XML 数据传输格式

李鸿岐

（lihongqi@nwpu.edu.cn）

2025/4/25

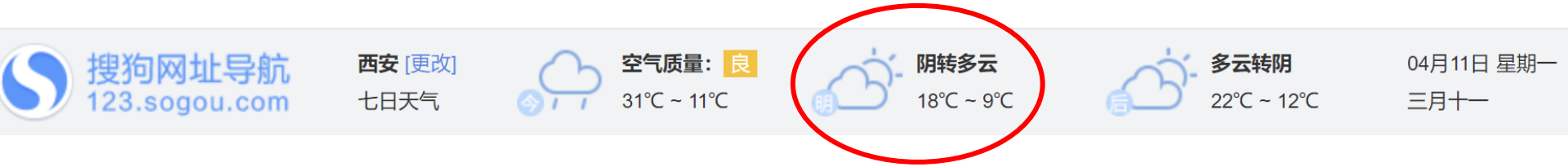
目录 | contents

目录 | contents

01-XML概述

西北工业大学 NORTHWESTERN POLYTECHNICAL UNIVERSITY

XML引入



- 数据一样，只是展现形式不同
- 数据来源：相关服务器（如：气象局数据库）
- 网页编程语言多样，所需的数据如何适用？
- 以哪种形式传递数据？
- **XML数据传输格式**

规范数据格式，使数据具有结构性，易读易处理

XML引入

什么是XML?

- 可扩展标记语言(eXtensible Markup Language)
- 是一种标记语言，很类似HTML
- XML被设计用来**传输**和**存储**数据，而非显示数据
- XML标签没有被预定义，需要自行定义标签
- 具有自我描述属性
- W3C的推荐标准

XML的例子

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--   Copyright w3school.com.cn   -->
<note>
    <to> George </to>
    <from> John </from>
    <heading> Reminder </heading>
    <body> Don't forget the meeting! </body>
</note>
```

- 具有自我描述属性。拥有标题以及留言，同时包含了发送者和接收者信息
- 仅仅是包装在XML标签中的纯粹的信息，没有做任何事情
- 需要编写软件或者程序才能发送、接收和显示该文档

XML的特点

XML是不作为的

- XML不会做任何事情，XML被设计用来**结构化**、**存储**以及**传输**信息

XML仅仅是纯文本

- 有能力处理纯文本的软件都可以处理XML
- 能够读懂XML的应用程序可以有针对性地处理XML的标签
- 标签的功能性意义依赖于应用程序的特性

通过XML可以发明自己的标签

- XML没有预定义的标签，可以定义自己的标签和文档结构

XML是独立于软件和硬件的信息传输工具

- XML不是对HTML替代，是补充
- XML用于传输数据，而HTML用于格式化并显示数据

01-XML概述-作用

XML的作用

- XML把数据从 HTML 分离
  - 数据能够存储在独立的 XML 文件，专注于使用 HTML 进行布局和显示
- XML简化数据共享
  - 在真实的世界中，计算机系统和数据使用不兼容的格式来存储数据
  - XML 数据以纯文本格式进行存储
  - 让创建不同应用程序可以共享的数据变得更加容易
- 简化数据传输、简化平台变更
  - 在不兼容的系统之间轻松地交换数据
- **用作配置文件** config.xml
- **存储数据**，充当小型数据库
- XML用于创建新的Internet语言

01-XML概述-文档组成

XML文档的组成

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>George</to>
<from>John</from>
<heading>Reminder</heading>
<body>Don't forget the meeting!</body>
</note>
```

XML编辑器：

- ✓ 记事本（.xml）
- ✓ Dreamweaver
- ✓ XMLSpy
- ✓ Eclipse

- 第一行是 XML 声明。
  它定义 XML 的版本 (1.0) 和所使用的编码 (ISO-8859-1 = Latin-1/西欧字符集)。
- 描述文档的根元素。
- 描述根的4个子元素（to、from、heading、body）。

01-XML概述-文档组成

西北工业大学 NORTHWESTERN POLYTECHNICAL UNIVERSITY

XML文档的声明

- XML文档开头的一些字符必须标记一个XML声明
- XML处理软件会根据声明来确定如何处理后面的内容

```
<?xml version="1.0" encoding="UTF-8"?>
```

XML元素

- XML元素指的是从(且包括)开始标签直到(且包括)结束标签的部分
- 根元素，XML文档最顶端的一个元素
- 元素的内容可以是字符数据、其他的嵌套元素，或二者的组合
- 包含在其他元素中的元素称为嵌套元素
- 包含在文档中的数据值称为文档的内容
- 不同类型的元素具有不同的名字，对于特定类型元素，XML并不提供表示这些类型元素的具体含义的方法，而是表示这些元素类型之间的关系

01-XML概述-文档组成

XML元素

元素可以拥有属性，在起始标签中添加属性

```
<bookstore>
<book category="CHILDREN">
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
<book category="WEB">
  <title>Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>
</bookstore>
```

- <bookstore> 和 <book> 都拥有元素内容，因为它们包含了其他元素。
- <author> 只有文本内容，因为它仅包含文本
- <book> 元素拥有属性 (category="CHILDREN")

元素可以扩展

- 向XML文档添加新的元素以表示额外的信息，不中断应用程序

01-XML概述-文档组成

XML文档的组成

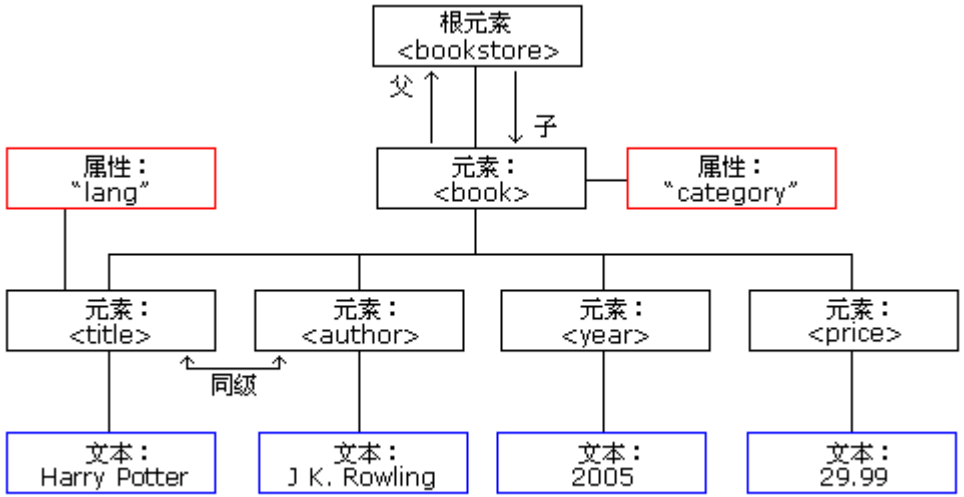
XML 文档形成了一种树结构，它从“根部”开始，然后扩展到“枝叶”

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

所有元素均可拥有子元素：

- XML文档**必须**包含**根元素**，该元素是其他所有元素的父元素
- XML文档中的元素形成了一棵文档树，从根部开始，并扩展到树的最底端
- 父子以及同胞等术语用于描述元素之间的关系：
  - 父元素拥有子元素
  - 相同层级上的子元素成为同胞（兄弟或姐妹）
- 所有元素均可拥有文本内容和属性

XML文档的组成



XML中的一本书

- 根元素：<bookstore>
- 所有 <book> 元素都被包含在 <bookstore> 中
- <book> 元素有 4 个子元素：<title>、< author>、<year>、<price>。

```
<bookstore>
<book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
</book>
<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
<book category="WEB">
  <title lang="en">Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>
</bookstore>
```

XML文档的组成


开始部分
<?xml version"1.0" encoding="UTF-8"?>
XML声明
<!- File Name: PurchaseOrder.xml -->
注释
根元素
<PurchaseOrder>
<Customer>
<Name> Clive James </Name>
<BillingAddress> .. </BillingAddress>
<ShippingAddress> .. </ShippingAddress>
<ShippingDate> 2004-09-22 </ShippingDate>
</Customer>
<Customer>
...
</Customer>
<Customer>
<Name> Julie Smith </Name>
<BillingAddress> .. </BillingAddress>
<ShippingAddress> .. </ShippingAddress>
<ShippingDate> 2004-12-12 </ShippingDate>
</Customer>
</PurchaseOrder>
根元素中的元素嵌套

目录 | contents

XML的语法规则

- XML文档头声明可有可无，但建议添加

语法正确！！！

```
<?xml version="1.0" encoding="UTF-8"?>
```

XML的语法规则

- 所有XML元素都必须有关闭标签
- XML标签对大小写敏感

<Message> 这是错误的。</message>

<message> 这是正确的。</message>

- XML必须正确地嵌套，标签不允许交叉

在哪个元素内打开，就在哪个元素内关闭

  嵌套示例

- XML文档必须有根元素
- XML的属性值必须加引号，可以是单引号也可以是双引号

<note data = “2022.1.1” </note>

XML的语法规则

XML实体引用

- 在 XML 中，一些字符拥有特殊的意义。

<message> if salary < 1000 then /message>　　错误！！！

- 用实体引用来代替特殊字符

<message> if salary < 1000 then /message>

- XML中，有五个预定义的实体引用

<	<	小于
>	>	大于
&	&	和号
'	'	单引号
"	"	引号

- 在 XML 中，只有字符 "<" 和 "&" 确实是非法的。
- “>” 是合法的，但是用实体引用来代替它是一个好习惯。

XML的语法规则

XML中的注释

```
<!--This is a comment -->
```

- 注释以<!-- 开始，以-->结束
- 注释不能出现在声明之前
- 注释不能出现在属性值中
- 注释不能嵌套在其他注释中

在XML中，空格会被保留

- HTML: Hello my name is David.

  输出: Hello my name is David.

- 在 XML 中，文档中的空格不会被删节。

XML以LF(换行符)存储换行

XML的语法规则

XML元素必须遵循以下命名规则：

- 名称可以含字母、数字以及其他的字符
- 名称不能以数字或者标点符号开始
- 名称不能以字符 “xml”（或者 XML、Xml）开始
- 名称不能包含空格
- 可使用任何名称，没有保留的字词
- 最佳命名习惯

  使名称具有描述性

  名称应当比较简短

  避免使用 ” -”,“ .”,“ :” 字符

XML的语法规则

- 属性 (Attribute) 提供关于元素的额外（附加）信息
- 属性是通过与元素关联的名—值对来表示的

- 其中值必须包括在单引号或双引号中

```
<person sex="female">
<person sex='female'>
```

正确！！！

- 如果属性值本身包含双引号，那么有必要使用单引号包围它

```
<gangster name='George "Shotgun" Ziegler'>
```

或者可以使用实体引用：

```
<gangster name="George &quot; Shotgun &quot; Ziegler">
```

XML的语法规则

- 在XML中，尽量使用元素来描述数据
- 避免 XML 属性？仅仅使用属性来提供与数据无关的信息

因使用属性而引起的一些问题：

- 属性无法包含多重的值（元素可以）
- 属性无法描述树结构（元素可以）
- 属性不易扩展（为未来的变化）
- 属性难以阅读和维护，解析属性会带来额外的代码

```
<note day="08" month="08" year="2008"
to="George" from="John"
heading="Reminder"  body="Don't forget the
meeting!">
 </note>
```

不要做这样的蠢事！！！

XML的语法规则

- CDATA
- CDATA 的意思是字符数据（character data）。
- CDATA 是不会被解析器解析的文本。

- 在这些文本中的标签不会被当作标记来对待，其中的实体也不会被展开。

<![CDATA[…不解析的内容…]]>

This XML file does not appear to hav

```
▼<note>
    <to>George</to>
    <from>John</from>
    <heading>Reminder</heading>
    <body>Don't forget the meeting!</body>
  ▼<knowledge>
      <![CDATA[ 1<2 ]]>
    </knowledge>
  </note>
```

注：

- 特殊字符较少时，使用实体替换；
- 较多时使用CDATA

目录 | contents

良构的XML

拥有正确语法的 XML 被称为“形式良好”（Well Formed）的 XML。

- XML 声明语句
- XML 文档必须有根元素，且只有一个
- XML 文档必须有关闭标签
- XML 标签对大小写敏感
- XML 元素必须被正确的嵌套
- XML 属性必须加引号

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>George</to>
<from>John</from>
<heading>Reminder</heading>
<body>Don't forget the meeting!</body>
</note>
```

有效的XML

有效的XML 文档是“形式良好”的 XML 文档，同样遵守**文档类型定义** (Document Type Definition, DTD) 的语法规则：

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
<book>
  <title>Harry Potter</title>
</book>
<书>
  <书名>Learning XML</书名>
</书>
</bookstore>
```

格式过于灵活

较乱

解析提取信息时存在问题

DTD可定义合法的XML文档构建模块。它使用一系列合法的元素来定义文档的结构。

DTD 可被成行地声明于 XML 文档中，也可作为一个外部引用。

内部的DOCTYPE声明

```
<!DOCTYPE 根元素 [元素声明]>
```

元素声明语法：

```
<! ELEMENT 元素名（子元素, 子元素…）>
```

数量词：

- + ： 表示出现1次或多次，至少1次
- ？： 表示出现0次或1次
- * ： 表示出现任意次

属性声明语法：

```
<!ATTLIST 元素名称 属性名称 属性类型 默认值>
```

- 属性类型： CDATA，表示字符数据（character data）
- 默认值： #REQUIRED 属性值是必需的
  #IMPLIED 属性值不是必需的

内部的DOCTYPE声明

- PCDATA
- PCDATA 的意思是被解析的字符数据（parsed character data）。
- 可把字符数据想象为 XML 元素的开始标签与结束标签之间的文本。
  - ： 用作只写文本，没有别的子标签
- PCDATA 是会被解析器解析的文本。这些文本将被解析器检查实体以及标记

内部的DOCTYPE声明

<!DOCTYPE 根元素 [元素声明]>

```
<?xml version="1.0"?>
<!DOCTYPE note [
  <!ELEMENT note (to,from,heading,body)>
  <!ELEMENT to      (#PCDATA)>
  <!ELEMENT from    (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body    (#PCDATA)>
]>
<note>
  <to>George</to>
  <from>John</from>
  <heading>Reminder</heading>
  <body>Don't forget the meeting!</body>
</note>
```

!DOCTYPE note 定义此文档是 **note** 类型的文档。

!ELEMENT note 定义 **note** 元素有四个元素：“to、from、heading,、body”

!ELEMENT to 定义 **to** 元素为 "#PCDATA" 类型

!ELEMENT from 定义 **from** 元素为 "#PCDATA" 类型

!ELEMENT heading 定义 **heading** 元素为 "#PCDATA" 类型

!ELEMENT body 定义 **body** 元素为 "#PCDATA" 类型

外部的DOCTYPE声明

```
<!DOCTYPE 根元素 SYSTEM "文件名">
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
<to>George</to>
<from>John</from>
<heading>Reminder</heading>
<body>Don't forget the meeting!</body>
</note>
```

Note.dtd

```
<!DOCTYPE note [
  <!ELEMENT note (to,from,heading,body)>
  <!ELEMENT to      (#PCDATA)>
  <!ELEMENT from    (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body    (#PCDATA)>
]>
```

总结：

为什么使用 DTD？

- 通过 DTD，每一个 XML 文件均可携带一个有关其自身格式的描述。
- 通过 DTD，独立的团体可一致地使用某个标准的 DTD 来交换数据。
- 应用程序也可使用某个标准的 DTD 来验证从外部接收到的数据
- 使用 DTD 来验证自身的数据。

XML Schema ：

- XML Schema 语言也称作 XML Schema 定义（XML Schema Definition，XSD）。
- XML Schema 是DTD的替代者
- 不仅可以定义XML文档的结构，还可以规范文档的内容
- XSD本身也是XML文档
- XSD采用XML文档来定义语义约束，比DTD要复杂一些，但功能更强大
  - 支持丰富的数据类型
  - 允许开发者自定义数据类型
  - 可读性强
  - 可针对未来需求进行扩展

03-良构与有效的XML

XML Schema：

- 定义可出现在文档中的元素
- 定义可出现在文档中的属性
- 定义哪个元素是子元素
- 定义子元素的次序
- 定义子元素的数目
- 定义元素是否为空，或者是否可包含文本
- 定义元素和属性的数据类型
- 定义元素和属性的默认值以及固定值

目录 | contents

XML 的解析：

对XML文件进行操作，包括创建XML，对XML文件进行增、删、改、查操作

常见的XML解析技术:

DOM解析

- 官方提供的解析方式，基于XML树解析的
- 比较耗资源
- 适用于多次访问XML

SAX解析

- 民间的解析方式，基于事件
- 消耗资源小
- 适用于数据量较大的XML

04-XML解析

XML 的解析： 常见的XML解析技术:

- JDOM 解析
  - 第三方提供，开源免费的解析方式
  - 比DOM解析更快
  - JDOM仅使用具体类而不使用接口
- DOM4J 解析（DOM for Java）
  - 第三方提供，开源免费的解析方式
  - 非常优秀的Java XML API，性能优异、功能强大
  - JDOM的升级版
  - 使用接口而不是实现类
- PHP 解析
  - 在PHP 5版本以后，其提供了一个非常强大的类库，Simple XML类库，专门用于实现对XML文档的解析操作

PHP解析XML：

- Bookstore.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
<book category="CHILDREN">
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
<book category="WEB">
  <title>Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>
</bookstore>
```

- Visual Studio Code
- 配置PHP开发环境

  https://www.jb51.net/article/210085.htm

```
<?php
//simplexml_load_file 解析xml文档，返回php对象
$x = simplexml_load_file('Bookstore.xml');
var_dump ($x);  //打印
```

04-XML解析-PHP解析

Bookstore.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
<book category="CHILDREN">
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
<book category="WEB">
  <title>Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>
</bookstore>
```

```
<?php
$x = simplexml_load_file('Bookstore.xml');
var_dump ($x);
```

问题 输出 调试控制台 终端

```
object(SimpleXMLElement)#1 (1) {
  ["book"]=>
  array(2) {
    [0]=>
    object(SimpleXMLElement)#2 (5) {
      ["@attributes"]=>
      array(1) {
        ["category"]=>
        string(8) "CHILDREN"
      }
      ["title"]=>
      string(12) "Harry Potter"
      ["author"]=>
      string(12) "J K. Rowling"
      ["year"]=>
      string(4) "2005"
      ["price"]=>
      string(5) "29.99"
    }
    [1]=>
    object(SimpleXMLElement)#3 (5) {
      ["@attributes"]=>
      array(1) {
        ["category"]=>
        string(3) "WEB"
      }
      ["title"]=>
      string(12) "Learning XML"
      ["author"]=>
      string(11) "Erik T. Ray"
      ["year"]=>
      string(4) "2003"
      ["price"]=>
      string(5) "39.95"
    }
  }
}
```

04-XML解析-PHP解析

Php调用simplexml_load_file函数，解析xml生成的对象

Php将XML节点以属性的形式存放

内容若有多条，以数组的形式存放

数组的值就是解析后的节点名字和内容，以对象属性的形式存放

```
<?php
$x = simplexml_load_file('Bookstore.xml');
var_dump ($x);
```

问题 输出 调试控制台 终端

```
object(SimpleXMLElement)#1 (1) {
  ["book"]=>
  array(2) {
    [0]=>
    object(SimpleXMLElement)#2 (5) {
      ["@attributes"]=>
      array(1) {
        ["category"]=>
        string(8) "CHILDREN"
      }
      ["title"]=>
      string(12) "Harry Potter"
      ["author"]=>
      string(12) "J K. Rowling"
      ["year"]=>
      string(4) "2005"
      ["price"]=>
      string(5) "29.99"
    }
    [1]=>
    object(SimpleXMLElement)#3 (5) {
      ["@attributes"]=>
      array(1) {
        ["category"]=>
        string(3) "WEB"
      }
      ["title"]=>
      string(12) "Learning XML"
      ["author"]=>
      string(11) "Erik T. Ray"
      ["year"]=>
      string(4) "2003"
      ["price"]=>
      string(5) "39.95"
    }
  }
}
```

04-XML解析-PHP解析

```
<?php
$x = simplexml_load_file('Bookstore.xml');
//var_dump ($x);

echo $x->book[0]->title;
```

Bookstore_simplexml.php

```
1  <?php
2  //simplexml_load_file 解析xml文档，返回php对象
3  $x = simplexml_load_file('Bookstore.xml');
4
5  //var_dump ($x);
6
7  echo $x->book[0]->title;
```

问题 输出 调试控制台 终端

```
PHP:  syntax error, unexpected '(' in D:\Program F
Harry Potter
```

问题 输出 调试控制台 终端

```
object(SimpleXMLElement)#1 (1) {
  ["book"]=>
  array(2) {
    [0]=>
    object(SimpleXMLElement)#2 (5) {
      ["@attributes"]=>
      array(1) {
        ["category"]=>
        string(8) "CHILDREN"
      }
      ["title"]=>
      string(12) "Harry Potter"
      ["author"]=>
      string(12) "J K. Rowling"
      ["year"]=>
      string(4) "2005"
      ["price"]=>
      string(5) "29.99"
    }
    [1]=>
    object(SimpleXMLElement)#3 (5) {
      ["@attributes"]=>
      array(1) {
        ["category"]=>
        string(3) "WEB"
      }
      ["title"]=>
      string(12) "Learning XML"
      ["author"]=>
      string(11) "Erik T. Ray"
      ["year"]=>
      string(4) "2003"
      ["price"]=>
      string(5) "39.95"
    }
  }
}
```

04-XML解析-PHP解析


```php
<?php
$x = simplexml_load_file('Bookstore.xml');

//遍历xml
foreach($x->book as $v){
    echo $v->title;
}
```

Bookstore_simplexml.php

```
1   <?php
2   //simplexml_load_file 解析xml文档，返回php对象
3   $x = simplexml_load_file('Bookstore.xml');
4
5   //var_dump ($x);
6
7   //echo $x->book[0]->title;
8
9   //遍历xml
10  foreach($x->book as $v){
11      echo $v->title;
12  }
```

问题 输出 调试控制台 终端

```
PHP:  syntax error, unexpected '(' in D:\Program Fi
Harry PotterLearning XML
```

```php
<?php
$x = simplexml_load_file('Bookstore.xml');

$c = count($x->book);
for ($i=0;$i<$c;$i++){
    echo $x->book[$i]->author;
}
```

Bookstore_simplexml.php

```
1   <?php
2   //simplexml_load_file 解析xml文档，返回php对象
3   $x = simplexml_load_file('Bookstore.xml');
4
5   //var_dump ($x);
6
7   //echo $x->book[0]->title;
8
9   //遍历xml
10  //foreach($x->book as $v){
11  //    echo $v->title;
12  //}
13
14  $c = count($x->book);
15  for ($i=0;$i<$c;$i++){
16      echo $x->book[$i]->author;
17  }
18
```

问题 输出 调试控制台 终端

```
PHP:  syntax error, unexpected '(' in D:\Program Files (
J K. RowlingErik T. Ray
```

04-XML解析-PHP解析

```
<?php
//simplexml_load_file 解析xml文档，返回php对象
$x = simplexml_load_file('Bookstore.xml');

$book = $x->addchild('book');

$book->addchild('title','Java Learning');
var_dump($x);
```

Bookstore_addchild.php

```
<?php
//simplexml_load_file 解析xml文档，返回php对象
$x = simplexml_load_file('Bookstore.xml');

$book = $x->addchild('book');

$book->addchild('title','Java Learning');
var_dump($x);

$x->asXML('Bookstore_addchild.xml');
```

```
object(SimpleXMLElement)#1 (1) {
  ["book"]=>
  array(3) {
    [0]=>
    object(SimpleXMLElement)#3 (5) {
      ["@attributes"]=>
      array(1) {
        ["category"]=>
        string(8) "CHILDREN"
      }
      ["title"]=>
      string(12) "Harry Potter"
      ["author"]=>
      string(12) "J K. Rowling"
      ["year"]=>
      string(4) "2005"
      ["price"]=>
      string(5) "29.99"
    }
    [1]=>
    object(SimpleXMLElement)#4 (5) {
      ["@attributes"]=>
      array(1) {
        ["category"]=>
        string(3) "WEB"
      }
      ["title"]=>
      string(12) "Learning XML"
      ["author"]=>
      string(11) "Erik T. Ray"
      ["year"]=>
      string(4) "2003"
      ["price"]=>
      string(5) "39.95"
    }
    [2]=>
    object(SimpleXMLElement)#5 (1) {
      ["title"]=>
      string(13) "Java Learning"
    }
  }
}
```

04-XML解析-PHP解析

Bookstore_addchild.xml

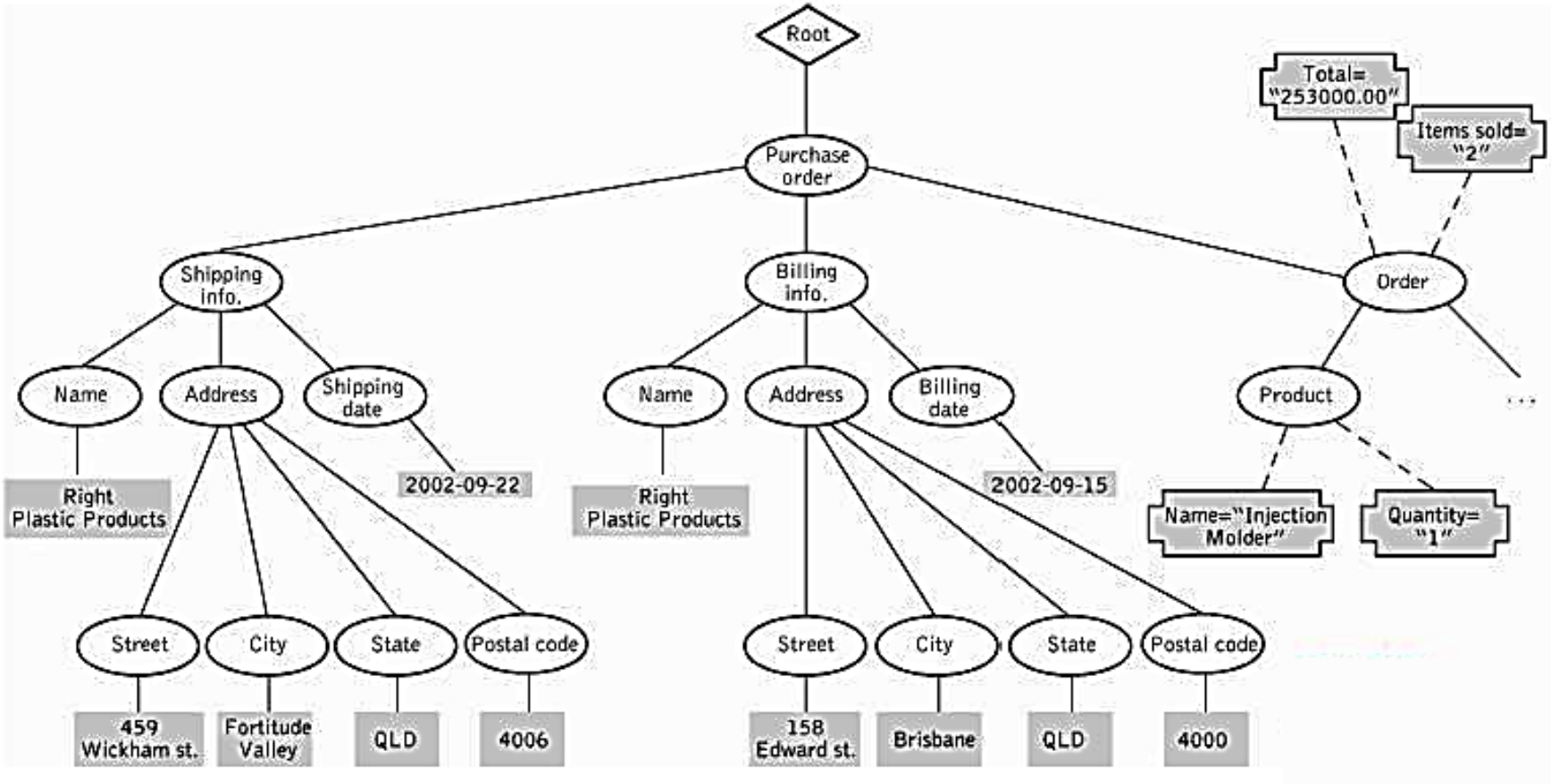
```xml
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
<book category="CHILDREN">
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
<book category="WEB">
  <title>Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>
<book><title>Java Learning</title></book></bookstore>
```

目录 | contents

XML路径语言Xpath概述

- Xpath是一门专门用来查找XML数据内容的语言；
- Xpath用来在XML文档中对元素及属性进行遍历；
- Xpath数据模型将文档视为一棵节点树；
  - 节点对应于文档组件，如元素、属性等
- Xpath使用一个系统性的分类来描述XML文档的层次标记，以及对孩子、后代、父母和祖先的引用
  - 父母是一个包含其他元素的元素
  - 祖先的元素列表包含它的父母
  - 后代列表包含了元素的孩子
  - 根或文档根，是一个容纳所有XML文档的逻辑构成

Xpath的树模型


Root
Purchase order
Total= "253000.00"
Items sold= "2"
Shipping info.
Billing info.
Order
Name
Address
Shipping date
Name
Address
Billing date
Product
...
Right Plastic Products
2002-09-22
Right Plastic Products
2002-09-15
Name="Injection Molder"
Quantity= "1"
Street
City
State
Postal code
Street
City
State
Postal code
459 Wickham st.
Fortitude Valley
QLD
4006
158 Edward st.
Brisbane
QLD
4000

05-Xpath语言

西北工业大学 NORTHWESTERN POLYTECHNICAL UNIVERSITY

Xpath语言概述

Bookstore_xpath.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
<book category="CHILDREN">
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
<book category="WEB">
  <title>Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>
<book>
<title>Java Learning</title>
</book>
</bookstore>
```

Bookstore_xpath.xml Bookstore_xpath.php

Bookstore_xpath.php

```
1  <?php
2  //simplexml_load_file 解析xml文档，返回php对象
3  $x = simplexml_load_file('Bookstore_xpath.xml');
4
5  $book = $x->xpath('/bookstore/book/title');
6
7  var_dump($book);
```

问题 输出 调试控制台 终端

```
PHP:  syntax error, unexpected '(' in D:\Program Files (x8
array(3) {
  [0]=>
  object(SimpleXMLElement)#2 (1) {
    [0]=>
    string(12) "Harry Potter"
  }
  [1]=>
  object(SimpleXMLElement)#3 (1) {
    [0]=>
    string(12) "Learning XML"
  }
  [2]=>
  object(SimpleXMLElement)#4 (1) {
    [0]=>
    string(13) "Java Learning"
  }
}
```

Xpath语言概述

Bookstore_xpath.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
<book category="CHILDREN">
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
<book category="WEB">
  <title>Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>
<book>
<title>Java Learning</title>
</book>
</bookstore>
```

Bookstore_xpath.php

```php
<?php
//simplexml_load_file 解析xml文档，返回php对象
$x = simplexml_load_file('Bookstore_xpath.xml');

$book = $x->xpath('/bookstore/book/title');

//var_dump($book);

//xpath 查找后返回 数组，数组中的值仍然是个 对象
//循环时，如果数组的值为对象，则可直接得到其值
foreach($book as $v){
    echo $v;
}
```

问题 输出 调试控制台 终端

```
PHP:  syntax error, unexpected '(' in D:\Program Files
Harry PotterLearning XMLJava Learning
```

//参数为路径，以/开始的为**绝对路径查找**

Xpath语言概述

Bookstore_xpath.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
<book category="CHILDREN">
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
<book category="WEB">
  <title>Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>
<book>
<title>Java Learning</title>
</book>
</bookstore>
```

```
17  //相对路径查找
18  $book = $x->xpath('//title');
19
20  var_dump($book);
21
```

问题 输出 调试控制台 终端

```
PHP:  syntax error, unexpected '(' in D:\Program Fi
array(3) {
  [0]=>
  object(SimpleXMLElement)#2 (1) {
    [0]=>
    string(12) "Harry Potter"
  }
  [1]=>
  object(SimpleXMLElement)#3 (1) {
    [0]=>
    string(12) "Learning XML"
  }
  [2]=>
  object(SimpleXMLElement)#4 (1) {
    [0]=>
    string(13) "Java Learning"
  }
}
```

//相对路径查找

Xpath语言概述

Bookstore_xpath.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
<book category="CHILDREN">
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
<book category="WEB">
  <title>Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>
<book>
<title>Java Learning</title>
</book>
</bookstore>
```

//使用*匹配所有节点

```
25  $book = $x->xpath('//book/*');
26
27  var_dump($book);
28
```

问题 输出 调试控制台 终端

```
array(9) {
  [0]=>
  object(SimpleXMLElement)#2 (1) {
    [0]=>
    string(12) "Harry Potter"
  }
  [1]=>
  object(SimpleXMLElement)#3 (1) {
    [0]=>
    string(12) "J K. Rowling"
  }
  [2]=>
  object(SimpleXMLElement)#4 (1) {
    [0]=>
    string(4) "2005"
  }
  [3]=>
  object(SimpleXMLElement)#5 (1) {
    [0]=>
    string(5) "29.99"
  }
  [4]=>
  object(SimpleXMLElement)#6 (1) {
    [0]=>
    string(12) "Learning XML"
  }
  [5]=>
  object(SimpleXMLElement)#7 (1) {
    [0]=>
    string(11) "Erik T. Ray"
  }
```

Xpath语言概述

Bookstore_xpath.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
<book category="CHILDREN">
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
<book category="WEB">
  <title>Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>
<book>
<title>Java Learning</title>
</book>
</bookstore>
```

//条件查找

判断

```
28  $book = $x->xpath('//book[price > 30]');
29
30  var_dump($book);
31
```

问题 输出 调试控制台 终端

```
PHP:  syntax error, unexpected '(' in D:\Program
array(1) {
  [0]=>
  object(SimpleXMLElement)#2 (5) {
    ["@attributes"]=>
    array(1) {
      ["category"]=>
      string(3) "WEB"
    }
    ["title"]=>
    string(12) "Learning XML"
    ["author"]=>
    string(11) "Erik T. Ray"
    ["year"]=>
    string(4) "2003"
    ["price"]=>
    string(5) "39.95"
  }
}
```

Xpath语言概述

Bookstore_xpath.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
<book category="CHILDREN">
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
<book category="WEB">
  <title>Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>
<book>
<title>Java Learning</title>
</book>
</bookstore>
```

//条件查找

Bookstore_xpath.php

```
26  //var_dump($book);
27
28  $book = $x->xpath('//book[price > 30]');
29
30  //var_dump($book);
31
32  foreach($book as $v){
33      echo $v->title;
34  }
```

问题 输出 调试控制台 终端

```
PHP:  syntax error, unexpected '(' in D:\Progr
Learning XML
```

```
36  $book = $x->xpath('//book[last()]');
37
38  foreach($book as $v){
39      echo $v->title;
40  }
41
```

问题 输出 调试控制台 终端

```
PHP:  syntax error, unexpected '(' in D:\Pr
Java Learning
```

目录 | contents

1. XML是一个可扩展的标记语言
2. XML被设计用来传输和存储数据，而非显示数据
3. XML文档必须包含根元素，标签需成对、自定义
4. XML中的实体替换
5. CDATA <![CDATA[…不解析的内容…]]>
6. 良构与有效的XML (DTD)
7. XML解析
8. Xpath 路径查找

敬请批评指正！

软件学院 李鸿岐