# 第二讲 PHP操作XML

# 回顾

1、XML介绍

1）XML基于文本格式

2）标签没有预定义，全部都是自定义,XML标签是没有特殊含义。

3）用于保存或传输结构化的数据

4）是元语言

5）是一种树形结构，从树根开始，扩展到树枝，树叶

6）XML用来保存数据，不是用来显示数据。

2、XML的规则

1）必须，只能有一个根。

2）XML区分大小写

3）XML不允许空标记的存在，所有的标记必须是容器标记

4）XML必须有正确的嵌套 <b><i></i></b>

5）标签命名规则：以字母、下划线、冒号开头，后面跟的是字母，数字、下划线、冒号、句号、连字符(中划线)。

3、XML的注释

1）、<!-- 注释的内容 -->

2）、注释不能放在声明的前面

3）、注释不能嵌套

4、XML数据的分类

1）PCDATA （parse character data ,可解析的字符数据）

2）CDATA (character data,字符数据，XML解析器不解析)

tip:元素都是PCDATA,属性都是CDATA。

5、XML声明

1）、XML的声明必须放在第一行

2）<?xml version="1.0" encoding="utf-8"?>

3）version：版本号，默认都是1.0

4）encoding：字符编码，有utf-8或gb2312

6、显示CDATA数据

<![CDATA[

cdata数据

]]>

7、实体

1）实体是一个存储单元，就类似于常亮。

2）实体分为预定义实体和自定义实体

3）预定义实体：

” &quot; ' &apos; < &lt; > &gt; & &amp;

4）、自定义实体分为一般实体和参数实体

5）、一般实体的声明 <!ENTITY 实体名 值>

6）、实体的调用 &实体名;

7）、参数实体在是一般实体名称前加一个%。

8、元素的声明

<!ELEMENT 元素的名称 (子元素1,子元素2...)>

+ >=1

\* >=0

? =0 =1

9、属性的声明

|  |  |
| --- | --- |
| 名称 | 语法 |
| 必填 | <!ATTLIST 元素名称 属性名称 CDATA #REQUIRED> |
| 隐含 | <!ATTLIST 元素名称 属性名称 CDATA #IMPLIED> |
| 默认值 | <!ATTLIST 元素名称 属性名称 CDATA "默认值"> |
| 固定值 | <!ATTLIST 元素名称 属性名称 CDATA #FIXED> |
| 枚举 | <!ATTLIST 元素名称 属性名称 (枚举1|枚举2|枚举3...) "默认枚举值"> |
| 唯一 | <!ATTLIST 元素名称 属性名称 ID #REQUIRED > |

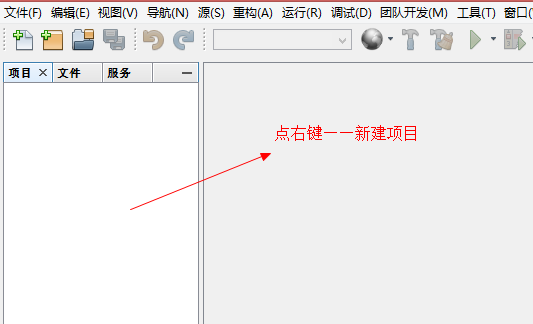
10、DTD是用于定义XML的规范。

11、要让一个XML文档被解析，最低满足标准是：格式良好

12、命名空间：DTD不支持命名空间。

# Netbeans

1、



2、



3、



# 例题：PHP简单读取XML



# DOM

document object model，文档对象模型，就是树形结构

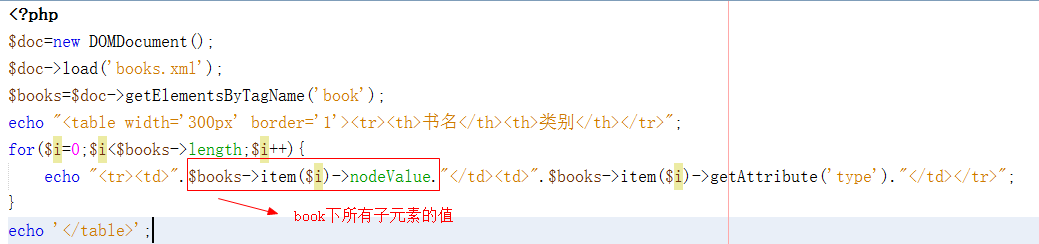
节点的分类

1. 元素节点
2. 属性节点
3. 值节点
4. 注释节点
5. 根节点（document才是根节点）

# 生成XML文档



# 读取XML的值



# 递归遍历XML

nodeType的值

|  |  |
| --- | --- |
| 值 | 节点 |
| 1 | 元素节点 |
| 2 | 属性节点 |
| 3 | 值节点 |
| 9 | 根节点 |

值节点的名称是“#text”，根节点的名称是“#document”

|  |
| --- |
| <?php  //递归XML文档  function getChilds($node){  echo "<ul>";  if($node->nodeType==3){  echo "<li>".$node->nodeValue."</li>"; //取出节点的值  }  else  {  echo "<li>".$node->nodeName."</li>"; //取出元素节点的名称  if($node->attributes->length>0) //节点的属性的个数大于0，表示此节点有属性  {  foreach ($node->attributes as $attr){  echo "<li>".$attr->value."</li>"; //取出节点的值  }  }  foreach($node->childNodes as $child){ //便利循环子元素  getChilds($child); //递归点  }  }  echo "</ul>";  }  $doc=new DOMDocument();  $doc->preserveWhiteSpace=false; //不保护空格，就是导入XML的时候去掉空白字符  $doc->load('books.xml');  $root=$doc->documentElement; //获得最顶层元素  getChilds($root); |

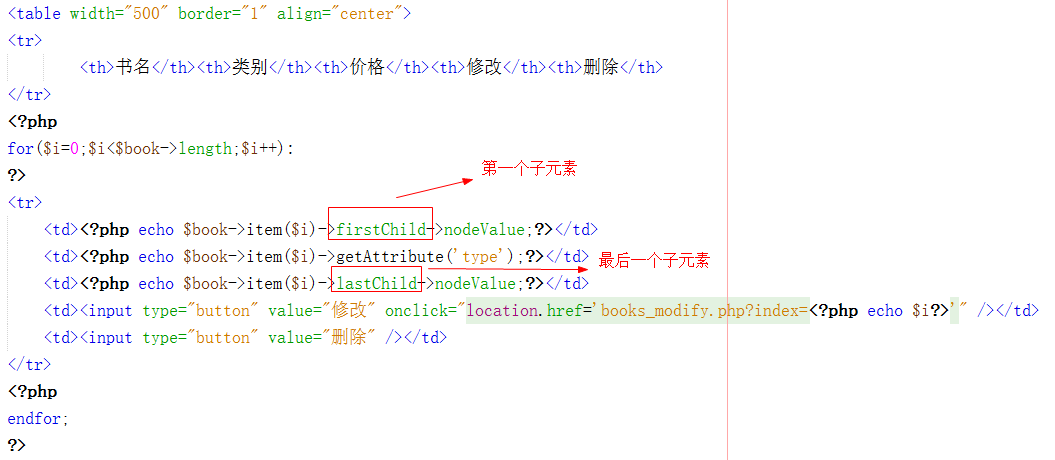
# 综合例题（对图书的操作）

1、管理界面

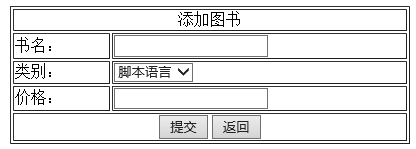
（1）通过ChildNodes属性来获取所有子节点，然后再取子节点的值。



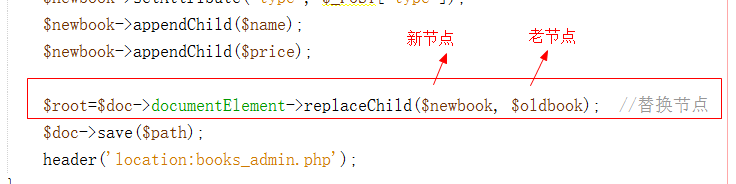
（2）



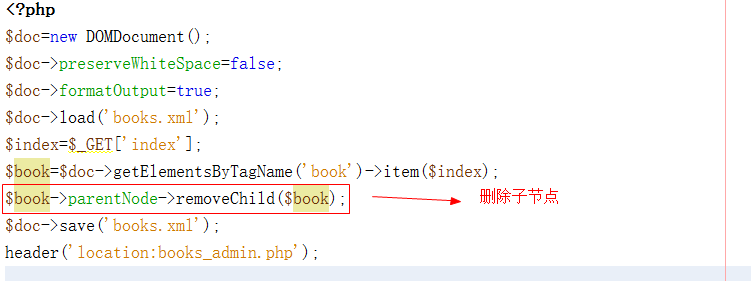
2、添加界面



3、修改



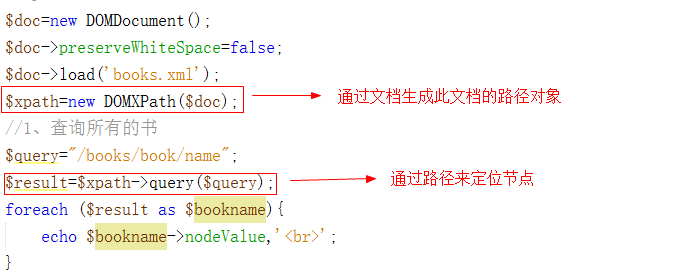
4、删除



parentNode:元素的父节点

# XPath

通过DOM结构定位节点，在数据量很大的情况下速度下降的很厉害。解决方法是XPath。Xpath的作用：用于快速定位节点

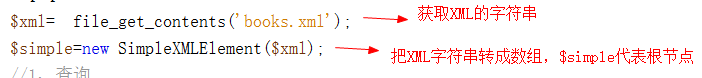


position()是节点的位置，节点的位置是从1开始。

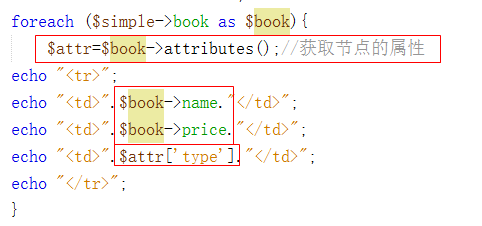
# simpleXML

PHP5以上提供了一个simpleXML对象来操作XML。把XML的节点转换成对象和数组去操作。

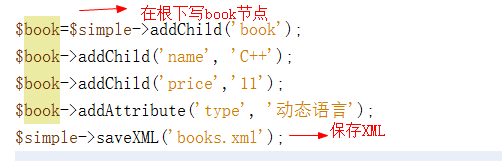
思路：先讲xml字符串转化成对象及数组。



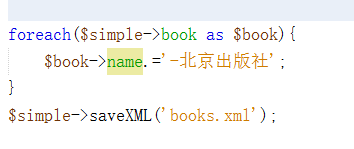
查询



2、添加



3、修改



4、删除

