# XMl

## 概念

eXtensible Markup Language 可扩展的标记语言。其实就是一个文件，文件的后缀为 .xml

如果说java是一门跨平台的语言，那XML就是跨平台的数据。

应用场景

1. 可以用来保存数据

2. 可以用来做配置文件：利用XML可以保存具有关系的数据的特性

3. 利用XML跨平台的特性，用来在不同的操作系统不同的开发语言之间传输数据

### 文档声明

简单声明， version : 解析这个xml的时候，使用什么版本的解析器解析

<?xml version="1.0" ?>

encoding : 解析xml中的文字的时候，使用什么编码来翻译

<?xml version="1.0" encoding="gbk" ?> encoding属性默认值为老外喜欢的iso8859-1

standalone : no - 该文档会依赖关联其他文档 ， yes-- 这是一个独立的文档

<?xml version="1.0" encoding="gbk" standalone="no" ?> 默认值为yes表示是一个独立文档。

注意：很多的解析器会忽略这个属性，但是学习知识要按标准去学，所以这个属性也要掌握。

encoding详解

在解析这个xml的时候，使用什么编码去解析。 ---解码。

老师演示了 ：下面xml内容正常点击保存拿到浏览器打开会报错：

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

<a>

<b>你好</b>

</a>

原因：windows系统默认是gbk编码来保存的：你在点击保存的时候其实用的是gbk编码保存，xml内容又是utf-8(拿到浏览器中渲染是根据xml头文件指定的编码来解析)就会报错；

解决办法：

第一：保存文件的时候使用另存为:编码选择为utf-8进行保存

第二：头文件encoding改为gbk，点击保存即可:因为windows默认就是ANSI(windows根据默认编码操作)来保存的

最后注意为了通用：建议使用UTF-8编码保存，以及encoding 都是 utf-8

### 元素(标签)

一个xml标签就是一个元素，标签分为开始标签、结束标签

标签体：在开始标签和结束标签之间的文本称为标签体。

自闭标签: 一个标签即不含标签体也不包含其他标签，把开始标签和结束标签进行合并，叫自闭标签。

格式良好的XML文档应该具有并且只能有一个根标签，其他标签都应该是这个跟标签的子孙标签。

XML 命名规则：

名称可以含字母、数字以及其他的字符 ，区分大小写，例如，<P>和<p>是两个不同的标记。

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、中间不能包含冒号（:）

属性

一个元素可以包含多个属性，属性值要用单引号或双引号括起来，属性的命名规范，参照元素的命名规范。

注释

<!-- 这是一段注释 --> 注意：注释不能出现在文档声明之前。

### CDATA区和转义字符

转义字符

& --> &amp;

< --> &lt;

> --> &gt;

" --> &quot;

' --> &apos;

被CDATA区扩起来的内容，将会被浏览器当作文本来处理。

<des><![CDATA[<a href="http://www.baidu.com">我爱黑马训练营</a>]]></des>

CDATA区和转义字符的区别

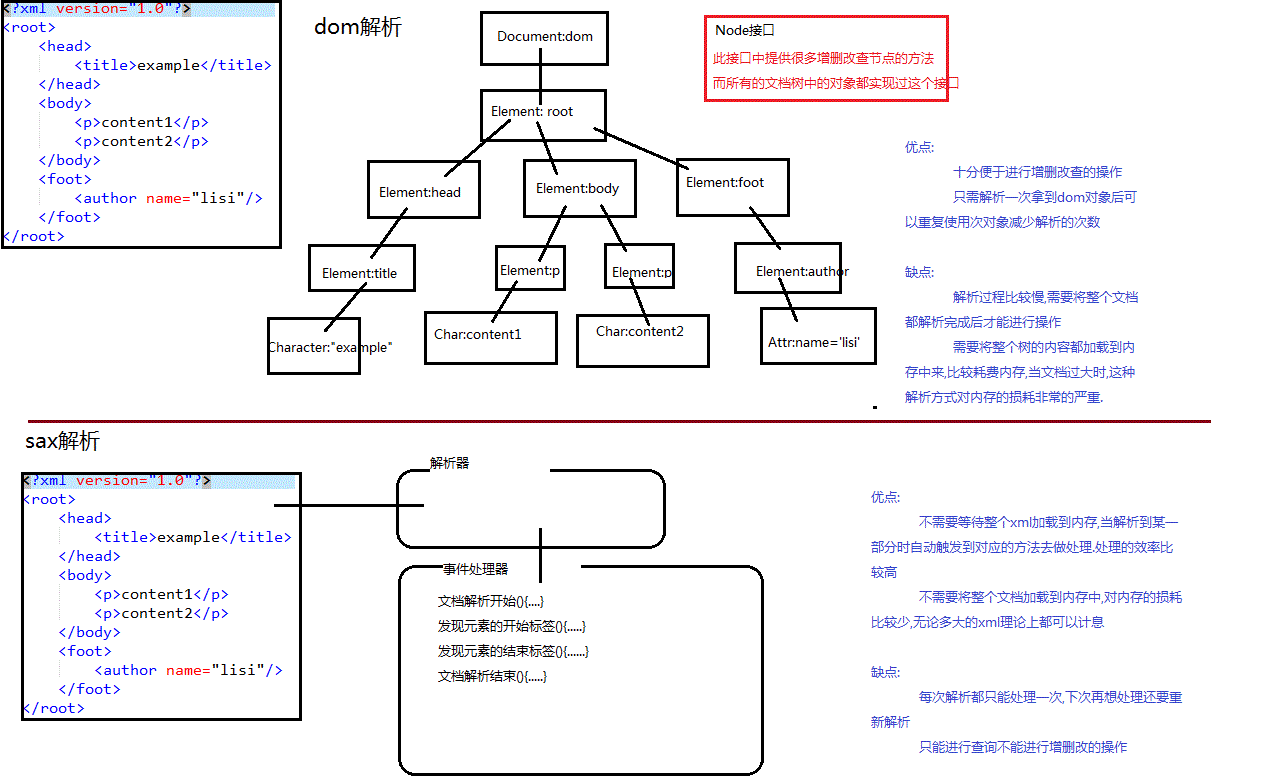
(1)CDATA区可以成段的进行转义，而转义字符一次只能转义一个字符

(2)CDATA区转义的字符可以保存数据本来的格式只是通知解析器按文本去处理。转义字符改变了数据本身的内容，利用其他字符替代了转义字符。

请思考，如果要转义的内容是一段js程序的话，如果用转义字符合适不合适？

## 解析XML

常用的有两种Dom和SAX：jaxp既有dom方式也有sax方式,并且这套解析api已经加入到j2se的规范中



### DOM

DOM4J是第三方的，基于Dom进行优化的。

需要导入包：dom4j-1.6.1.jar和hibernate包中的jaxen-1.1-beta-7.jar

<书架>

<书 id="1">

<书名>倚天屠龙记</书名>

<作者>金庸</作者>

<价格>100</价格>

</书>

<书 id="2">

<书名>陆小凤</书名>

<作者>古龙</作者>

<价格>99</价格>

</书>

</书架>

整体简单流程

//第一步 创建对象 通过流read读取全部xml内容放入内存中

SAXReader sr = new SAXReader();

Document d=sr.read(Demo1.class.getClassLoader().getResourceAsStream(”a.xml”));

//第二步 获得根节点对象Element对象

Element root=d.getRootElement();

//第三步 获得根节点中的下一个标签<书>：用根节点的对象root获取

获取元素有3种方式 ①获取root下指定的标签：root.element("书");

②获取全部：list获取root.elements();迭代：root.elementIterator()

③获取重名的名字全部：list获取root.elements(“书”);迭代：root.elementIterator(“书”)

List<Element> list=root.elements();

Element shu=list.get(1);

Element shu2=root.element("书");

//第四步 获取属性得到对象Attribute

Attribute id=shu.attribute("id");

String idName=id.getName();//打印：id 就是属性名字不是值

String idValue2=id.getValue();//打印：值 ：1

String idValue=id.getStringValue();//也是打印：1

System.out.println(shu.element("作者").getStringValue());//只有StringValue这个属性，打印：金庸

Element shuming=shu.element("书名");

shuming.getText(); //获取到书名：倚天屠龙记

System.out.println();

封装方法：Dom4j是一次性加载到内存中的：可以重复调用

封装加载xml方法，使用成员变量接收document对象

public static void loadDocument(){ //封装

try{

SAXReader saxReader = new SAXReader();

doc = saxReader.read(new File("a.xml"));

}catch (Exception e) {

// TODO: handle exception

e.printStackTrace();

}

}

查询操作

public static void showPhoneInfo() {

//第一步 获取XML的根节点

Element root = doc.getRootElement();

//第二步 遍历所有的Brand标签

for (Iterator itBrand = root.elementIterator(); itBrand.hasNext();) {

Element brand = (Element) itBrand.next();

// 输出标签的name属性

System.out.println("品牌：" + brand.attributeValue("name"));

// 遍历Type标签

for (Iterator itType = brand.elementIterator(); itType.hasNext();) {

Element type = (Element) itType.next();

// 输出标签的name属性

System.out.println("\t型号：" + type.attributeValue("name"));

}

}

}

添加操作

public static void addNewPhoneInfo(){

// 获取XML的根节点

Element root = doc.getRootElement();

// 创建Brand标签

Element el = root.addElement("Brand");

// 给Brand标签设置属性

el.addAttribute("name", "三星");

// 创建Type标签

Element typeEl = el.addElement("Type");

// 给Type标签设置属性

typeEl.addAttribute("name", "Note4");

saveXML("src/收藏信息.xml");

}

修改操作

public static void updatePhoneInfo(){

// 获取XML的根节点

Element root = doc.getRootElement();

int id = 0;

for (Iterator itBrand = root.elementIterator(); itBrand.hasNext();) {

Element brand = (Element) itBrand.next();

id++;

brand.addAttribute("id", id + "");

}

saveXML("src/收藏信息.xml");

}

删除操作

public static void deleteItem(){

// 获取XML的根节点

Element root = doc.getRootElement();

int id = 0;

for (Iterator itBrand = root.elementIterator(); itBrand.hasNext();) {

Element brand = (Element) itBrand.next();

if (brand.attributeValue("name").equals("华为")) {

brand.getParent().remove(brand);

}

}

// saveXML("src/收藏信息.xml");

}

进行修改、增加、删除最后需要进行重新保存操作

public static void saveXML(String path){

try {

OutputFormat format = OutputFormat.createPrettyPrint();

format.setEncoding("GBK"); // 指定XML编码

XMLWriter writer;

writer = new XMLWriter(new FileWriter(path), format);

writer.write(doc);

writer.close();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

### SAX

SAX的优点：

解析速度快，占用内存少

SAX的缺点：

无法知道当前解析标签（节点）的上层标签，及其嵌套结构，仅仅知道当前解析的标签的名字和属性，要知道其他信息需要程序猿自己编码

只能读取XML，无法修改XML

无法随机访问某个标签（节点）

SAX解析适用场合

对于CPU资源宝贵的设备，如Android等移动设备

对于只需从xml读取信息而无需修改xml

解析步骤

1、得到xml文件对应的资源，可以是xml的输入流，文件和uri

2、得到SAX解析工厂（SAXParserFactory）

3、由解析工厂生产一个SAX解析器（SAXParser）

4、传入输入流和handler给解析器，调用parse()解析

# 服务器

## Tomcat

### 安装和配置

1、tomcat5需jdk1.4以上，tomcat6需jdk1.5以上，tomcat7需jdk1.6以上。安装目录不能包含中文和空格

2、JAVA\_HOME环境变量: 指定tomcat启动时使用的jdk所在的位置

3、Catalina\_Home环境变量: startup.bat启动哪个tomcat由此环境变量指定，如果不配置则启动当前tomcat，推荐不要配置此环境变量

### 目录结构

bin ----存放tomcat启动关闭所用的批处理文件

conf --- tomcat的配置文件，最终要的是server.xml(更改tomcat运行所在的端口号，从8080改为80)

lib --- tomcat运行时需要的jar包所在的目录

logs ---运行时产生的日志文件

temp --- tomcat运行时产生的临时文件存放的目录,不需要我们管理

webapps --- 开发中最常用的目录,web应用放置到此目录下浏览器可以直接访问

work --- 工作目录,tomcat运行时产生的工作文件存放在这个目录中

### 配置访问路径

#### 虚拟主机

一个tomcat可以认为是一台真实主机.在一台真实主机中可以配置多个站点,这些站点在访问者看来访问他们就像在访问各自独立的主机一样,所以我们可以认为这些站点都运行在tomcat这台真实主机当中的各自的虚拟主机当中.一个网站就可以认为是一个虚拟主机

默认的虚拟主机名称为：localhost，虚拟主目录：webapps

<Host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true"> </Host>

自定义配置虚拟主机

第一步在conf/server.xml中<Engin>标签下配置<Host>标签就可以为tomcat增加一台虚拟主机

<Host name="www.wwl.com" appBase="D:\zlf"></Host> 意思是：配置了名称为www.wwl.com，虚拟主目录：d盘下的zlf文件夹

name -- 指定虚拟主机的名称,浏览器通过这个名称访问虚拟主机

appBase -- 虚拟主机管理的目录,放置在这个目录下的web应用当前虚拟主机可以自动加载

第二步 在C:\Windows\System32\drivers\etc\hosts添加内容

127.0.0.1 www.wwl.com 把此www.wwl.com转换为ip:127.0.0.1

这两步就可以通过：www.wwl.com:8080/a.jsp访问

注意：localhost是不用配置的windows默认就有

由于浏览器访问地址www.wwl.com时,需要将地址翻译成对应的ip才能找到服务器,windows系统先去找hosts文件有没有，然后才去找网络上的dns服务器翻译.此时可以使用hosts文件模拟dns的功能

配置默认虚拟主机

通过http://127.0.0.1:8080/访问进入的是默认tomcat的猫主页，进行更改的步骤如下：

<Engine name="Catalina" defaultHost="localhost">把defaultHost的值改为自己配置的虚拟主机即可

#### web工程

一个虚拟主机中会有很多的web资源,但是web资源不能直接交给虚拟主机管理,需要按照一定方式组织成web应用虚拟主机才能使用.一般来说我们会按照功能将某一功能相关的所有的web资源组织成一个web应用后再交给虚拟主机.

目录结构

工程根路径：Html、css、js、jsp、WEB-INF(目录它必须有)

WEB-INF目录下：classes(java字节码)、lib(包)、web.xml

web.xml文件的作用：

某个web资源配置为web应用首页

将servlet程序映射到某个url地址上

为web应用配置监听器

为web应用配置过滤器

但凡涉及到对web资源进行配置，都需要通过web.xml文件

Tomcat配置WEB工程有三个方法

第一种：最简单的tomcat默认配置好的目录

直接将web工程放入虚拟主机管理的webapps目录下即可：如工程名为a文件夹

访问：http://localhost:8080/a/user

如果把工程名为a文件夹中的所有资源放入webapps/ROOT/下访问就不用工程名访问(缺省web工程名)

访问：http://localhost:8080/a.jsp

第二种 自定义目录和webapps目录一样的功能(需要重启tomcat服务 )

步骤：在conf/server.xml的<Host>标签下配置<Context path="虚拟路径" docBase="真实路径目录">

配置为<Context path="/c" docBase="D:\wwl"/> 注意：它是自闭标签格式要写对，docBase路径是指定到目录

访问http://localhost:8080/c/ a.jsp即可

配置为<Context path="" docBase="D:\wwl"/> path为空，即为缺省路径

http://localhost:8080/a.xml

第三种在tomcat/conf/引擎目录名/虚拟主机目录名/之下建立一个.xml文件

1、其中文件名用来指定虚拟路径，如果名字有’\_’的用#代替表示，文件名为ROOT.xml是缺省web工程名

2、文件内容：<Context docBase="真实目录">，

注意：Tomcat默认的路径：conf\Catalina\localhost;查看方式：server.xml中：<Engine name="Catalina" defaultHost="localhost">

### 配置代码自动更新

Conf目录下的context.xml内容：

<Context>

<WatchedResource>WEB-INF/web.xml</WatchedResource> //这就是每个web工程为什么要这样配置WEB-INF/web.xml

<Context reloadable="true"> 让tomcat能够自动重新加载，我们修改过的代码和配置

注意： reloadable设为true，目的是为了方便开发阶段， 它会影响tomcat性能；

</Context>

# 协议

## HTTP协议

## 一、HTTP协议详解之URL篇

http（超文本传输协议）是一个基于请求与响应模式的、无状态的、应用层的协议，常基于TCP的连接方式，HTTP1.1版本中给出一种持续连接的机制，绝大多数的Web开发，都是构建在HTTP协议之上的Web应用。

URI与URL区别

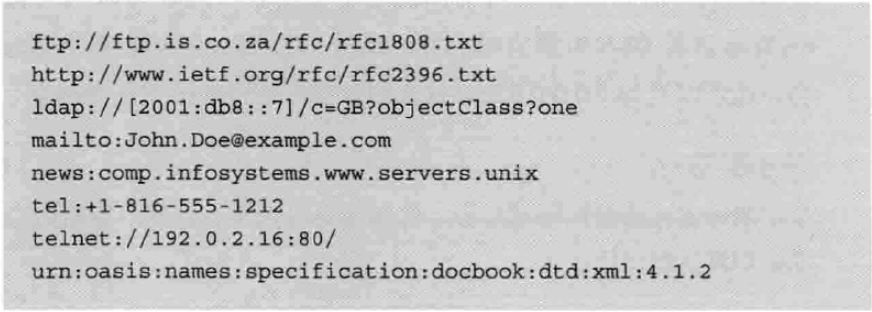
URL：(Uniform/Universal Resource Locator 的缩写，统一资源定位符)。

URI：(Uniform Resource Identifier 的缩写，统一资源标识符)（代表一种标准）。

关系：URI 属于 URL 更高层次的抽象，一种字符串文本标准。URI 属于父类，而 URL 属于 URI 的子类。

区别:URI 表示请求服务器的路径，定义这么一个资源。而 URL 同时说明要如何访问这个资源（http://）。

下图是URI示例：



http://host[":"port][ path]的解释：

http表示要通过HTTP协议来定位网络资源；

host表示合法的Internet主机域名或者IP地址；

port指定一个端口号，为空则使用缺省端口80；

path指定请求资源的URI；如果URL中没有path，请求URI时，是以“/”的形式给出(系统自动)

1、输入：www.guet.edu.cn浏览器自动转换成：http://www.guet.edu.cn/

2、http:192.168.0.116:8080/index.jsp

## 二、HTTP协议详解之请求篇

http请求由三部分组成，分别是：请求行、消息报头（请求头）、请求正文（请求体）

### 请求行

请求行以一个方法符号开头，以空格分开，后面跟着请求的URI和协议的版本，

格式如下：Method Request-URI HTTP-Version CRLF ： POST /WEB12/form.html HTTP/1.1

其中 Method：请求方法；Request-URI是一个统一资源标识符；HTTP-Version：HTTP协议版本；CRLF表示回车和换行（除了作为结尾的CRLF外，不允许出现单独的CR或LF字符）。

请求方法（所有方法全为大写）有多种，各个方法的解释如下：

GET 请求获取Request-URI所标识的资源 除了表单提交指定方法外都是GET请求

POST 在Request-URI所标识的资源后附加新的数据 常用于提交表单

HEAD 请求获取由Request-URI所标识的资源的响应响应头

PUT 请求服务器存储一个资源，并用Request-URI作为其标识

DELETE 请求服务器删除Request-URI所标识的资源

TRACE 请求服务器回送收到的请求信息，主要用于测试或诊断

CONNECT 保留将来使用

OPTIONS 请求查询服务器的性能，或者查询与资源相关的选项和需求

HEAD方法与GET方法几乎是一样的，对于HEAD请求的回应部分来说，它的HTTP头部中包含的信息与通过GET请求所得到的信息是相同的。利用这个方法，不必传输整个资源内容，就可以得到Request-URI所标识的资源的信息。该方法常用于测试超链接的有效性，是否可以访问，以及最近是否更新。

### 请求头

Accept: 客户端向服务器端表示，我能支持什么类型的数据。

Referer ： 真正请求的地址路径，全路径

Accept-Charset请求报头域用于指定客户端接受的字符集 Accept-Charset:iso-8859-1,gb2312

Accept-Language: 支持语言格式

User-Agent: 用户代理 向服务器表明，当前来访的客户端信息。

Content-Type： 提交的数据类型。经过urlencoding编码的form表单的数据

Accept-Encoding： gzip, deflate ： 压缩算法 。

Host ： 主机地址

Content-Length： 数据长度

Connection : Keep-Alive 保持连接

Cache-Control ： 对缓存的操作

请求报头举例：

GET /form.html HTTP/1.1 (CRLF)

Accept: image/jpeg,application/x-shockwave-flash,application/vnd.ms-powerpoint,application/msword,\*/\* (CRLF)

Accept-Language:zh-cn (CRLF)

Accept-Encoding:gzip,deflate (CRLF) 如果请求消息中没有设置这个域服务器假定客户端对各种内容编码都可以接受

If-Modified-Since:Wed,05 Jan 2007 11:21:25 GMT (CRLF)

If-None-Match:W/"80b1a4c018f3c41:8317" (CRLF)

User-Agent:Mozilla/4.0(compatible;MSIE6.0;Windows NT 5.0) (CRLF)

Host:www.guet.edu.cn (CRLF)

Connection:Keep-Alive (CRLF)

(CRLF)

### 请求体

发送的数据呈现的是key=value ,如果存在多个数据，那么使用 &

firstname=zhang&lastname=sansan

### GET和POST的区别

请求中：POST多了两个请求头：Content-Type和Content-Length

Content-Type：是普通内容还是要上传文件类型

Content-Type: application/x-www-form-urlencoded 这是普通内容

Content-Length：请求体中的参数内容长度

Content-Length: 25 最新浏览器都隐藏了一个&：firstname=zhang&lastname=sansan

## 三、HTTP协议详解之响应篇

HTTP响应也是由三个部分组成，分别是：状态行、消息报头、响应正文

### 响应行

响应行格式如下：

HTTP-Version Status-Code Reason-Phrase CRLF HTTP/1.1 200 OK （CRLF）

HTTP-Version：服务器HTTP协议的版本；Status-Code：服务器发回的响应状态代码；Reason-Phrase：状态代码的文本描述。

状态代码有三位数字组成，第一个数字定义了响应的类别，且有五种可能取值：

1xx：指示信息--表示请求已接收，继续处理

2xx：成功--表示请求已被成功接收、理解、接受

3xx：重定向--要完成请求必须进行更进一步的操作

4xx：客户端错误--请求有语法错误或请求无法实现

5xx：服务器端错误--服务器未能实现合法的请求

常见状态代码、状态描述、说明：

200 OK //客户端请求成功

400 Bad Request //客户端请求有语法错误，不能被服务器所理解

401 Unauthorized //请求未经授权，这个状态代码必须和WWW-Authenticate报头域一起使用

403 Forbidden //服务器收到请求，但是拒绝提供服务

404 Not Found //请求资源不存在，eg：输入了错误的URL

500 Internal Server Error //服务器发生不可预期的错误

503 Server Unavailable //服务器当前不能处理客户端的请求，一段时间后可能恢复正常

### 响应头

Location响应报头域用于重定向接受者到一个新的位置。Location响应报头域常用在更换域名的时候。

Server: 服务器是哪一种类型。 Tomcat

Content-Type ： 服务器返回给客户端你的内容类型 Content-Type:text/html;charset=ISO-8859-1

Content-Length ： 返回的数据长度

Date ： 通讯的日期，响应的时间

Last-Modified实体报头域用于指示资源的最后修改日期和时间

### 响应体

如果是：Content-Type：text/html;charset=ISO-8859-1则返回整个HTMl页面内容