XML的面向对象语言特性

况 旭,刘 波
(华南师范大学 计算机学院,广东 广州 510631)

摘要:面向对象程序设计思想是现在程序设计的主流思想,它通过给程序中加入扩展语句,把函数"封装"进编程所必需的"对象"中,使得复杂的工作条理清晰、编写容易。有着广泛应用的XML也必包含这一思想。文中首先详细介绍了XML的起源与特点,接着对面向对象语言特性进行了简单的阐述:如抽象、封装、继承、多态性,最后通过XML实例与面向对象语言的特性相结合来着重分析XML的面向对象的语言特性。因此,通过使用面向对象程序设计的思想与方法,能够更加灵活方便地构造和设计XML。

关键词:XML;面向对象;抽象;封装;继承;多态
中图分类号:TP31 **文献标识码**:A **文章编号**:1673-629X(2010)01-0054-04

XML Object-Oriented Language Characteristic

KUANG Xu, LIU Bo
(Dept. of Computer, South China Normal University, Guangzhou 510631, China)

Abstract: The object-oriented programming design concept is popular in programming mainstream thought, it added to the program through the function expansion statements, "packaging" into a programming required "object", makes the complex work clear and written easy. XML, which is used very widely should contain this thought. First introduce the origins and characteristics of XML, then elaborate the object-oriented language characteristic, such as abstract, encapedation, inheritance, polymorphism, and finally through the combination of XML example and the characteristics of object-oriented language to focus on analysis of the XML object-oriented language characteristic. Therefore, be able to conveniently construct and design XML by the object-oriented programming design thoughts and methods.

Key words: XML; object-oriented; abstract; encapedation; inheritance; polymorphism

0 引 言

XML语言现在已经应用得非常广泛。在应用领域,XML语言已经成为信息交流的通用媒介。在中间件领域,XML已经整合成一种标准协议,如SOAP。在系统领域,便利的自我描述的XML成为格式化和非格式化数据的存储方式[1]。在面向对象设计思想流行的今天,应用广泛的XML也应具有此特性。

1 XML概述

1.1 XML的起源

SGML(Standard Generalized Markup Language,标准通用化标记语言)和HTML(Hypertext Markup Language,超文本标记语言),它们都是非常成功的标记语言。SGML能够创建成千上万的标记语言,同时具有极好的可扩展性,因此在分类和索引数据中非常有用。目前,SGML多用于科技文献和政府办公文件中。但是由于SGML的复杂性、昂贵性以及几大IE厂商拒绝使用SGML,造成了SGML在互联网上传播和利用的巨大障碍。相反,HTML的简单、免费性以及广泛的支持,使得它在世界范围内得到广泛的应用。但是HTML的简单性、代码臃肿以及数据与表现混杂,直接导致了人们开始思考一种全新的标记语言来代替SGML和HTML,使它既具有SGML的强大功能和可扩展性,同时又具有HTML的简单性。因此,1996年,万维网协会(http://www.w3c.org)决定专门成立一个SGML专家小组来从事这项工作,设计一种可扩展的标记语言(Extensible Markup Language),Sun公司的Jon Bosak担任小组的指挥。Jon Bosak和他的专家们对SGML进行了大量的删减,同时也保留了SGML的所有精华。此后,XML不断发展演化,最后于1998年修订完成。W3C于1998年2月批准了XML的1.0

收稿日期:2009-04-29;**修回日期**:2009-07-23
基金项目:国家技术创新基金项目(08C26214411198)
作者简介:况 旭(1984-),男,江西宜春人,硕士研究生,研究方向为网络计算;刘 波,副教授,研究方向为网格技术。

版本,一个崭新而大有前途的语言诞生了[2]。

1.2 XML 的特点

(1)描述数据内容的语言。本身并不决定数据该如何显示,数据的显示由 XSL 决定。

(2)可由用户按需要增加标记,可扩展。如数学标记语言 MATHML、财经标记语言 FPML、电子商务标记语言 EBXML 等。

(3)对语法有严格的要求。所有 XML 的文件都必须经过严格的"验证"过程才算完成,文件格式容易转换。

1.3 XML 的优势

XML 最大的优势在于对各种数据的管理。任何系统都可以通过 XML 解析器来读取 XML 数据,因此它的数据可以通行各处,而不用担心系统不支持的问题[2]。

(1)具有强大的数据检索功能。用语义标记作为搜索索引,在文件中截取关键部分。

(2)具有强大的数据显示功能。XML 将数据保存的格式与数据显示的方式分开,使得 XML 文件可以轻易地更换数据显示的方式。

(3)具有强大的数据交换功能。XML 语法简单,可以轻易地在所有机器上解读和各种平台上使用,使得 XML 有潜力成为一个通行四海皆准的标记语言。

2 面向对象的语言特性

2.1 抽　象

抽象就是忽略一个主题中与当前目标无关的那些方面,以便更充分地注意与当前目标有关的方面。抽象并不打算了解全部问题,而只是选择其中的一部分,暂时不用考虑部分细节[3]。

2.2 封　装

封装是把过程和数据包围起来,对数据的访问只能通过已定义的界面。面向对象计算始于这个基本概念,即现实世界可以被描绘成一系列完全自治、封装的对象,这些对象通过一个受保护的接口访问其他对象[3]。

2.3 继　承

继承提供了创建新类的一种方法,这种方法就是说,一个新类可以通过对已有类进行修改或扩充来满足新类的要求。新类共享已有类的行为,而自己还具有修改的或额外添加的行为。因此,可以说继承的本质特征是行为共享[3]。

2.4 多态性

多态性是指允许不同类的对象对同一消息作出响应。多态性语言具有灵活、抽象、行为共享、代码共享的优势,很好地解决了应用程序函数同名问题。

3 XML 面向对象的语言特性

3.1 XML 面向对象语言的抽象特性

抽象就是从众多相类似的事物中抽取一些共同的特征。XML Schema[4]是 XML 的定义语言,它的主要作用就是封装 XML 元素。比如,学生 student,抽取出共有的特性:学号:Number,姓名:Name,班级:Class 等几个主要的特征。下面实现了对一个 student 的抽象。

DataTypes.xsd:

```
<? xml version="1.0" encoding="UTF-8"? >
< xs: schema xmlns: xs = " http://www. w3. org/2001/XMLSchema">
<xs:complexType name="student" abstract="true">
<xs:sequence>
<xs:element name="Name" type="xs:string"/>
<xs:element name="Number" type="xs:string"/>
<xs:element name="Class" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:schema>
```

从 TypeData.XSD 文件中可以得出,Student 是一个复杂元素类型,具有面向对象语言的抽象特性。

3.2 XML 面向对象语言的封装特性

在面向对象思想中,封装是指把对象的过程与数据包围起来形成一个黑箱,这样当使用对象时就不知道它的内部工作原理,它就是面向对象语言中的一个类。而在 XML 中,XML Schema 变成了创建一种复杂的类型,这种类型是事先定义好的,在别的文档中引用此类型就可以方便地创建数据和应用数据。

比如:有一个学生 student,他有学号:Number,姓名:Name 以及班级:Class。因此开始创建一个 student 类型,并放到 Student.xsd 数据模式中。该模式包含可以被许多不同模式使用的一般数据类型。例如,Mschool(中学),Hschool(大学)等模式都需要 student 的定义。

Student.xsd:

```
<? xml version="1.0" encoding="UTF-8"? >
< xs: schema xmlns: xs = " http://www. w3. org/2001/XMLSchema">
<xs:complexType name="student">
<xs:sequence>
<xs:element name="Name" type="xs:string"/>
<xs:element name="Number" type="xs:string" />
<xs:element name="Class" type="xs:string"/>
</xs:sequence>
</xs:complexType>
```

</xs:schema>

在这个文档中,定义了一个名为 Student 的元素。其实它只是个复杂元素类型,每个复杂元素类型都描述一个元素的详细信息。为了成为可引用的元素,元素必须是全局元素,也就是说,是<schema>的直接子元素。局部元素是嵌套在另一个组件内的元素声明,例如,Number 元素是全局元素 student 内的局部元素。现有一个元素 Mschool(中学),它包含学生,教师等类型的人。当编辑 Mschool 的 XSD 文档时,可以直接将<xs:element name="Student"type="Student"/>引入即可,这样便实现了面向对象思想中的封装性。

3.3 XML 面向对象语言的继承特性

继承在面向对象中是一种层次模型,它鼓励类的重用和重构,因此它也实现了软件的重用。在面向对象语言中,子类对父类的继承实现了对父类属性和方法的重用[5]。在 XML 语言中,XML Schema 也可以使用抽象类型或仅使用指定基类型扩展或派生的标记来实现这一重用。

3.3.1 通过定义抽象数据类型实现继承

抽象类型不能在实例文档中使用,它们只是为其派生的类型提供占位符[6]。在下面的例子中,将 student 定义为抽象类型,同时定义另一个元素中学生(Hstudent),此元素不但有姓名(Name),学号(Number),班级(Class),还有年级(Grade)。很明显,Hstudent 由 student 派生而来,故可以继承 Name, Number, Class 三个元素而后添加 Grade 元素。为了将 student 定义为抽象类,需要添加 abstract="true" 属性。<xs:complexType name="student" abstract="true''>(通过抽象来实现继承也可以是普通元素 element)。而 Hstudent 的继承是通过<extension base="student">语句来实现的。

```
<xs:complexType name="Hstudent">
<xs:complexContent>
<xs:extension base="student">
< xs:sequence maxoccurs="unbounded">
<xs:element name="Grade" type="xs:string"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
```

3.3.2 通过扩展实现继承

这种方式可以不用将 student 变为抽象类型,而只通过使用 extension base 关键字扩展 student 类型以创建 Hstudent 类型。

```
<xs:complexType name="Hstudent">
<xs:complexContent>
<xs:extension base="student">
<xs:sequence>
<xs:element name="Grade" type="xs:string"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
```

以上代码说明,不管什么时候引用 Hstudent,它始终包含四个元素:name, class, number 和 grade,但要注意的是,通过扩展来实现继承,student 必需是全局元素。

3.3.3 通过限制实现继承

使用这种方法可以实现部分继承,类似 C++,java 等面向对象语言中类的 Public, Private 等成员属性。例如,幼儿园的学生(Nstudent)都在一起学习、生活。因此可以屏蔽 Student 类型中的 Class 元素。通过使用 restrition base="student"语句。

```
<xs:complexType name="Nstudent">
<xs:complexContent>
<xs:restriction base="student">
<xs:sequence>
<xs:element name="Name" type="xs:string"/>
<xs:element name="Number" type="xs:string" />
</xs:sequence>
</xs:restriction>
</xs:complexContent>
</xs:complexType>
```

3.3.4 通过 final 关键字实现禁止继承

在面向对象编程中,可以将一些接口和类声明为 final,通过这种方式它们就不能产生子类[7]。例如,一所大学的名称、地址、校长一般不轻易让人继承或扩展。因此可以将复杂元素 School 声明为 final。

```
<xs:complexType name="School" final="#all">...</xs:complexType>
<xs:complexType name="School" final="extention">...</xs:complexType>
<xs:complexType name="School" final="restriction">...</xs:complexType>
```

当 final 为 #all 时,禁止 School 扩展和限制,final 为 extention 或 restriction 时,是禁止 School 扩展或限制。

3.4 XML 面向对象语言的多态性

多态性意味着在不同的上下文中对某对象赋予不同的意义或用法的能力,具体而言,就是允许对象有多种形式。在 C++,JAVA 这样的编程语言中,多态性指的是对输入的不同反应。详细地说,它是子类对相同消息做出不同反应的能力。简单的继承允许两个子

类各自添加不同的方法,而多态性实现同一功能,但用的是不同的方法和不同的子类。由于 XML 不是行为语言,而是一种标记语言,所以多态性出现在属性级别[8]。采用前文的继承示例,Hsutdent 从 student 中继承了 name, number, class 属性,但是希望 Hstudent 的 Number 属性与 Student 的 Number 属性不同,而是某些特定的数字号码,因此可以在模式中实现这一规则,当 Hstudent 中 Number 不是这些特定的数字号码时,那么它将报告错误。

代码如下:

```
//定义 Hstudent 复杂数据类型
<xs:complexType name="Hstudent">
<xs:complexContent>
<xs:restriction base="student">
<xs:sequence>
<xs:element name="name" type="xs:string"/>
<xs:element name="class" type="xs:string"/>
<xs:element name="number" type="extentNumber"/>
</xs:sequence>
</xs:restriction>
</xs:complexContent>
</xs:complexType>
//定义 number 元素
<xs:simpleType name="number">
<xs:restriction base="xs:string"/>
</xs:simpleType>
//定义 extentNumber 元素
<xs:simpleType name="extentNumber">
<xs:restriction base="number">
<xs:pattern value="[0-9]{10}"/>
</xs:restriction>
</xs:simpleType>
```

4　结束语

通过上文的分析,XML 具有良好的面向对象特性。随着 XML 应用不断深入,规模不断扩大,设计 XML 的复杂度也就随之增加。因此,文中将更加深入的研究与分析,通过借鉴软件工程的思想来快速方便地构建 XML。

参考文献:

[1] Bohrer K, Liu Xuan, McLaughlin S, et al. Object Oriented XML Query by Example[M]. Berlin, Heidelberg: Springer - Verlag, 2003.

[2] Goldfarb C F, Prescod P. XML 手册[M]. 第 4 版. 王艳斌, 译. 北京:电子工业出版社, 2003:2 - 15.

[3] Dewhurst S C. C++ 必知必会[M]. 荣　耀, 译. 北京:人民邮电出版社, 2006:1 - 5.

[4] Lee D. Comparative analysis of six XML schema languages [J]. ACM SIGMOD Record, 2002(29):117 - 151.

[5] 周凯波,金　斌,周剑岚,等. 基于 XML 的面向对象案例表示方法[J]. 武汉理工大学学报:信息与管理工程版, 2005, 27(3):78 - 80.

[6] 李　浩,孙统风,孟现飞,等. 基于面向对象思想构建 XMLSchema[J]. 微机发展(现更名:计算机技术与发展), 2003, 13(6):59 - 61.

[7] Wang Guo ren. Extending XML schema with object - oriented features[J]. Information Technology Journal, 2005, 4(1):44 - 54.

[8] 姜岩一,宫义山,王国仁,等. 基于面向对象 XML 文档的面向方面模型[J]. 沈阳建筑大学学报, 2006, 22(4):673 - 676.

(上接第 53 页)

不增加系统复杂性的基础上,提高了系统的快速反应能力,同时保证 Agent 的理性思维能力。具有一定的理论意义和应用价值。

参考文献:

[1] Tokoro M. Computational Field Model: Toward a New Computing Model/Methodology for Open Distributed Environment [C]//Proceeding of 2nd IEEE Workshop on Future Trends in Distributed Computing System. Cairo, Egypt: Amazon. co. uk, 1990.

[2] Osawa E. A Scheme for Agent Collaboration in Open Multi - Agent Environment[C]//Bajcsy R. Proceeding of IJCAI'93. Chambery, France: [s. n.], 1993:352 - 358.

[3] 杨建池,王运吉,黄柯棣. 一种 Agent 体系结构 A2FC 研究[J]. 系统仿真学报, 2008, 20(10):2560 - 2567.

[4] 张　健,曾广周. 一种满足弱概念的混合型 Agent 结构[J]. 计算机工程与应用, 2008, 44(17):6 - 9.

[5] John R, Graha M, Keith S, et al. DECAF - A: Flexible Multi Agent System Architecture [J]. Autonomous Agents and Multi - Agent Systems, 2003(7):7 - 27.

[6] Wooldridge M. Intelligent Agents: Theory and Practice[J]. Knowledge Engineering Review, 1995, 10(2), 115 - 152.

[7] Nwana H. Software Agent: an Overview[J]. Knowledge Engineering Review, 1997, 12(2):205 - 245.

[8] 张友军,朱森良,吴春明,等. 自主式移动机器人系统的体系结构[J]. 机器人, 1997, 19(5):378 - 383.

[9] Nilsson N. Logic and artificial intelligence[J]. Artificial Intelligence, 1991, 47(1):31 - 56.