
元素内容类型	4-9

简单类型	4-9
带有简单内容的复杂类型	4-10
带有复杂内容的复杂类型	4-10
混合内容类型	4-12
空内容类型	4-14
任何内容类型	4-15
使用量词(Quantifier)符号	4-15
加号“+”的使用	4-16
星号“★”的使用	4-17
问号“?”的使用	4-18
括号“()”的使用	4-19
括号的使用实例	4-19
括号的使用实例	4-20
括号的使用实例	4-21
属性类型声明例	4-22
属性类型	4-23
CDATA 类型	4-23
枚举类型(Enumerated)	4-24
ID 类型	4-25
ID 类型的其他约束	4-26
IDREF 类型与 IDREFS 类型	4-27
ENTITY 与 ENTITYS 类型	4-28
NMTOKEN 与 NMTOKENS 类型	4-29
NOTATION 类型	4-30
属性默认值	4-31
#IMPLIED	4-31
#REQUIRED	4-32
#FIXED	4-32
使用 #FIXED 默认值实例	4-33
具体默认值("x" 或 'x')	4-34
使用具体默认值实例	4-34
实体(Entity)	4-35

实体分类 entity)	4-35
一般实体(General Entity)	4-36
一般实体的声明	4-36
声明内部一般实体	4-37
声明外部已析实体	4-38
未析实体	4-39
声明外部未解析实体	4-39
一般实体的引用	4-40
在 XML 实例文档中引用一般内部实体实例	4-41
在 DTD 中引用一般内部实体实例	4-42
引用一般外部已析实体	4-43
引用一般外部已析实体实例	4-43
引用外部未解析实体	4-44
引用外部未解析实体实例	4-44
参数实体(Parameter Entity)	4-45
参数实体的声明	4-45
声明内部参数实体	4-45
声明外部参数实体	4-46
参数实体的引用	4-46
引用内部参数实体实例(contact.dtd)	4-47
引用外部参数实体实例(main_contact.dtd)	4-48
实体名称	4-49
预定义实体	4-49
符号(Notation)	4-49
符号声明	4-50
符号和 XML 数据	4-51
符号和未解析实体	4-51
DTD 与命名空间	4-52
一个使用了命名空间的 XML 实例文档	4-53
带命名空间的 DTD(customer.dtd)	4-54
带命名空间的 DTD(contact.dtd)	4-55
本章小结	4-55

Unit 5 – XML Schema5-1

模式简介	5-1
模式的用途	5-2
模式设计	5-3
主要模式语言	5-4
DTD 的缺点	5-5
进入模式语言	5-5
术语说明(1 of 2)	5-6
术语说明(2 of 2)	5-6
XML 模式组件	5-7
声明和定义	5-7
全局组件与局部组件的比较	5-8
元素及属性	5-9
数据类型性	5-10
命名类型和匿名类型	5-11
简单类型和匿名类型	5-11
复杂类型定义中的一些概念	5-12
派生复杂类型	5-13
命名空间和 XML Schema	5-14
模式文档及模式组成	5-15
XML 实例文档和模式	5-16
元素声明	5-16
<element> 元素的主要属性	5-17
全局元素声明	5-18
局部元素声明	5-19
元素声明的数据类型	5-20
元素默认值和固定值	5-21
元素默认值	5-21
元素固定值	5-22
属性声明值	5-22
<attribute> 元素的主要属性	5-23
全局属性声明	5-23

局部属性声明	5-24
指定属性的类型	5-25
属性默认值和固定值	5-26
属性默认值	5-26
属性固定值	5-27
简单类型	5-27
简单类型的种类	5-27
简单类型的定义	5-28
<simpleType> 元素的主要属性	5-28
命名简单类型定义	5-28
匿名简单类型定义	5-29
简单类型的限制 (Restriction)	5-29
定义限制 (Restriction)	5-30
面 (facet) 的概述	5-30
面的类型	5-31
继承和限制	5-32
枚举类型	5-33
列表 (List) 类型	5-34
联合 (Union) 类型	5-35
内置简单类型	5-36
内建原始类型(1 of 2)	5-36
内建原始类型(2 of 2)	5-37
内建派生类型	5-37
复杂类型	5-38
定义复杂类型	5-38
命名复杂类型定义	5-39
匿名复杂类型定义	5-39
复杂类型定义选择	5-40
元素内容和内容模型	5-40
在复杂类型定义中使用元素类型	5-41
使用模型组	5-41
sequence 组	5-42

choice 组	5-43
sequence 和 choice 组的嵌套	5-44
all 组	5-45
all 组的规则	5-46
在复杂类型定义中使用属性类型	5-47
派生复杂类型	5-48
扩展复杂类型	5-49
简单内容扩展	5-49
复杂内容扩展	5-50
混合内容扩展	5-51
空内容扩展	5-52
属性扩展	5-53
复杂类型的限制	5-54
简单内容限制	5-54
复杂内容限制	5-55
混合内容限制	5-56
空内容限制	5-57
属性限制	5-58
验证对复杂类型的限制是否合法(1 of 2)	5-59
验证对复杂类型的限制是否合法(2 of 2)	5-60
本章小结	5-60

第三部分:可扩展样式表语言(XSL)

Unit 6 – XPath

什么是 XML 转换?	6-1
XML 转换的几种技术	6-1
级联样式表(CSS2)	6-2
CSS 的缺陷	6-2
可扩展样式表语言(XSL)(1 of 2)	6-3
可扩展样式表语言(XSL)(2 of 2)	6-3
XPath 介绍	6-4
节点相关的一些概念	6-4

XPath 中的节点类型	6-5
文档顺序(Document Order)	6-5
根节点(Root Node)	6-6
元素节点	6-6
属性节点	6-7
文本节点	6-7
XML 实例——一年的 XML 表达	6-8
XPath 表达式介绍	6-9
节点集(Node Set)对象	6-10
布尔对象	6-11
数字对象	6-11
字符串对象	6-12
XPath 中的定位路径(Location Paths)	6-12
绝对定位步长及绝对定位路径	6-13
相对定位步长及相对定位路径	6-13
构造定位步长(Location Step)	6-13
构造定位步长的轴(Axis)部分	6-14
parent 轴	6-15
ancestor 轴	6-15
descendant 轴	6-16
following-sibling 轴	6-16
preceding-sibling 轴	6-17
following 轴	6-17
preceding 轴	6-18
attribute 轴	6-18
namespace 轴	6-19
self 轴	6-19
descendant-or-self 轴	6-19
ancestor-or-self 轴	6-20
child 轴	6-20
XPath 中轴的总结	6-21
构造定位步长的节点测试(Node-Test)部分	6-22

构造定位步长的判断语(Predicate)部分	6-23
定位路径实例讲解——XML 实例文档	6-24
定位路径实例讲解——文档节点树	6-25
<books>节点树	6-25
表达式及路径定位实例	6-26
XPath 核心函数介绍	6-26
XPath 核心函数类型	6-27
节点集函数组介绍	6-27
布尔函数组介绍	6-28
数字函数组介绍	6-28
字符串函数组介绍	6-29
本章小结	6-29

Unit 7- 可扩展样式表语言转换(XSLT)

XSLT 介绍	7-1
XSLT 是一种编程语言	7-2
Hello, World! (XSLT)——XML (hello.xml)	7-2
Hello, World! (XSLT)——XSLT (hello.xsl)	7-3
Hello, World! (XSLT)——转换结果	7-4
XSLT 是基于规则的(Rule-based)	7-4
XSLT 处理的几个基本步骤	7-5
XML 树状表示及 XPath	7-5
XSLT 处理器	7-6
在 XML 文档中连接 XSLT 文件	7-7
XSLT 转换的执行	7-7
XSLT 转换过程	7-8
XSLT 文档结构及处理	7-9
一个复杂的 XSLT 文档	7-10
模板规则(Template Rules)	7-15
匹配模式(Match Patterns)	7-15
模板(Template)	7-16
模板规则的处理	7-16

模板规则的处理过程	7-16
解决模板规则冲突	7-17
内置模板规则	7-17
XSLT 顶层元素(Top-level Element)	7-18
模板规则相关的 XSLT 元素	7-18
<xsl:template> 元素	7-18
<xsl:template> 元素——匹配属性“match”	7-19
<xsl:template> 元素——名称属性“name”	7-19
<xsl:template> 元素——优先级属性“priority”	7-20
<xsl:template> 元素——模式属性“mode”	7-21
<xsl:apply-templates> 元素(1 of 2)	7-22
<xsl:apply-templates> 元素(2 of 2)	7-22
<xsl:apply-templates> 元素——选择属性“select”	7-23
<xsl:apply-templates> 元素——模式属性“mode”	7-23
命名模板规则	7-24
<xsl:call-template> 元素	7-24
调用模板时传入参数	7-24
常用 XSLT 元素	7-25
<xsl:value-of> 元素	7-25
<xsl:value-of> 元素——选择属性“select”	7-25
<xsl:value-of> 元素——disable-output-escaping”属性	7-26
<xsl:for-each> 元素	7-26
<xsl:for-each> 元素——选择属性“select”	7-26
XSLT 条件判断及排序元素	7-27
<xsl:if> 元素	7-27
<xsl:if> 元素——测试属性“test”	7-27
<xsl:choose> 元素	7-28
<xsl:when> 元素	7-28
<xsl:otherwise> 元素	7-29
<xsl:sort> 元素(1 of 2)	7-29
<xsl:sort> 元素(2 of 2)	7-29
<xsl:sort> 元素——选择属性“select”	7-30

<xsl:sort> 元素——语言属性“lang”	7-30
<xsl:sort> 元素——数据类型属性“data-type”	7-30
<xsl:sort> 元素——排序顺序属性“order”	7-31
<xsl:sort> 元素——大小写顺序属性“case-order”	7-31
使用 <xsl:output> 顶层元素控制输出选项	7-31
<xsl:output> 元素语法	7-31
输出方法属性“method”	7-32
版本属性“version”	7-32
编码属性“encoding”	7-32
“omit-xml-declaration”属性	7-32
“standalone”属性	7-33
“doctype-public”属性	7-33
“doctype-system”属性	7-33
缩进属性“indent”	7-34
本章小结	7-34

第四部分:XML 解析器

Unit 8- Document Object Model8-1

什么是 XML 解析	8-1
XML 解析处理 API 类型	8-2
什么是 XML 解析器(Parser)	8-2
使用解析器解析 XML 文档的基本步骤	8-3
解析器种类	8-3
文档对象模型(Document Object Model, DOM)	8-4
DOM 的优点与缺点	8-4
DOM 节点类型	8-4
基本接口	8-5
文档节点及 Document 接口	8-5
元素节点及 Element 接口	8-5
文本节点及 Text 接口	8-6
属性节点及 Attr 接口	8-6
Node 接口	8-6

节点列表及 NodeList 接口	8-7
命名节点列表及 NamedNodeMap 接口	8-7
使用 DOM 解析的步骤概述	8-8
使用 DOM 解析的步骤	8-8
1. 指定一个解析器(parser)实现(可选)	8-9
2. 创建 Document 构造器对象(或 XML 解析器)	8-10
设置解析器属性/特性	8-10
3. 解析 XML 文档并创建 Document 接口	8-10
输入源(Input Source)类型	8-11
4. 规格化(Normalize)文档树(Tree)	8-11
5. 得到根节点	8-11
Node 接口中定义类成员变量	8-12
Node 接口的常用方法(1 of 2)	8-12
Node 接口的常用方法(2 of 2)	8-13
Document 接口常用方法	8-13
Element 接口常用方法	8-14
Attr 接口常用方法	8-14
DOM 实例	8-15
DOM 使用总结	8-15
本章小结	8-16

Unit 9- SAX

什么是 SAX?	9-1
为什么要 SAX?	9-2
SAX 最基本特征	9-2
SAX 事件模型	9-2
SAX 事件	9-3
文档事件	9-3
元素事件	9-3
字符事件	9-4
其他高级事件	9-4
解析器事件示例	9-4

使用 SAX 解析的步骤	9-5
创建 SAX 解析器	9-5
编写内容处理程序	9-6
SAX 中的事件所对应的回调函数(1 of 2)	9-6
SAX 中的事件所对应的回调函数(2 of 2)	9-7
XML 内容处理程序范例	9-8
编写错误处理程序	9-9
错误处理程序范例	9-9
把内容及错误处理程序合在一起	9-10
使用解析器对 XML 进行解析	9-10
给解析器设置内容及错误处理程序	9-10
一个比较完整的实例代码	9-11
SAX 总结	9-12
SAX 的缺点	9-12
DOM 还是 SAX ?	9-12
本章小结	9-13

课程描述:

- 本课程提供对 XML 以及相关技术的介绍。学完本课程,学生可以掌握使用 XML 的必备知识,包括:XML 概念以及实践。本课程针对架构师、设计分析人员、开发人员在构建电子商务应用时要用到的 XML 基本知识。
- 预备知识:熟悉互联网技术、掌握 HTML 与 Java 技术为佳。
- 推荐课时:48 课时
- 授课对象:企业应用架构师、设计分析人员、开发人员

Unit 1– XML 简介

章节目标

通过本章学习,您应该能够:

- 1、了解 HTML 与标记语言
- 2、了解从 HTML 到 XML 的发展
- 3、了解 XML 相关技术标准
- 4、了解 XML 的优点

HTML 与标记语言

- HTML 是当今最流行、应用最广泛的一种标记语言。
- 下面我们简单地介绍一下 HTML、标记语言及它们间的关系。

注释:

HTML 简介

- HTML 是 Hyper-text Markup Language 的缩写,即超文本标记语言。
- HTML 是一种纯文本类型的语言,您可以使用任何文本编辑打开、编辑及查看
- HTML 源代码,当然也可以用专门的可视化编辑器,例如:
FrontPage
Macromedia Dreamweaver。
- HTML 是一种表现语言,用来定义 Web 上文字、图像及声音等的显示及格式。

注释:HTML 简介

- 只要您连接到了 Internet,HTML 便会出现在我们的身边。只要您打开一个网页,便从网上获取了一个 HTML 文件,然后通过浏览器解析成我们熟悉的可视化界面。例如,在 IE 浏览器中输入以下 URL:<http://www.ibm.com/cn>
- 一个 HTML 文件就会在 IE 浏览器中打开,当然您看到的是一个解析后的显示信息,要查看该 HTML 文档的源代码,可以选择浏览器的主菜单【查看->源文件】,您将看到一个文本文件在编辑器中打开,文件中那些用尖括号括起来的就是用 HTML 语言编写的标记语言。
- HTML 是 Hyper-text Markup Language 的缩写,即超文本标记语言。从它的名称上看,我们可以知道 HTML 首先是一种标记语言,它不仅能保存文本信息,还能保存非文本信息,例如图像、声音及电影等,所以叫做超文本。事实上,HTML 不是能保存图象等本身的信息,应该是它们的显示信息。
- HTML 是一种纯文本类型的语言,使用 HTML 编写的网页文件也是标准的纯文本文件。您可以使用任何文本编辑打开、编辑及查看 HTML 源代码,当然也可以用专门的可视化编辑器,例如 FrontPage,Macro media Dreamweaver 等。自从 Web 出现以来,HTML 便成为了创建 Web 页面的标准语言,它几乎可以满足 Web 页面中的所有需求。

HTML 基本语法

- HTML 的语法非常简单。
- 主要标记:
<HTML>
<HEAD>
<BODY>

<TABLE>, <TR>, <TD>, <TH>
<IMAGE>
<A>

- HTML 语言的起始标记是
<HTML>, 然后包含 <HEAD>
及 <BODY>。

注释: 基本语法

- 例 1-1 订单信息 HTML (order.html)

```
<HTML>
<HEAD>
<META http-equiv="Content-Type"
content="text/html; charset=GB2312">
<TITLE>订单详细信息</TITLE>
</HEAD>
<BODY>
<H2>订单详细信息</H2>
<H3>订货日期:2001-01-01</H3>
<H3>客户基本信息</H3>
<TABLE border="0">
  <TBODY>
    <TR>
      <TD>姓名:</TD>
      <TD>Li SG</TD>
    </TR>
    <TR>
      <TD>Email:</TD>
      <TD>lisg@sample.com</TD>
    </TR>
    <TR>
      <TD>电话:</TD>
      <TD>62334455</TD>
    </TR>
  </TBODY>
</TABLE>
<P>
<H3>送货地址</H3>
北京北京市上地六街
<br>
邮政编码:100085
<br>
送货方式:送货上门
</P>
<P>
<H3>收款地址</H3>
北京北京市上地六街
<br>
邮政编码:100085
```

```
<br>
收款方式:货到付款
</P>
<H3>产品信息</H3>
<TABLE border="1">
  <TBODY>
    <TR>
      <TH>产品号</TH>
      <TH>产品名</TH>
      <TH>数量</TH>
      <TH>单价</TH>
      <TH>小计</TH>
    </TR>
    <TR>
      <TD>872-AA</TD>
      <TD>精通 EJB</TD>
      <TD>1</TD>
      <TD>48.95RMB</TD>
      <TD>48.95</TD>
    </TR>
    <TR>
      <TD>926-AA</TD>
      <TD>IBM ThinkPad T40</TD>
      <TD>1</TD>
      <TD>18000.00RMB</TD>
      <TD>18000.00RMB</TD>
    </TR>
  </TBODY>
</TABLE>
<BR>
订单总价:18048.95
</BODY>
</HTML>
```

- HTML 语言的起始标记是 <HTML>, 然后下面包含两个标记 <HEAD> 及 <BODY>, <HEAD> 用来提供 HTML 页面的头信息, 例如文档类型、编码等, <BODY> 里面是该 HTML 页的真正显示内容。