XML

什么是XML

XML指可扩展标记语言（extensible markup language）

XML被设计用来存储和传输数据

XML很重要，但是也很容易学习。

元语言：用来定义其他语言的语言

XML能干什么

数据载体: 存放及有一定结构的数据
作为配置文件：增加程序的灵活性

XML语法（良构的XML）

第一行写 XML 的处理指令 <? ?\>

在处理指令中可以指定XML的版本号和编码格式

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

如上所示，version属性指XML的版本，encoding指代该XML文件的编码格式，standalone = "yes/no" 指代该文件是否独立。

有且只有一个root元素

元素的规则

- 开始结束标签配对出现

- 开始标签中可以包含属性
- 可以出现空元素
- 元素内可跟元素
- 子元素
- 文本内容
- 空（空格、制表符、换行等都不是空）
- 混合内容
- 不能交叉嵌套
- 命名规则
- 以字母、_开始
- 后面可以跟字母、数字、-、.、_
- 大小写敏感
- 不能使用保留字
- 没有长度限制
- 注意
  - 命名尽量表达出含义
  - 名字有多个单词组成，可以使用驼峰表示法
  - 或者将多个单词用_隔开
- 属性的规则
- key=value配对出现
- 属性只能出现在开始标签中，同一开始标签可以出现多个属性，但属性不能重名
- 属性必须有值，而且值必须用引号引起来
- 良构的XML：满足XML语法规则的XML文件

注释

```
<!-- 我是一段注释 -->
```

PCDATA、CDATA、实体

- PCDATA

- Parsed Character Data

- 可以被XML文档解析器解析的字符数据，但有些字符需要借助实体才可以被解析器解析

- 示例：

```
<sometag>&lt;sometagcontent&gt;</sometag>
```

- CDATA

- Character Data

- 不可以被XML文档解析器解析的数据，按字符原样输出

- 示例：

XML
</p> ¨K10K

  - 实体

  - 实体的使用：&实体名;

- 定义格式：

```
<!DOCTYPE courses[<!ENTITY company "mihoyo.co.td">]>
```

问题

- 良构的XML与有效的XML
- 有效的XML是指满足XML的语法规则，还要满足DTD或者XML Schema的限制
- 有效的一定是良构的，但是良构的不一定是有效的
- XML与HTML
- HTML标签是预先定义的，而XML中的标签我们可以自行定义
- HTML主要关注文档内容的展示，而XML主要关注文档的结构、语义
- HTML语法检查不严格，XML严格
- 都是标记语言，都是自SGML发展而来都是W3C维护的国际标准

DTD

什么是DTD

DTD：Document Type Difinition，文档类型定义用来对XML文档内容进行限制，如：XML中可以出现的元素的名字、元素出现的顺序、元素出现的频率等。

在XML中引入dtd

- XML和DTD在不同文件中

```
<!DOCTYPE rootElement SYSTEM "path">
```

- XML和DTD在同一文件中

```
<!DOCTYPE rootElement[...]>
```

- DTD 放在互联网上

```
<!DOCTYPE rootElement PUBLIC "describe" "URL>
```

语法

元素声明

语法:

```
<!ELEMENT elementName (contentModle)>
```

- contentModle: 内容模式，用来限制元素中可以跟什么样的内容
- #PCDATA:限定元素内容为可以解析的字符内容
- EMPTY：限定元素内容只能为空
- ANY 限定元素内容为任意内容（空、文本、子元素、混合）
- elements：限定元素内容只能为子元素
- ，表示子元素按顺序出现
- | 表示多个子元素中选一个
- 无标识 表示子元素必须要出现1次
- + 表示子元素可以出现1次或多次
- * 表示子元素可以出现0次或多次
- ？ 表示子元素因为已出现0或1次
- mixed 限定元素内容为文本、子元素。

属性声明

语法:

1)

```
<!ATTLIST elementName attName attType attDefault>
...
<!ATTLIST elementName attName attType attDefault>
```

2)

```
<!ATTLIST elementName
```

```
  attName attType attDefault
  ...
  attName attType attDefault
>
```

- attType
- ID:限定属性唯一，而且取值要满足XML的命名规则
- enumeration
  - 限定属性取值在一定范围内
- CDATA 限定属性取值为字符内容
- attDeault
- #REQUIRED：限定元素出现，属性那个就必须出现
- #MPLIED： 限定元素出现，属性可以出现也可以不出现，如果属性不出现，属性没有值
- value:限定元素出现，属性可以出现也可以不出现；如果属性不出现，属性值为所指定的value值
- #FIXED:限定元素出现，属性可以出现也可以不出现；如果属性出现，其取值必须是所指定的值

实体

DTD的不足

- 不支持具体的数据类型
- DTD语法和XML语法不兼容
- 不能进行具体的限制

XML Schema

命名空间

NameSpace：命名空间，元素、属性名称的集合，解决命名冲突问题

命名空间使用注意

- 定义命名空间前缀，并应用在元素上，那么该元素下所有的子元素和属性都在该命名空间
- 可以用xmlns定义默认命名空间，默认命名空间只作用于元素，不作用于属性
- 在同一标签中可以定义多个命名空间前缀
- 多个命名空间前缀不能重名
- 多个命名空间前缀可以指向同一URI
- 判断元素、属性是否相同，除了要看命名空间前缀和元素名/属性名外，还要看命名空间前缀所指向的URI是否相同

XML Schema简介

XML Shema和DTD一样，用来对XML文件的内容进行限制
XML Schema中提供了一些内置数据类型，而且还允许用户自己定义Simple Type(简单类型),Complex Type(复杂类型)

XML Schema语法

Simple Type

可以用来修饰、限制元素和属性，不能携带属性

```
<simpleType name="typeName">
 <restriction base="type">
  ...
 </restriction>
</simpleType>
```

ComplexType

只能用来修饰、限制元素，可以携带属性

```
<complexType name="">
 ...
</complexType>
```

内容模式

```
<!--1)Simple:只能有字符内容和属性-->
<briup id="100">Test</briup>
 <complexType name="">
  <simpleContent>
   ......
  </simpleContent>
 </complexType>

<!--2)Empty:只能有属性-->

<briup id="100"></briup>
<briup id="100"/>
 <complexType name="">
  <attribute ...>...
 </complexType>

<!--3)Element Only:只能有子元素和属性-->

<briup>
 <first_name></first_name>
 <last_name></last_name>
</briup>
<!--complexElement.xsd-->
 <complexType name="">
  <sequence>
   <element>...
  </sequence>
  <attribute>...
 </complexType>

<!--4)Mixed:可以出现子元素、字符内容、属性-->

<briup id="">
 <first_name></first_name>
 <last_name></last_name>
  Hello
</briup>
 <complexType name="" mixed="true">
  <sequence>
   <element>...
  </sequence>
  <attribute>...
 </complexType>
```

Schema 元素声明

语法:

```
<element name="elementName" type="elementType" minOccus="" maxOccus="">...</element>
```

minOccus:限定元素最少出现次数
maxOccus:限定元素最多出现次数

属性名

语法：

```
<attribute name="attName" type="attType" use="" default="" fixed="">
</attribute>
```

use取值:
1)optional:表示属性可有可无
2)required:表示属性必须要有
default:属性不出现时的默认值
fixed:属性可以出现，也可以不出现，如果属性出现，值一定是fixed后指定的值