XML 原理与应用

Summer X

RUC

CH3 DTD

本章学习目标

- 了解 DTD 的作用
- 掌握 DTD 的语法规则和使用方法
- 能够利用 XML 编辑工具实现 DTD 的编辑和验证

目录 I

目录 II

3.1 DTD 的作用

老师请每一个同学把最近一年看过的 5 本书写出来，并进行汇总统计，供大家交流学习，要求大家采用 XML 进行表示，图书信息暂时只包含作者和书名。由于 XML 表示方式非常自由，不同同学的表示结果很可能不同。

第 1 个同学的 XML 文档

```xml
<?xml version="1.0"?>
<book>
    <name author="罗贯中">三国演义</name>
    <name author="曹雪芹">红楼梦</name>
    <name author="施耐庵">水浒传</name>
    <name author="吴承恩">西游记</name>
    <name author="荷马">荷马史诗</name>
</book>
```

第 2 个同学的 XML 文档

```xml
<?xml version="1.0"?>
<books>
    <book title="神曲" author="但丁"/>
    <book title="哈姆雷特" author="莎士比亚"/>
    <book title="浮士德" author="歌德"/>
    <book title="哈克贝利芬历险记" author="马克.吐温"/>
    <book title="少年维特之烦恼" author="歌德"/>
</books>
```

问题

- 不同同学所采用的描述方式却不尽相同，这使得后续的自动汇总处理难以实现
- 在实际应用中，“格式良好”仅是一个基本要求，还需要满足一些额外的约束
- DTD 通过独特的语法，规定了 XML 文档编写应遵守的约束
- DTD 示例：

```
1 <!ELEMENT books (book)+>
2 <!ELEMENT book EMPTY>
3 <!ATTLIST book
4    title CDATA #REQUIRED
5    author CDATA #REQUIRED
6 >
```

3.2 DTD 的关联方式

- 内部 DTD
- 外部 DTD
- 公用 DTD
- 内外结合关联方式

内部关联方式

```
<?xml version="1.0"?>
<!DOCTYPE books [
   <!ELEMENT books (book)+>
   <!ELEMENT book EMPTY>
   <!ATTLIST book
      title CDATA #REQUIRED
      author CDATA #REQUIRED
   >
]>
<books>
   <book title="神曲" author="但丁"/>
   <book title="哈姆雷特" author="莎士比亚"/>
   <book title="浮士德" author="歌德"/>
   <book title="哈克贝利芬历险记" author="马克.吐温"/>
   <book title="少年维特之烦恼" author="歌德"/>
</books>
```

外部关联方式

- 当对结构相同的多个 XML 文档进行验证时，使用外部 DTD 是最为合适的选择方案
- 既可以使 DTD 在多个文档中得到复用，方便维护管理，也有利于数据的传输。

XML 文档

```
1 <?xml version="1.0"?>
2 <!DOCTYPE books SYSTEM "3-2.dtd">
3 <books>
4     <book title="神曲" author="但丁"/>
5     <book title="哈姆雷特" author="莎士比亚"/>
6     <book title="浮士德" author="歌德"/>
7     <book title="哈克贝利芬历险记" author="马克吐温"/>
8     <book title="少年维特之烦恼" author="歌德"/>
9 </books>
```

DTD 文档

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT books (book)+>
<!ELEMENT book EMPTY>
<!ATTLIST book
   title CDATA #REQUIRED
   author CDATA #REQUIRED
>
```

公用 DTD

- 关键字 SYSTEM 并非关联外部 DTD 文档的唯一方式，SYSTEM 关键字适用于在私有组织内部或个人多个文档之间对 DTD 的共享使用。使用 PUBLIC 关键字的公用 DTD 是另一种常见的关联方式
- XML 文档与公用 DTD 的关联方式
  - <!DOCTYPE 根元素名称 PUBLIC FPI "DTD_URL">
  - FPI: Formal Public Identifier，规范化公共标识符

初始化字符串//DTD 所有者名称//DTD 描述//语言标识符

- 初始化字符串
  - 如果 DTD 是一个 ISO 标准，则 DTD 名称以字符串“ISO”开始；
  - 如果 DTD 由某个非 ISO 组织所同意，那么名称以“+”开始；
  - 如果没有标准化组织同意该 DTD，则名称以“-”开始。
- 例子

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3
4 <!DOCTYPE CONTACTS PUBLIC "-//xiatian//Contact Data//EN"
5         "http://www.mydomain.com/dtds/contacts.dtd">
```

内外结合方式

- 内部 DTD 和外部 DTD 合并使用

E.g. 外部 DTD

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!ELEMENT universities (university)+>
3 <!ELEMENT university (#PCDATA)>
4 <!ENTITY RUC "中国人民大学">
```

E.g. XML 文档

```
1 <?xml version="1.0"?>
2 <!DOCTYPE universities SYSTEM "3-5.dtd"[
3   <!ENTITY RUC "人民大学">
4 ]>
5 <universities>
6   <university>&RUC;</university>
7   <university>北京大学</university>
8   <university>清华大学</university>
9 </universities>
```

- 同时使用了外部 DTD 和内部 DTD 定义方式
- 内部 DTD 会覆盖外部 DTD 已有定义

3.3 DTD 元素

- 元素类型声明
- 空元素
- 文本类型元素
- 元素内容模型与混合内容元素

3.3.1 元素类型声明

语法：<!ELEMENT 元素名称元素内容规范 >

- ELEMENT 作为关键字，用于对指定元素进行声明
- ELEMENT 单词必须全部大写
- ELEMENT 与“<!”之间不能留有空格
- 元素内容规范指明了元素内可以包含的内容形式
  - 空元素（EMPTY）
  - 任意元素（ANY）
  - 父元素
  - 文本元素
  - 混合元素

Example

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!ELEMENT books ANY>
3 <!ELEMENT book (title, author, publish)>
4 <!ELEMENT title (#PCDATA)>
5 <!ELEMENT author (#PCDATA)>
6 <!ELEMENT publish (#PCDATA|ISBN)*>
7 <!ELEMENT ISBN EMPTY>
```

3.3.2 空元素

- 语法

```
<!ELEMENT 元素名 EMPTY>
```

- 例子

```
1 <!ELEMENT 人物 EMPTY>
2 <!ELEMENT IMG EMPTY>
```

- 使用场景
  - 元素把全部数据放在属性里，而元素本身没有任何 PCDATA 内容
  - 例如：

```
1 <人物 姓名="林冲" 籍贯="东京" 职业="强盗" 特长="枪棒"/>
2 <img src="baby.jpg" alt="baby" title="宝贝的照片"/>
```

3.3.3 文本类型元素

- 语法：

```
<!ELEMENT 元素名 (#PCDATA) >
```

- 例子

```
<!ELEMENT 姓名 (#PCDATA)>
<!ELEMENT 籍贯 (#PCDATA)>
```

  - #PCDATA 为元素数据的指定类型
    - Parsed Character Data：可解析字符数据
    - 元素把全部数据放在属性里，本身没有任何 PCDATA 内容
- DTD 类型限制
  - 不能区分文本和数字, 无法区分整数和实数
  - 不能定义用户数据类型
  - 不能限制数据的取值范围

以下定义方式是否正确?

```
1 <!ELEMENT 姓名 #PCDATA>
2 <!ELEMENT 姓名 PCDATA>
```

3.3.4 元素内容模型与混合内容元素

元素内容模型 ECM（Element Content Models）就是用来描述元素内具体可以包含哪些子元素的一组正则表达式

元字符	用法	范例
,	将各个元素依指定顺序排列	(A,B,C)
字符串	一个元素仅能出现一次	(A)
*	元素可以出现 0 次或 0 次以上	A*
+	元素可以出现 1 次或 1 次以上	A+
?	一个元素可以出现 0 次或一次	A?
( )	分组符号，常结合“,”或“|”使用	(A,B)
|	逻辑或，只能出现符号作用范围的任一元素 1 次	(A|B|C)

Table : 元素内容模型的元字符以及相应用法与范例

具有严格顺序的子元素序列

对内容规范为逗号分隔的子元素序列，文档必须严格按照规定的顺序来书写子元素

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE person [
   <!ELEMENT person (name, telephone, email)>
   <!ELEMENT name (#PCDATA)>
   <!ELEMENT email (#PCDATA)>
   <!ELEMENT telephone (#PCDATA)>
]>
<person>
   <name>Summer</name>
   <email>summer@ruc.edu.cn</email>
   <telephone>010-82500673</telephone>
</person>
```

可重复元素

- 若某子元素名后跟有星号“*”或加号“+”，则该元素可以重复若干次
- + 代表出现 1 到多次，即至少出现一次
- * 代表出现 0 到多次，即可以不出现

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE person [
   <!ELEMENT person (name, telephone, email+)>
   <!ELEMENT name (#PCDATA)>
   <!ELEMENT email (#PCDATA)>
   <!ELEMENT telephone (#PCDATA)>
]>
<person>
   <name>Summer</name>
   <telephone>010-82500673</telephone>
   <email>summer@ruc.edu.cn</email>
   <email>xia@ruc.edu.cn</email>
</person>
```

分组与逻辑或

- 逻辑或“|”用于多选一情况，经常与分组符号“()”结合使用
- Eaxmple：

```
<!ELEMENT person (name, (telephone | email) )>
```

  - 元素 person 的子元素 name 后面只能有一个 telephone 元素或者 email 元素，而不能两个都有。
  - 错误写法：

```
<!ELEMENT person (name, telephone | email )>
```

- 以下 DTD 语句的含义是什么？

```
<!ELEMENT person (name, (telephone | email | (telephone, email) ) )>
```

可选元素

- 子元素后面跟有问号符号“?”, 属于可选元素，可以出现 0 次或 1 次
- 当无法确定一个元素是否会出现而且就算出现也只出现一次，可以使用该符号
- E.g.

```
<!ELEMENT person (name, (areaCode?, telephone), email )>
```

  - person 元素中的联系电话部分必须出现 telephone 子元素，而区号 areaCode，则可以出现，也可以不出现

混合内容

- 假设采用 XML 对人物信息进行描述，描述内容采用文本形式，但同时希望文本内出现的人物姓名、籍贯、职业等信息采用元素方式进行结构化表示，也就是说，子元素可以任意散布在一段段的文本之中。
- 可采用混合内容描述方法：

```
<!ELEMENT 元素名 (#PCDATA | 子元素1 | 子元素2 | ... | 子元素n)* >
```

混合内容示例

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE 人物 [
   <!ELEMENT 人物 (#PCDATA|姓名|籍贯|职业|特长)* >
   <!ELEMENT 姓名 (#PCDATA)>
   <!ELEMENT 籍贯 (#PCDATA)>
   <!ELEMENT 职业 (#PCDATA)>
   <!ELEMENT 特长 (#PCDATA)>
]>
<人物>
   <姓名>林冲</姓名>，<籍贯>东京</籍贯>人氏，人称豹子头，
官方认定其为水泊梁山<职业>强盗</职业>。
生性耿直，爱交好汉，擅长<特长>枪棒</特长>。
</人物>
```

- <!ELEMENT 人物 (#PCDATA| 姓名 | 籍贯 | 职业 | 特长)* >

3.4 DTD 属性

- 属性声明
- 属性类型
- 属性的默认形态
- 特殊属性

3.4.1 属性声明

- 语法：

```
<!ATTLIST 元素名 属性名 属性类型 缺省声明 (属性名 属性类型
    缺省声明... ) >
```

  - 元素名指明了属性所属的元素名称
  - 属性名表示指定属性的名称
  - 属性类型决定了属性值应具备的类型，如常用的 CDATA
  - 缺省声明规定对属性值的要求和指定默认值

属性声明示例

- 如下 XML 文档片段如何声明对应的 DTD？

```
<人物 姓名="林冲"/>
```

属性声明示例

- 声明方式：

```
1 <!ELEMENT 人物 EMPTY>
2 <!ATTLIST 人物 姓名 CDATA  #REQUIRED >
```

  - <!ELEMENT> 标记声明“人物”元素为一个空元素
  - <!ATTLIST> 标记表明了“人物”元素拥有属性“姓名”，
  - 人物属性类型为 CDATA
  - 人物属性使用时必须指明。

属性简写方式

- XML:

```
<人物 姓名="林冲" 籍贯="东京" 职业="强盗" 特长="枪棒"/>
```

- 属性逐个声明方式：

```
1 <!ELEMENT 人物 EMPTY>
2 <!ATTLIST 人物   姓名 CDATA #REQUIRED >
3 <!ATTLIST 人物   籍贯 CDATA #IMPLIED >
4 <!ATTLIST 人物   职业 CDATA #IMPLIED >
5 <!ATTLIST 人物   特长 CDATA "武术" >
```

- 属性一起声明方式：

```
1 <!ATTLIST 人物   姓名 CDATA #REQUIRED
2           籍贯 CDATA #IMPLIED
3           职业 CDATA #IMPLIED
4           特长 CDATA "武术">
```

3.4.2 属性类型

类型	含义
CDATA	属性值为任意合法的字符串，最为常用的属性类型
Enumerated	从若干个指定值列表中选择正确的值
ID	属性值必须唯一，不能与中其他 ID 类型属性取相同值
IDREF	属性值引用了某一个已经指定的 ID 类型的属性值
IDREFS	属性值由多个 IDREF 类型的属性值组成，并由空格隔开
NMTOKEN	Name Token，由符合标记命名要求的字符串组成
NMTOKENS	该属性值是由空格分隔的一系列 NMTOKEN 的集合
ENTITY	属性值为已经由 DTD 声明的实体名称
ENTITIES	属性值是由空格分隔的一系列 ENTITY 的集合
NOTATION	属性值必须匹配 NOTATION 名称列表中的某个名称

Table : DTD 的属性类型

CDATA 属性类型

- 字符数据（Character DATA），是最基本的属性类型
- 属性值为不包含小于号“<”和“”的任意文本字符串
  - 如出现“<”和“&”，需要转义

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <!DOCTYPE todolist [
3    <!ELEMENT todolist (todo)*>
4    <!ELEMENT todo EMPTY>
5    <!ATTLIST todo content CDATA #REQUIRED>
6  ]>
7  <todolist>
8    <todo content="当股价>100的时候抛出'全聚德""/>
9    <todo content="周六&amp;amp;周日回山东临朐"/>
10 </todolist>
```

Enumerated 枚举类型

- 语法：

```
<!ATTLIST 元素名 属性名 (属性值1 | 属性值2 | ... | 属性值n) 缺省声明>
```

- 例子：

```
<!ATTLIST 交易   货币单位 (人民币 | 日元 | 美元 | 欧元 | 英镑) "人民币">
```

NOTATION 符号类型

```
1 <!ELEMENT 图片 (#PCDATA)>
2 <!NOTATION gif SYSTEM "IMAGE/gif">
3 <!NOTATION jpg SYSTEM "IMAGE/jpeg">
4 <!NOTATION png SYSTEM "IMAGE/png">
5 <!ATTLIST 图片 格式 NOTATION (gif | jpg | png) #REQUIRED>
6
7 <图片 格式="png"/>
```

ID、IDREF 与 IDREFS 属性类型

- ID:
  - 该类型的属性无缺省值，缺省声明必须为 #REQUIRED 或 #IMPLIED
  - 一个元素不能拥有超过一个 ID 类型的属性
- IDREF(ID Reference)
  - 对 ID 的引用，当某个属性引用了另一个 ID 类型的属性值时，可将其声明为 IDREF
- IDREFS(ID References)
  - 多个标识符的引用，属性值引用了多个由空格分隔的 ID 类型的属性值

ID、IDREF 与 IDREFS 示例

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <!DOCTYPE 班级 [
3     <!ELEMENT 班级 (学生)*>
4     <!ELEMENT 学生 (#PCDATA)>
5     <!ATTLIST 班级 名称 CDATA #REQUIRED>
6     <!ATTLIST 班级 班长 IDREF #IMPLIED>
7     <!ATTLIST 班级 班委 IDREFS #IMPLIED>
8     <!ATTLIST 学生 学号 ID #REQUIRED>
9  ]>
10 <班级 名称="政务信息一班" 班长="S001" 班委="S001 S002 S004">
11    <学生 学号="S001">姜海明</学生>
12    <学生 学号="S002">李学健</学生>
13    <学生 学号="S003">方路</学生>
14    <学生 学号="S004">刘杨</学生>
15 </班级>
```

NMTOKEN 与 NMTOKENS 属性类型

- NMTOKEN: Name Token
  - 限定属性值为有效的 XML 名称
- NMTOKENS: Name Tokens
  - 一组 NMTOKEN 类型的字符串，每个 NMTOKEN 之间以空格分隔

```
1 <!ELEMENT 数据 (#PCDATA) >
2 <!ATTLIST 数据   授权用户 NMTOKENS #IMPLIED>
3
4 <数据 授权用户="Lucy Summer
      Macy">如何学好XML的超级技巧.....</数据>
```

ENTITY 与 ENTITIES 属性类型

- ENTITY
  - 限定属性值为已经声明过的实体名称，
- ENTITIES
  - 可以包含多个实体，实体之间通过空格进行分隔。

ENTITY 与 ENTITIES 示例

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE 高校列表 [
3    <!ELEMENT 高校列表 (高校)*>
4    <!ELEMENT 高校 (#PCDATA)>
5    <!ATTLIST 高校 LOGO ENTITY #REQUIRED
6              校园风景 ENTITIES #IMPLIED>
7    <!NOTATION JPG SYSTEM "image/jpeg">
8    <!ENTITY LOGO2 SYSTEM "ruc_logo.jpg" NDATA JPG>
9    <!ENTITY 风景_人大_1 SYSTEM "ruc_1.jpg" NDATA JPG>
10   <!ENTITY 风景_人大_2 SYSTEM "ruc_2.jpg" NDATA JPG>
11 ]>
12 <高校列表>
13   <高校 LOGO="LOGO2" 校园风景="风景_人大_1
       风景_人大_2">人民大学</高校>
14 </高校列表>
```

3.4.3 属性的默认形态

- #REQUIRED
  - 与该属性关联的元素必须为属性指定一个明确的值，不能缺省
- #IMPLIED
  - 可有可无的属性
- #FIXED
  - 固定值，不能更改
- 默认值
  - 属性具有一个指定的默认值，与 #FIXED 不同，用户可以通过设置元素的属性值，改变其默认值。

FIXED 与默认值

```
1 <!ATTLIST person gender CDATA #FIXED "男">
2
3 <person gender="男"/>    <!-- 正确 -->
4 <person/>                <!-- 正确，person拥有属性gender，其值为男-->
5 <person gender="女"/>    <!-- 非法使用-->
```

```
1 <!ATTLIST person gender CDATA "男">
2
3 <person gender="男"/>    <!-- 正确 -->
4 <person/>                <!-- 正确，person拥有属性gender，其值为男-->
5 <person gender="女"/>    <!-- 正确，属性gender值为女-->
```

3.4.4 XML 特殊属性

- XML 为所有元素预留了三个特殊属性
  - xml:lang —语种选择
  - xml:space —白空符处理
  - xml:id —标识符

3.5 DTD 实体

实体有利于快速更替特定文字，简化编写和方便修改

- 实体类型与实体引用
- 内部可解析通用实体
- 外部可解析通用实体
- 外部非解析通用实体
- 内部参数实体
- 外部参数实体

3.5.1 实体类型与实体引用

- 1. 通用实体和参数实体
  - 通用实体: 实体只能在 XML 文档中被引用，不能在 DTD 中引用
  - 参数实体: 实体只能在 DTD 中被引用，不能在 XML 文档中引用
- 2. 内部实体和外部实体
  - 内部实体: 实体值以内嵌方式定义
  - 外部实体: 实体值包含在外部资源中
- 3. 可解析实体和非解析实体
  - 可解析实体: 实体值被 XML 处理程序解析为 XML/DTD 文档
  - 非解析实体: 实体值不能被 XML 处理程序解析

五种 DTD 实体类型

- 内部可解析通用实体

```
<!ENTITY name ”value” >
```

- 外部可解析通用实体

```
<!ENTITY name SYSTEM URI > ......
```

- 外部非解析通用实体

```
<!ENTITY name SYSTEM URI NDATA type> ......
```

- 内部参数实体

```
<!ENTITY \% name ”value” >
```

- 外部参数实体

```
<!ENTITY \% name SYSTEM URI > ......
```

实体引用方法

- 可解析通用实体

```
&实体名称;
```

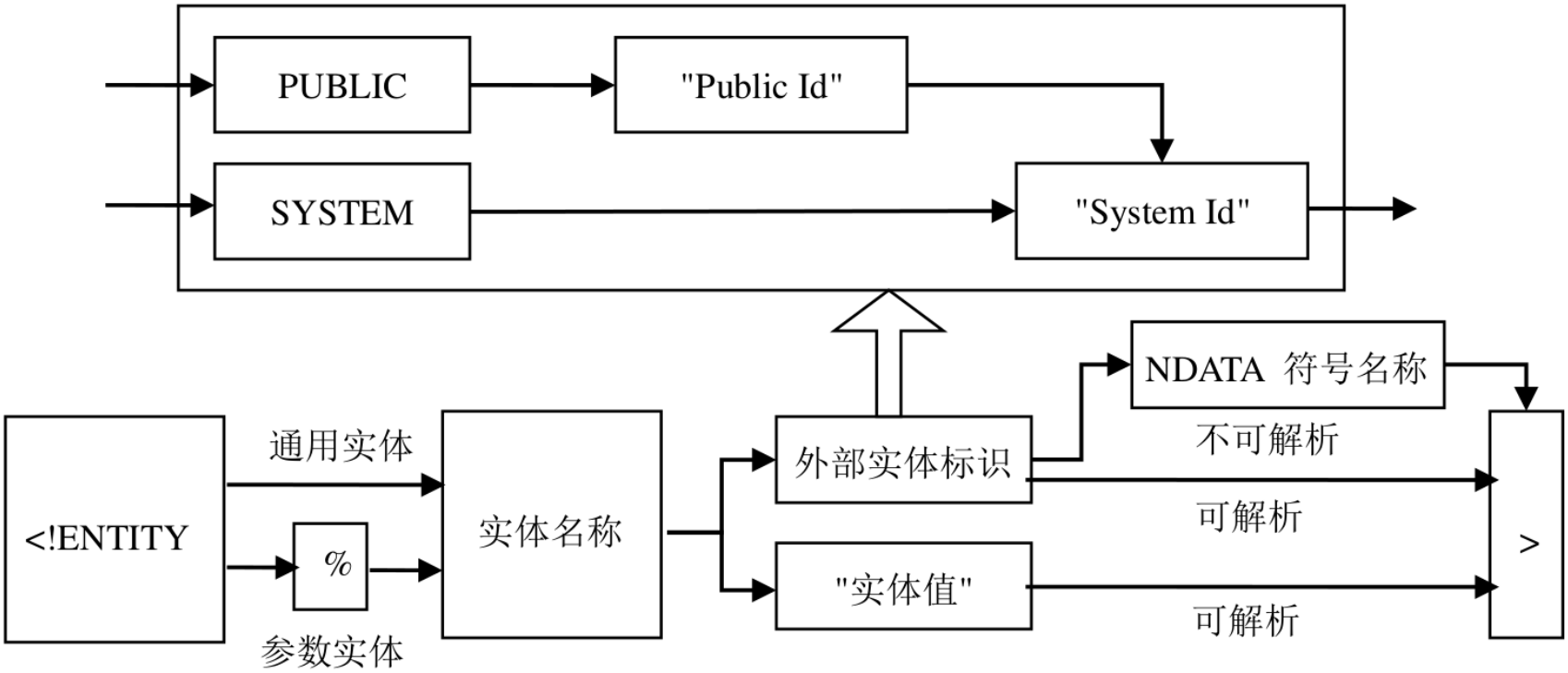
- 非解析通用实体

```
属性名称="实体名称"（属性限定为ENTITY或ENTITIES类型）
```

- 参数实体

```
\%实体名称;
```

实体语法示意图


PUBLIC
"Public Id"
SYSTEM
"System Id"
<!ENTITY
通用实体
%
参数实体
实体名称
外部实体标识
"实体值"
NDATA 符号名称
不可解析
可解析
可解析
>

3.5.2 内部可解析通用实体

- 语法形式：

```
<!ENTITY 实体名称 "实体值" >
```

- 例子

```
<?xml version = "1.0" encoding="utf-8"?>
<!DOCTYPE 数据 [
  <!ELEMENT 数据 (#PCDATA)>
  <!ENTITY 版权声明 "自由信息，由人民大学编写">
]>
<数据>&版权声明;</数据>
```

示例

- 可在实体的定义之中引用已定义实体

```
<!ENTITY 机构 "人民大学">
<!ENTITY 版权声明 "自由信息，由&机构;编写">
```

- 不可以相互引用，如：

```
<!ENTITY 机构 "人民大学, &版权声明;">
<!ENTITY 版权声明 "自由信息，由&机构;编写">
```

- 不能针对 DTD 的元素定义采用通用实体引用

```
<!ENTITY 教师基本信息 "(姓名, 性别, 出生年月, 研究方向)">
<!ELEMENT 教授 &教师基本信息;>
<!ELEMENT 副教授 &教师基本信息;>
<!ENTITY 讲师 &教师基本信息;>
```

3.5.3 外部可解析通用实体

- 语法形式：

```
<!ENTITY 实体名称 SYSTEM "系统标识符" >
```

```
<!ENTITY 实体名称 PUBLIC FPI "公共标识符" >
```

- 例子

```
1 <?xml version = "1.0" encoding="utf-8"?>
2 <!DOCTYPE 数据 [
3   <!ELEMENT 数据 (#PCDATA)>
4   <!ENTITY 介绍数据 SYSTEM "intro.txt">
5 ]>
6 <数据>&介绍数据;</数据>
```

3.5.4 外部非解析通用实体

- 外部可解析通用实体可以把外部文件中的文本内容与 DTD 实体建立关联，并被 XML 解析器读取处理；
- 对于图片、音频、视频等二进制文件，如果在实体引用时仍然被解析，则会因特殊字符的存在而导致语法错误，为此，W3C 引入了非解析实体（Unparsed Entity）概念
- 外部非解析通用实体的语法形式：

```
<!ENTITY 实体名称 SYSTEM "系统标识符" NDATA Type>
```

```
<!ENTITY 实体名称 PUBLIC FPI "公共标识符" NDATA Type>
```

  - Type 代表实体类型，为已声明的 NOTATION

外部非解析通用实体示例

```
1 <?xml version = "1.0" encoding="utf-8"?>
2 <!DOCTYPE 数据 [
3     <!ELEMENT 数据 (#PCDATA)>
4     <!NOTATION JPG SYSTEM "image/jpeg">
5     <!ENTITY 插图 SYSTEM "ruc_1.jpg" NDATA JPG>
6     <!ATTLIST 数据 插图 ENTITY #IMPLIED>
7 ]>
8 <数据 插图="插图">外部非解析通用实体示例</数据>
```

3.5.5 内部参数实体

- 引入参数实体原因
  - 通用实体虽然可以在 DTD 中声明，但却无法在 DTD 中加以引用
- 参数实体一定是可解析实体
  - 内部参数实体
  - 外部参数实体
- 语法形式：

```
<!ENTITY % 实体名称 "实体值" >
```

  - 参数实体定义中的“%”前后有空格

内部参数在内部子集中的声明和引用示例

```
1  <?xml version = "1.0" encoding="utf-8"?>
2  <!DOCTYPE 教工信息 [
3     <!ELEMENT 教工信息 ANY>
4     <!ELEMENT 姓名 (#PCDATA)>
5     <!ELEMENT 专业 (#PCDATA)>
6     <!ENTITY % 教授信息 "<!ELEMENT 教授 (姓名, 专业)>">
7     <!ENTITY % 讲师信息 "<!ELEMENT 讲师 (姓名, 专业)>">
8     %教授信息;
9     %讲师信息;
10 ]>
11 <教工信息>
12    <教授><姓名>钱钟书</姓名> <专业>国学</专业></教授>
13 </教工信息>
```

- 思考 :"教授"和"讲师"元素具备相同的声明内容，是否可以把这部分内容单独声明为参数实体，并在元素声明中加以引用？

内部参数在内部子集中的声明和引用示例

```
1  <?xml version = "1.0" encoding="utf-8"?>
2  <!DOCTYPE 教工信息 [
3     <!ELEMENT 教工信息 ANY>
4     <!ELEMENT 姓名 (#PCDATA)>
5     <!ELEMENT 专业 (#PCDATA)>
6     <!ENTITY % 教师基本信息 "(姓名, 专业)">
7     <!ELEMENT 教授 %教师基本信息;>
8     <!ELEMENT 讲师 %教师基本信息;>
9  ]>
10 <教工信息>
11    <教授><姓名>钱钟书</姓名> <专业>国学</专业></教授>
12 </教工信息>
```

- 禁止在 DTD 内部子集中使用这种方式

内部参数实体在外部子集中的声明和引用示例

- DTD 文档：3-18.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY % 教师基本信息 "(姓名, 专业)">
<!ELEMENT 教授 %教师基本信息;>
<!ELEMENT 讲师 %教师基本信息;>
```

- XML 文档：3-18.xml

```
<?xml version = "1.0" encoding="utf-8"?>
<!DOCTYPE 教工信息 SYSTEM "3-18.dtd"[
    <!ELEMENT 教工信息 ANY>
    <!ELEMENT 姓名 (#PCDATA)>
    <!ELEMENT 专业 (#PCDATA)>
    <!ENTITY % 教师基本信息 "(专业,姓名)">
]>
<教工信息>
    <教授>
```

3.5.6 外部参数实体

- 外部参数实体用于包含来自外部资源的声明
  - 参数实体总是可解析的，一个到外部参数实体的引用（% 参数实体名称;）将被替换为解析后的内容
- 外部参数实体的语法

```
<!ENTITY % 实体名称 SYSTEM "系统标识符">
<!ENTITY % 实体名称 PUBLIC FPI "公共标识符">
```

- 作用
  - 可以把多个较小的 DTD 文件组成一个更大的 DTD 声明，或者把一个大的 DTD 分解为小的、更便于管理的模块，方便理解和重复使用

外部参数实体示例 I

- 外部 DTD 文件“3-20_1.dtd”

```
1 <!ELEMENT 外观 (颜色, 体积)>
2 <!ELEMENT 颜色 (#PCDATA)>
3 <!ELEMENT 体积 (#PCDATA)>
```

- 外部 DTD 文件“3-20_2.dtd”

```
1 <!ELEMENT 联系方式 (客服电话, 网址)>
2 <!ELEMENT 客服电话 (#PCDATA)>
3 <!ELEMENT 网址 (#PCDATA)>
```

外部参数实体示例 II

- XML 文件“3-20.xml”

```
1  <?xml version = "1.0" encoding="utf-8"?>
2  <!DOCTYPE 家具 [
3      <!ELEMENT 家具 (名称, 外观, 联系方式)>
4      <!ELEMENT 名称 (#PCDATA)>
5      <!ENTITY % 外观 SYSTEM "3-20_1.dtd">
6      <!ENTITY % 联系方式 SYSTEM "3-20_2.dtd">
7      %外观;
8      %联系方式;
9  ]>
10 <家具>
11     <名称>布艺沙发</名称>
12     <外观>
13         <颜色>白色</颜色>
14         <体积>3.4*2.55</体积>
15     </外观>
```

外部参数实体示例 III

```
16    <联系方式>
17      <客服电话>400-800-1234</客服电话>
18      <网址>http://www.example.com/</网址>
19    </联系方式>
20  </家具>
```

3.6 DTD NOTATION

- 用于识别非解析实体格式的名字
- 用于 DTD 引用外部文件数据的情况
  - 图像、声音等二进制文件可能包含各种 XML 无法处理的特殊符号，此时需要借助于外部的应用程序进行处理，这类情况的解释描述可以通过 NOTATION 实现
- DTD NOTATION 的语法格式

```
<!NOTATION 名称 SYSTEM "系统标识">
<!NOTATION名称 PUBLIC "公共标识" >
```

- 例子

```
1 <!NOTATION gif SYSTEM "image/gif">
```

3.7 DTD 的包含与忽略

- INCLUDE
- IGNORE
- 语法格式

```
<!INCLUDE [
   此处为DTD声明内容
]]>
<!INGORE [
   此处为DTD声明内容
]]>
```

INCLUDE 与 IGNORE 示例

```
1 <!-- Tables Module ..... -->
2 <!ENTITY % xhtml-table.module "INCLUDE" >
3 <![%xhtml-table.module;[
4 <!ENTITY % xhtml-table.mod
5         PUBLIC "-//W3C//ELEMENTS XHTML Tables 1.0//EN"
6         "http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod" >
7 %xhtml-table.mod;]]>
```

- 行 2 和行 3 展示了 INCLUDE 的使用方式，
- 如不包含该段声明，只需要把第 2 行中的参数实体值更改为 IGNORE 即可
- 行 4 和行 5 则展示了外部参数实体的作用。

END