XML

第一章 为什么需要XML

***1.1 html一些缺点:***

1.标签是规定好的.无法自定义

2.html文件显示样式和数据绑定在一起,数据无法复用

3.html标签是没有含义的

***1.2 XML的作用***

1. XML可以作为程序间通讯的标准(AJAX使用的 text xml json)

2.可以作为配置文件使用(php中用的很多)

如果在php中连接数据库

<?php

$conn = new mysqli (‘localhost’,’root’,’6217512’,’aaa’);

//php可以操作XML文件

?>

配置文件

<dbinfo>

<host>123.44.55.63</host>

<user>fangyuan</user>

<type>mysql</type>

<password>123456</password>

</dbinfo>

3. 可以当作小型数据库

在有些数据放在数据库中代价比较高,这样就可以考虑直接放在文件中

***1.3 XML的语法***

***1.3.1一个XML文件有以下几个部分组成***

* + - 文档声明
      * <?xml version= ‘1.0’ encoding=’utf-8’ standalone=’yes’?>
      * version -->--> 文档符合XML1.0规范(还有1.4,2.0)
      * encoding-->--> 文档字符编码,比如’utf-8’
      * standalone-->--> 文档定义是否独立使用
        + standalone=’yes’
        + standalone=’no’ 默认
    - 元素
      * 1.每个XML文档必须有且只有一个根元素
        + 根元素是一个完全包括文档中其它所有元素的元素
        + 根元素的起始标记要放在所有其它元素的起始标记之前
        + 根元素的结束标记要放在所有其它元素的结束标记之后
      * 2.XML元素指XML文件中的标签,一个标签分为开始标签与结束标签,一个标签有如下几种书写形式
        + 包含标签体: <a>www.aaa.com</a>
        + 不包含标签体: <a></a> 简写为<a/>
      * 3.一个标签中也可以嵌套若干子标签,但所有标签必须合理嵌套,不能交叉嵌套. 比如:
        + <a>hello jack<> <b>www.aaa.com</a></b> 这是错误的
      * 对于XML标签中的**所有空格与换行**,php的XML解析器当作标签内容处理. 比如

这个<stu>xiaoming</stu> 和

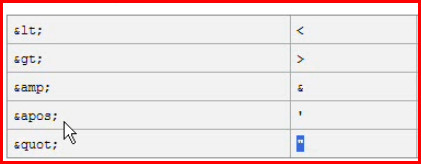
这个<stu>

xiaoming

<stu>

是意义不同的,在编写XML文档时要特别小心

* + - * 一个XML标签可以包括字母,数字,及其它一些可见字符,但是必须遵守下面的规范:
        + 区分大小写,比如,<P>与<p>是两个不同的标签
        + 不能以数字或\_下划线开头
        + 不能以xml或(XML,Xml等)开头
        + 不能包含空格
        + 名称中间不能含有冒号:
      * 标签中的属性值注意事项
        + 属性值用双引号”或单引号’分隔
        + 属性值中有单引号又有双引号,可以使用实体字符



<stu stuNo='a0&apos;0&apos;1'>

* + - * + 一个元素可以有多个属性,基本格式:

<标签名(元素) 属性名1=’属性值1’ 属性名2=’属性值2’ ...>

但是特定的属性名只能在同一个标签(元素)中出现一次



<student ID=’100’>

<name>tom</name>

</student>

* + - * 属性值不能是< > &



* + - * 注释细节
        + 注释的形式<!—注释内容-->
        + 注释不要嵌套



* + - * 可以在除标签(元素)意外的任何地方注释
    - CDATA区,特殊字符
      * XML文件中某个标签内容需要包含特殊字符则使用CDATA

比如<简介><![CDATA[sfasf$%<><>>]]></简介>

<头像><![CDATA[$^&\*()<>>]]></头像>

* + - * + CDATA的基本语法

***<![CDATA[***内容***]]>***

如何使用xml文件传递,保存图片

可以先使用file\_get\_contents()或file()或Curl函数来获取网页的内容,然后将需要的图片内容保存到xml文件的CDATA区

* + - 处理指令(processing,instruction)

<?xml version='1.0' encoding='utf-8' standalone='yes'?>

<!--XML文件也可以用css来修饰,使用pi指令-->

<?xml-stylesheet type="text/css" href="test.css"?>

<班级>

<stu stuNo='a0&apos;0&apos;1'>

<name>孙悟空</name>

<sex>男</sex>

<age>500</age>

<简介><![CDATA[sfasf$%<><>>]]></简介>

<头像><![CDATA[$^&\*()<>>]]></头像>

</stu>

<stu stuNo='a002'>

<name>白骨精</name>

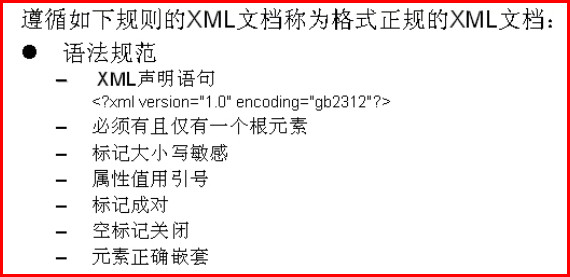
<sex>女</sex>

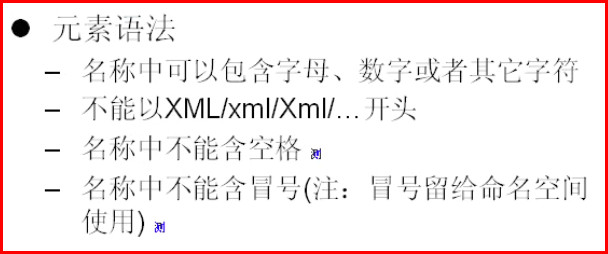
<age>200</age>

</stu>

</班级>

**格式正规的XML语法规范**

****

****

***1.3.1 快速入门***

用XML来描叙一个班级信息

<?xml version='1.0' encoding='utf-8' standalone='yes'?>

<?xml-stylesheet type="text/css" href="test.css"?>

<班级>

<stu stuNo='a0&apos;0&apos;1'>

<name>孙悟空</name>

<sex>男</sex>

<age>500</age>

<简介><![CDATA[sfasf$%<><>>]]></简介>

<头像><![CDATA[$^&\*()<>>]]></头像>

</stu>

<stu stuNo='a002'>

<name>白骨精</name>

<sex>女</sex>

<age>200</age>

</stu>

</班级>

* 第二章 DTD(document type definition) 文档类型定义

***2.1 基本介绍***

在xml技术里,可以**编写一个文档来约束一个XML文档**的书写规范,称为XML约束

而DTD便是一种约束技术,常用的还有Schema,他们可以用来约束html,xhtml

***2.2 快速入门***

class.xml文件代码

<?xml version='1.0' encoding='utf-8'?>

**<!DOCTYPE 班级 SYSTEM "myClass.dtd">**

<班级>

<学生>

<名字>周星驰</名字>

<年龄>23</年龄>

<介绍>未来的喜剧天王</介绍>

<面积>22</面积>

</学生>

<学生>

<名字>林青霞</名字>

<年龄>29</年龄>

<介绍>未来的好演员</介绍>

</学生>

</班级>

myClass.dtd代码

<!ELEMENT 班级 (学生+)>

<!ELEMENT 学生 (名字,年龄,介绍)>

<!ELEMENT 名字 (#PCDATA)>

<!ELEMENT 年龄 (#PCDATA)>

<!ELEMENT 介绍 (#PCDATA)>

说明:在默认情况下,浏览器不检测,需要自己写代码来测试(js代码)

checkDTD.html

<html>

<head>

<title>测试DTD</title>

<meta http-equiv="content-type" content="text/html charset=utf-8"/>

<script language='javascript'>

<!--

//创建一个xml解析器

var xmldoc = new ActiveXObject("Microsoft.XMLDOM");

//开启校验功能

xmldoc.validateOnParse=ture;

//指定校验的xml文档

xmldoc.load("class.xml");

//获取错误信息

document.write('错误信息='+xmldoc.parseError.reason+'<br/>');

document.write("错误的行号="+xmldoc.parseError.line);

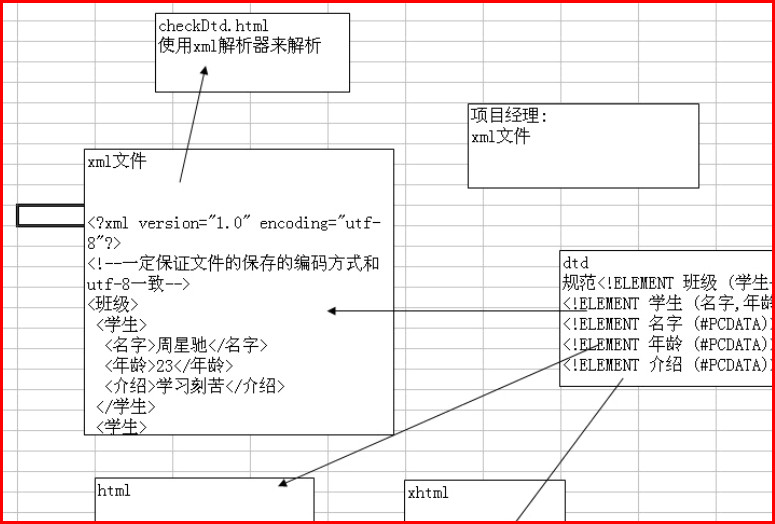
//-->

</script>

</head>

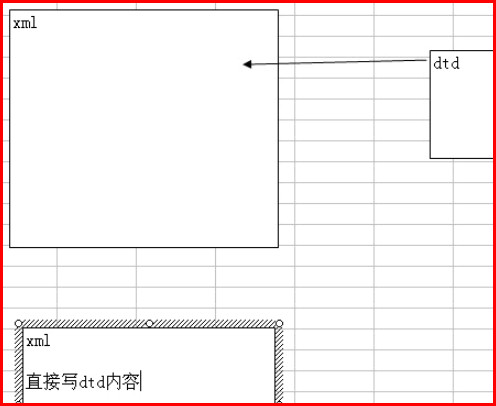
</html>

三个文件关系



***2.3 细节说明***

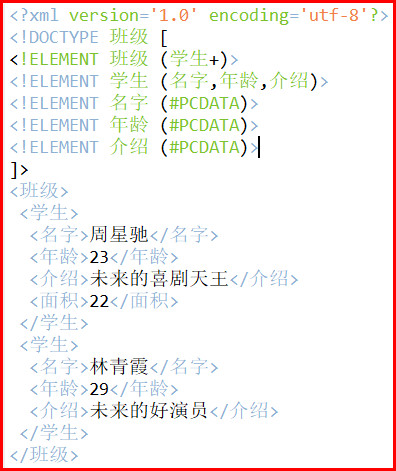
**1. DTD文档的声明与引用**



- 内部DTD文档

<!DOCTYPE 根元素 [定义内容]>

案例如下



- 外部文档

**<!DOCTYPE** 根元素 **SYSTEM** “DTD文档路径”**>**(引用文件在本地)

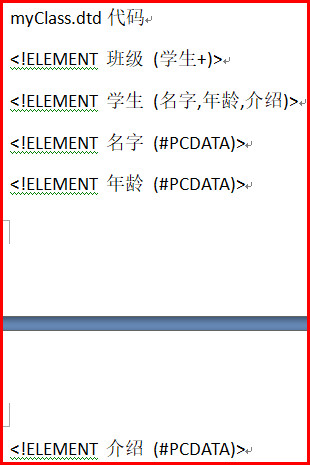
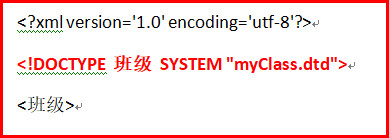
**<!DOCTYPE** 根元素 **PUBLIC** “DTD名称” “DTD文档URL”**>**

案例如图:

不在本地:

2018-01-21_135317.jpg

在本地:



- 内外部DTD文档结合

<!DOCTYPE 根元素 SYSTEM “DTD文档路径”

定义内容

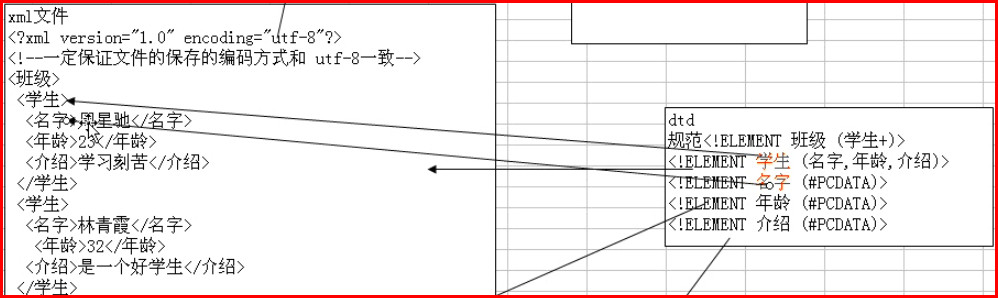
>

**2. DTD文档的语法**

**1.语法**:

**<!ELEMENT** NAME CONTENT**>**

说明:

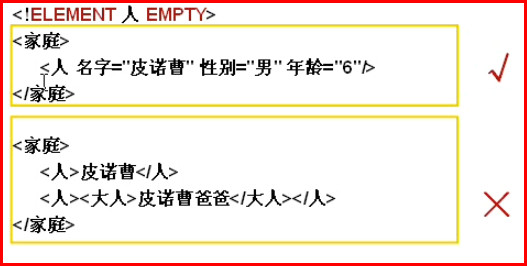


1.**ELEMENT** 是关键字

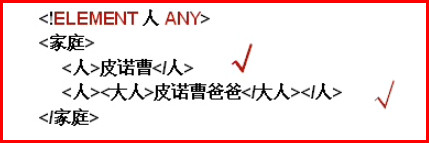
2.NAME 是元素名称

3.CONTENT 表示元素类型(大写)

EMPTY :表示该元素不能包含子元素与文本,但可以有属性

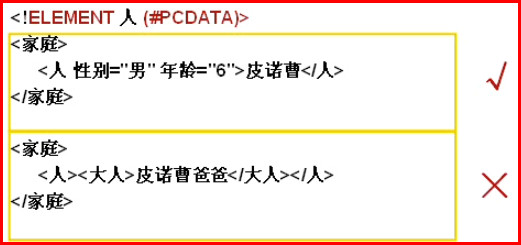


ANY :表示该元素可以包含在DTD中定义的元素内容



注意:这个<大人></大人>是一定在DTD文档中定义过的

#PCDATA :表示可以包含任何字符数据,但是不能在其中包含任何子元素其它类型(组合)



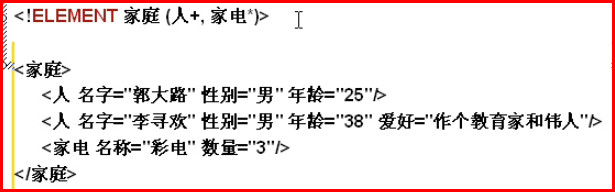
组合类型

<!ELEMENT NAME (name1修饰符1,name2修饰符2.....)>

元素的修饰符:



案例:



**2.属性定义-语法**

**<!ATTLIST** 元素名称

属性名称 类型 属性特点

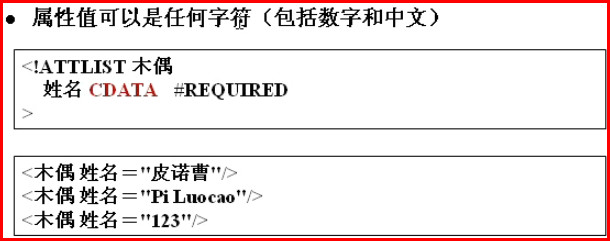
属性名称2 类型 属性特点

**>**

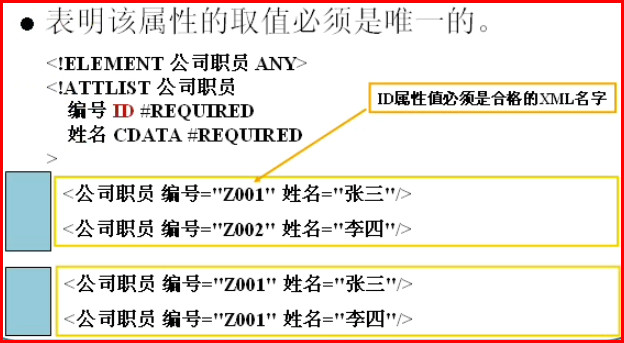
元素属性列表说明:

类型有以下几种

**CDATA (属性值可以是字符或数字)**

****

**ID (属性值是唯一的并且要以字母开头)**

****

**IDREF/IDREFS (id引用)[其中IDREF表示只能参考一个,IDREFS表示可以参考多个]**

****

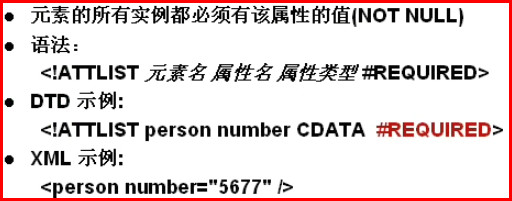
**Enumerated (枚举类型)**



**ENTITY/ENTITIES (实体)**

属性特点有以下几种

***#REQUIRED (该属性必须要给)***

******

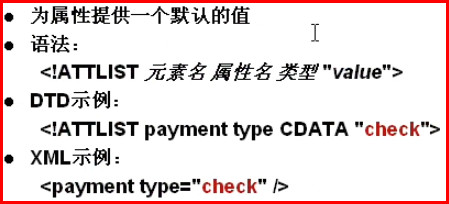
***#IMPLIED (该属性可有可无)***

******

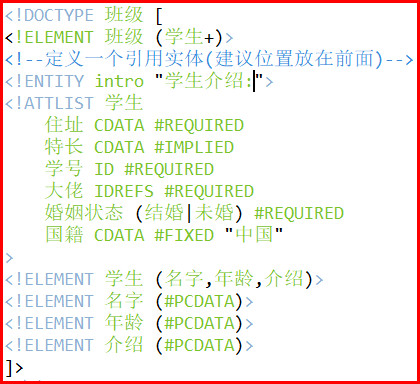
***#FIXED value (该属性值是固定值) [有必须写指定值,没有不报错]***

******

***Default value (该属性值不给就用默认值)***

******

综合案例(使用内部DTD文档说明):

2018-01-21_154420.jpg 

**3.实体定义**

1.基本介绍

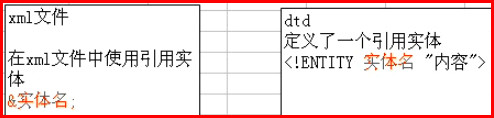
实体用于为一段内容创建一个别名,以后在XML文档中就可以使用别名引用这段内容

$str = “abcd”;

在DTD定义中,一条<!ENTITY ....>语句用于定义一个实体

实体可分为两种类型: **引用实体**和**参数实体**

2.实体定义—引用实体



1.引用实体主要在XML文档中被应用

2.语法格式(DTD中定义)

<!ENTITY 实体名称 “实体内容” > :直接转变成实体内容

3.引用方式(XML中引用):

&实体名称;

举例:

<!ENTITY copyrigh “I am a hero”>

........

&copyright;

内部DTD文档(外部DTD文档类似)

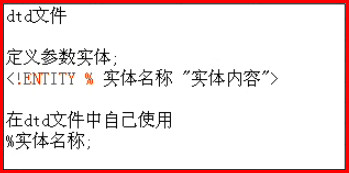
2018-01-21_154111.jpg

xml文件中引用

2018-01-21_154122.jpg

3.实体定义—参数实体

原理图:



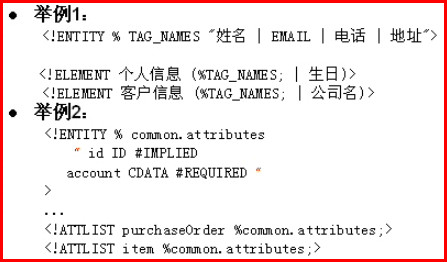
参数实体被DTD文件自身使用

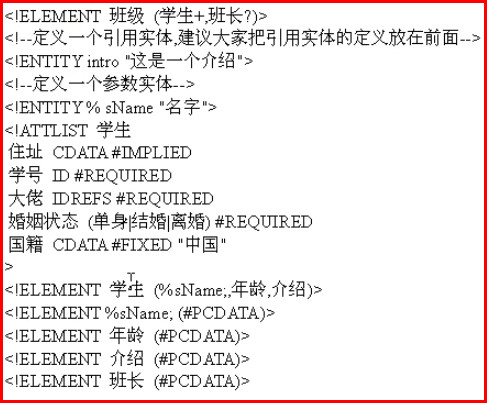
语法格式:

<!ENTITY % 实体名称 “实体内容”>

引用方式

%实体名称;





***2.4 dtd实际操作***

具体开发dtd文件来约束xml案例:

使用工具为zend studio,开发目标是xml与dtd

综合练习,项目经理定义了一个dtd文件,要求根据dtd文件来编写一个合法且符合dtd规范的xml,

代码如下:

<?xml version='1.0' encoding='utf-8'?>

<!DOCTYPE CATALOG [

<!ENTITY AUTHOR "John Doe">

<!ENTITY COMPANY "JD Power Tools, Inc.">

<!ENTITY EMAIL "jd@jd-tools.com">

<!ELEMENT CATALOG (PRODUCT+)>

<!ELEMENT PRODUCT

(SPECIFICATIONS+,OPTIONS?,PRICE+,NOTES?)>

<!ATTLIST PRODUCT

NAME CDATA #IMPLIED

CATEGORY (HandTool|Table|Shop-Professional) "HandTool"

PARTNUM CDATA #IMPLIED

PLANT (Pittsburgh|Milwaukee|Chicago) "Chicago"

INVENTORY (InStock|Backordered|Discontinued) "InStock">

<!ELEMENT SPECIFICATIONS (#PCDATA)>

<!ATTLIST SPECIFICATIONS

WEIGHT CDATA #IMPLIED

POWER CDATA #IMPLIED>

<!ELEMENT OPTIONS (#PCDATA)>

<!ATTLIST OPTIONS

FINISH (Metal|Polished|Matte) "Matte"

ADAPTER (Included|Optional|NotApplicable) "Included"

CASE (HardShell|Soft|NotApplicable) "HardShell">

<!ELEMENT PRICE (#PCDATA)>

<!ATTLIST PRICE

MSRP CDATA #IMPLIED

WHOLESALE CDATA #IMPLIED

STREET CDATA #IMPLIED

SHIPPING CDATA #IMPLIED>

<!ELEMENT NOTES (#PCDATA)>

]>

<CATALOG>

<PRODUCT NAME='冰箱' CATEGORY="HandTool" PLANT="Chicago" INVENTORY="Backordered">

<SPECIFICATIONS WEIGHT="120kg" POWER="1.2kw">这是一款经典双开门海尔冰箱</SPECIFICATIONS>

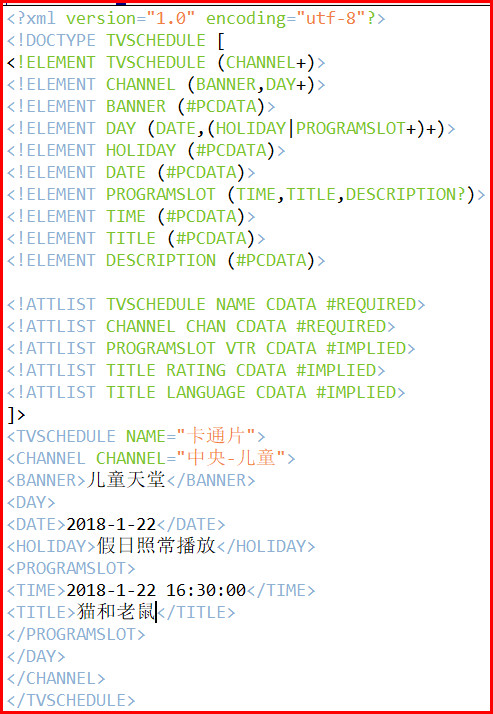
<PRICE>2800.00</PRICE>

<NOTES>轻轻的拿,轻轻的放</NOTES>

</PRODUCT>

</CATALOG>

homework1.xml



第三章 php --XML

***3.1 常用技术***

php操作xml技术有

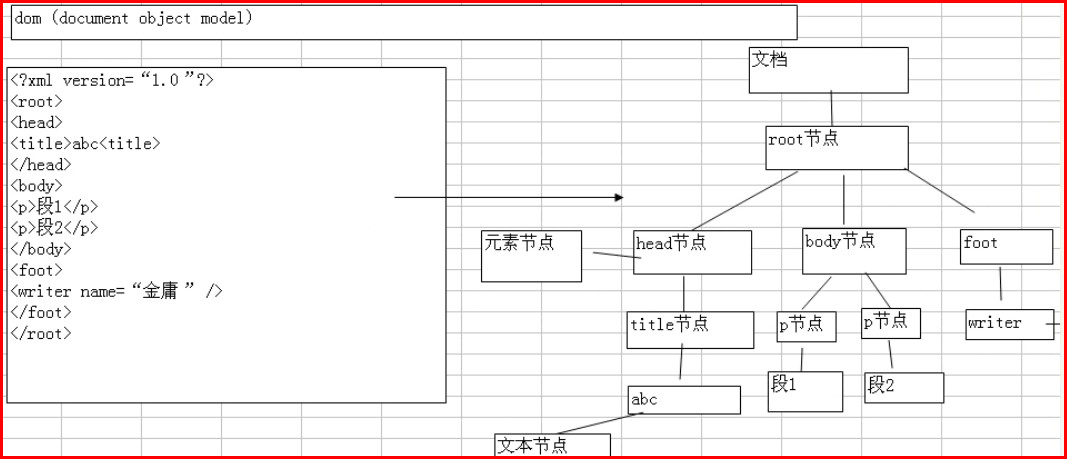
* + PHP 与 DOM
  + PHP 与 XPath
  + SimpleXML

***3.2 PHP与 DOM***

1.DOM的基本介绍

DOM(document object model) 文档-对象-模型,核心思想是把一个xml(html,xhtml)文件看作一个对象模型,通过对象的方式来操作我们的xml文件

2.基本图解:

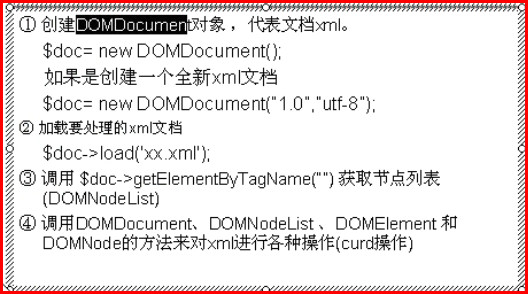


3.快速入门案例:

对前面的的class.xml班级信息进行增删改查(crud)

tips:如果希望知道自己创建的DOMDocument类的实例有哪些方法或属性,可以使用var\_dump(对像);

php操作步骤以及需要类方法



4.php对xml文件查询





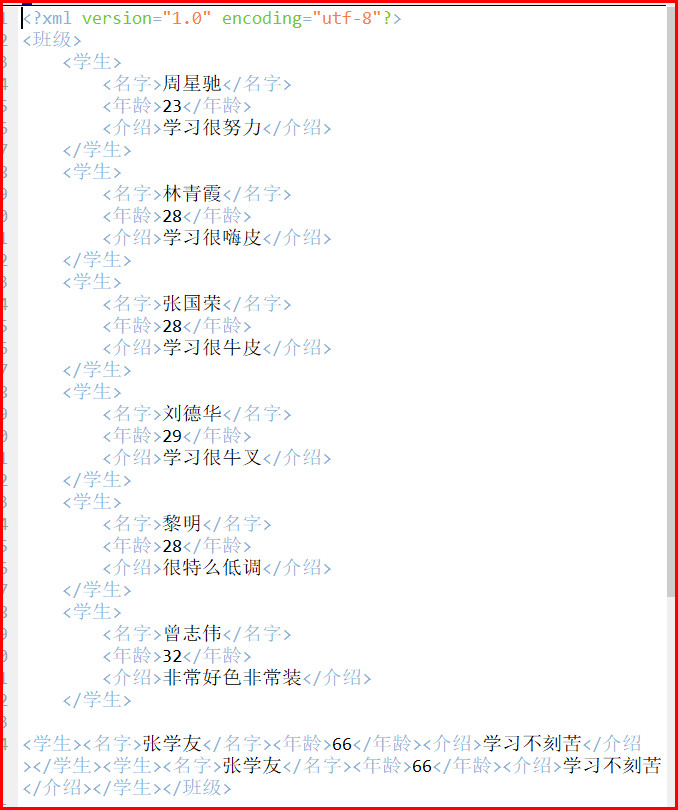
5.php对xml文件的增,删,改

5.1增加



**如果save方法中是原文件名就增加,新文件名,就创建一个新文件放入数据!!!**

2018-01-22_200933.jpg



**新文件内容与php操作的xml文件内容一样,只是会加入添加内容**

5.2 删除

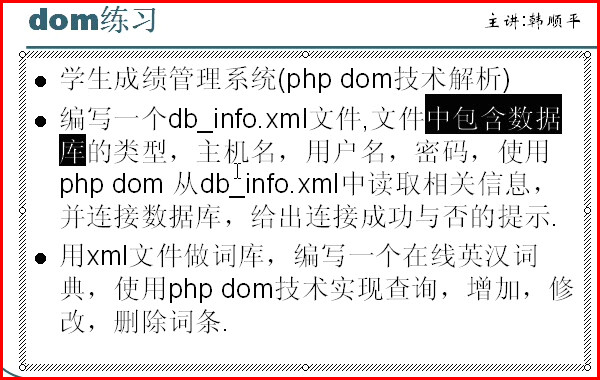


**删除节点,可以不获取root节点,直接用删除的节点调用parentNode来删除自身**

5.3 修改



DOM综合案例练习

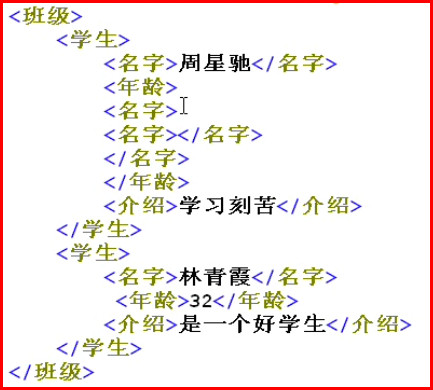


***3.3 PHP与 xpath***

1.xpath的介绍:

xpath的好处: 有一个问题引出: 我们可不可以夸层取出节点:

php的DOM可以跨层取节点,但是如果这种



直接跨层取就无法保证层次关系,**这时,xpath就可以解决这种问题**

XPath(XML Path Language) 是一种标示节点位置的语言,可以用来描述XML元素的位置,如同硬盘上的文件夹路径,比如:

C:\wamp\www\uuu\myClass.xml

上面的路径指出xml文件所在位置,而XPath指令可以指出XML(节点)元素在XML文件中的位置,也就是树状结构的节点位置

XPath语言主要目的是描叙节点相对其它节点的位置,可以取得所有符合条件的节点,称为位置路径(Location Path)

XPath的设计核心思想: 可以通过XPath迅速简捷定位你希望查找的节点

XPath使用解析

**1.php程序在载入XML文件建立DOMDocument对象后,就可以建立DOMXPath对象,如下所示:**

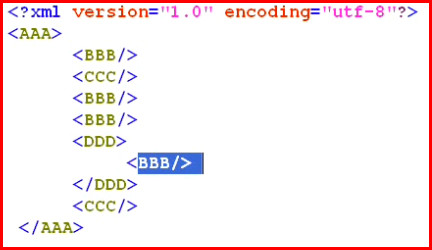
**$xpath = new DOMXPath($doc);**

**2.在建立好的DOMXPath 对象$xpath后,可以使用query()方法找出XPath位置路径表达式的节点对象,比如:**

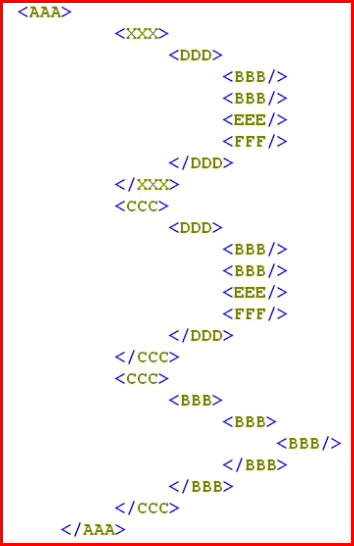
**$items = $xpath->query(“xpath路径表达式”);**

**上述query()方法的参数是XPath表达式,传回DOMNodeList对象,如果没有找到节点,就返回空**

2.快速入门











***3.4 PHP与 SimpleXML***

1.基本介绍

SimpleXML可以结合XPath来使用,该技术的核心思想是以面向对象的方式来操作XML文件.

PHP5及以上版本的SimpleXML可以将XML文件的元素都转换成对象,相同的子元素转换成对象数组,可以使用foreach取得指定元素名称.内容与属性值

通过simplexml\_load\_file()函数来载入xml文件档案,还有

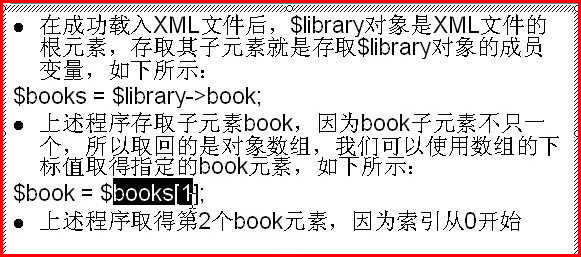
simplexml\_import\_dom();

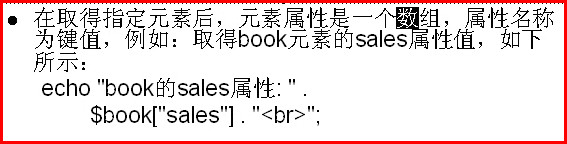
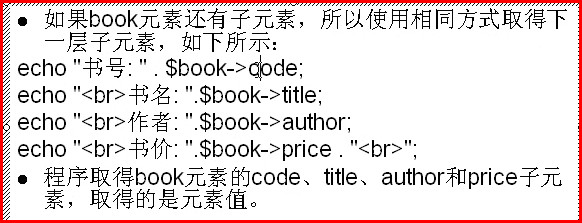
simplexml\_load\_string();

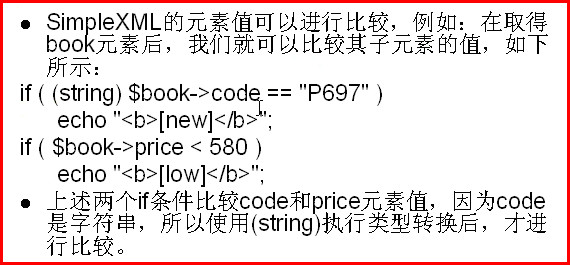
比如:

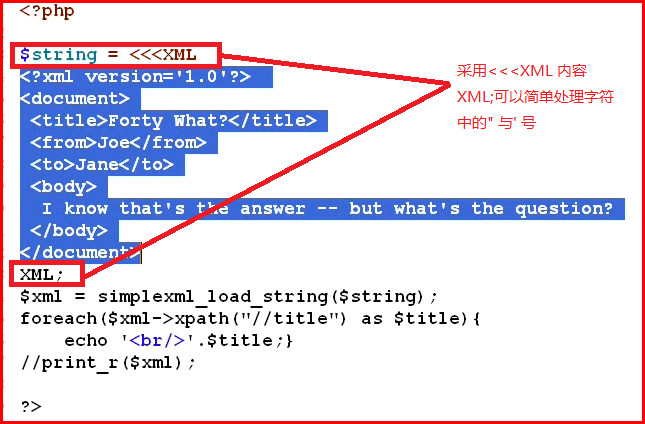
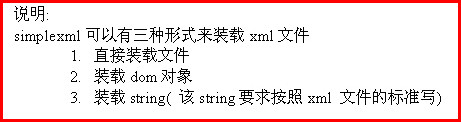
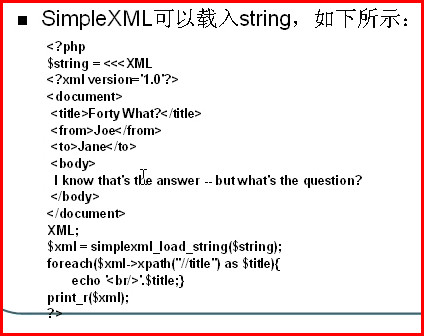
$libary = simplexml\_load\_file(“book.xml”);

上述程序载入XML文档book.xml,函数返回SimpleXMLElement类型对象,载入失败则返回false









2.快速入门案例





