

硕士研究生课程 《智能信息处理》

语义网基础理论

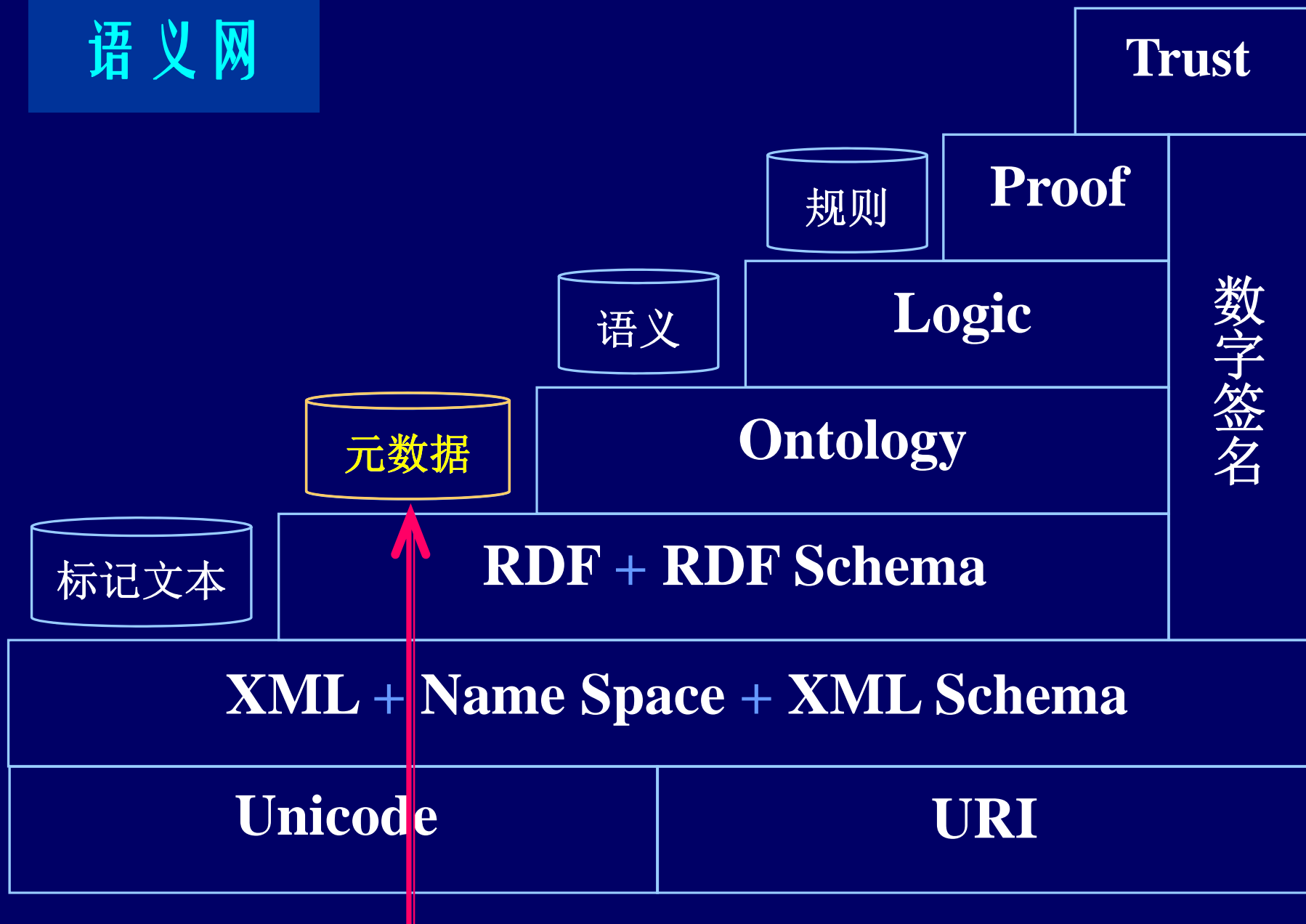
研究生教材

宋炜, 张铭. 语义网简明教程

高等教育出版社, 2004

大连海事大学信息科学技术学院

语义网



元数据

信息(作为属性用于描述资源)

RDF + RDF Schema

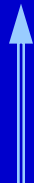
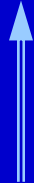
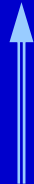
信息(被元数据所标记)

XML + XML Schema

信息(统一字符/统一定位)

Unicode + URI

信息



第2章

元数据

XML

XML Schema

元数据

元数据定义

元数据作用

元数据获取

元数据类型

元数据表示语言

元数据(metadata)



定义：描述数据的数据

性质：便于人/机器快速理解

形式：独立于所描述数据 图书馆目录
包含于所描述数据 书籍的CIP

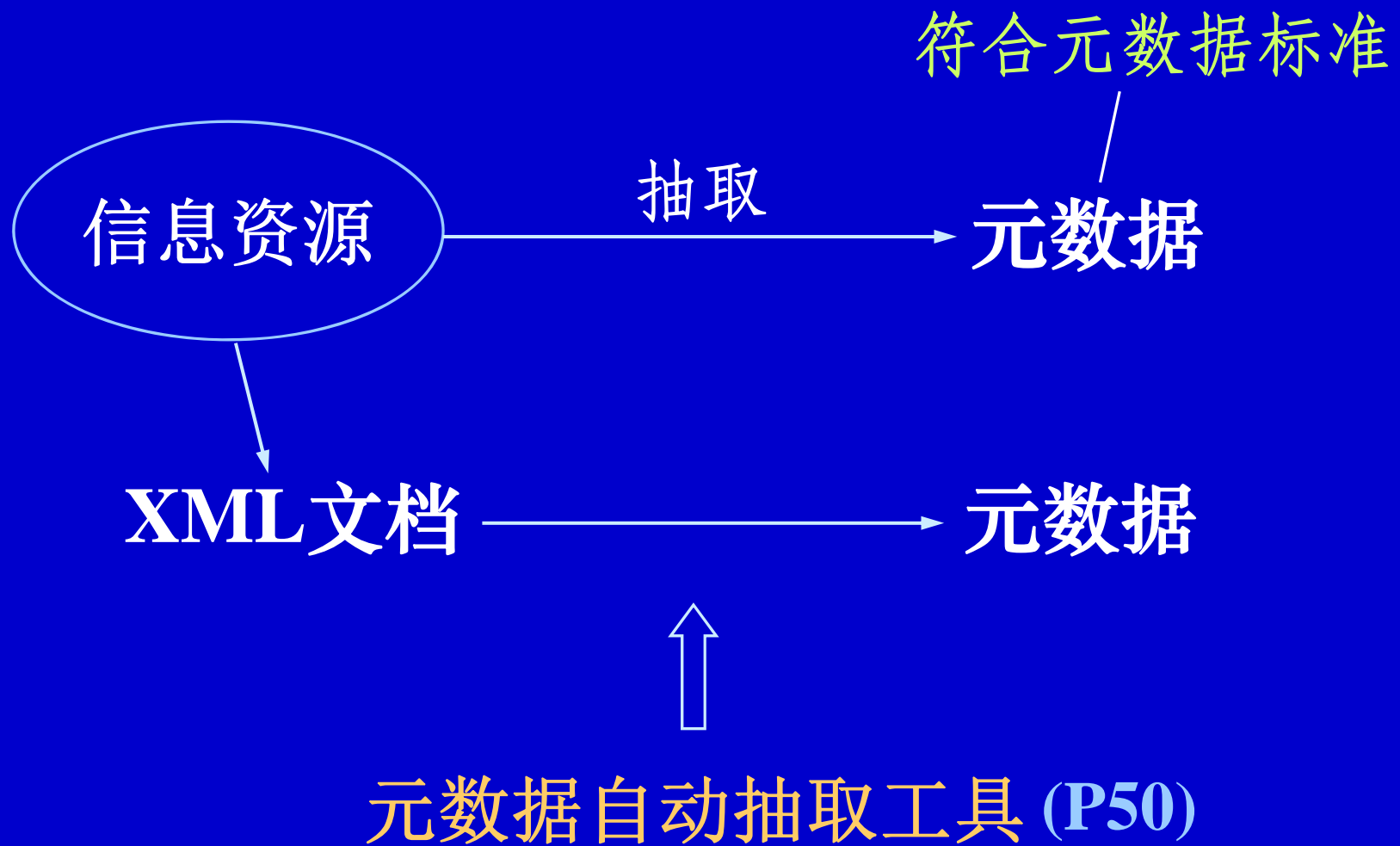
元数据的重要作用 (P4)

- 描述信息资源
- 增强各种资源之间的可交换性
- 提供资源的可访问性
- 沟通不同的数据格式

元数据的主要作用(P32)

- 组织和管理网络信息, 挖掘信息资源
- 帮助用户查询所需信息
- 组织和维护一个机构对数据的投资
- 建立数据目录和数据交换中心
- 提供数据转换方面的信息

元数据的获取(P33)



元数据类型 (P4/P34)

元数据 $\xrightarrow{\text{描述}}$ 信息

内容元数据
管理元数据
参考元数据
载体元数据



信息的内容
信息的历史
信息的链接
信息的外观

元数据表示语言 (基于元数据的标记语言)

SGML

(Standard Generic Markup Language)

HTML

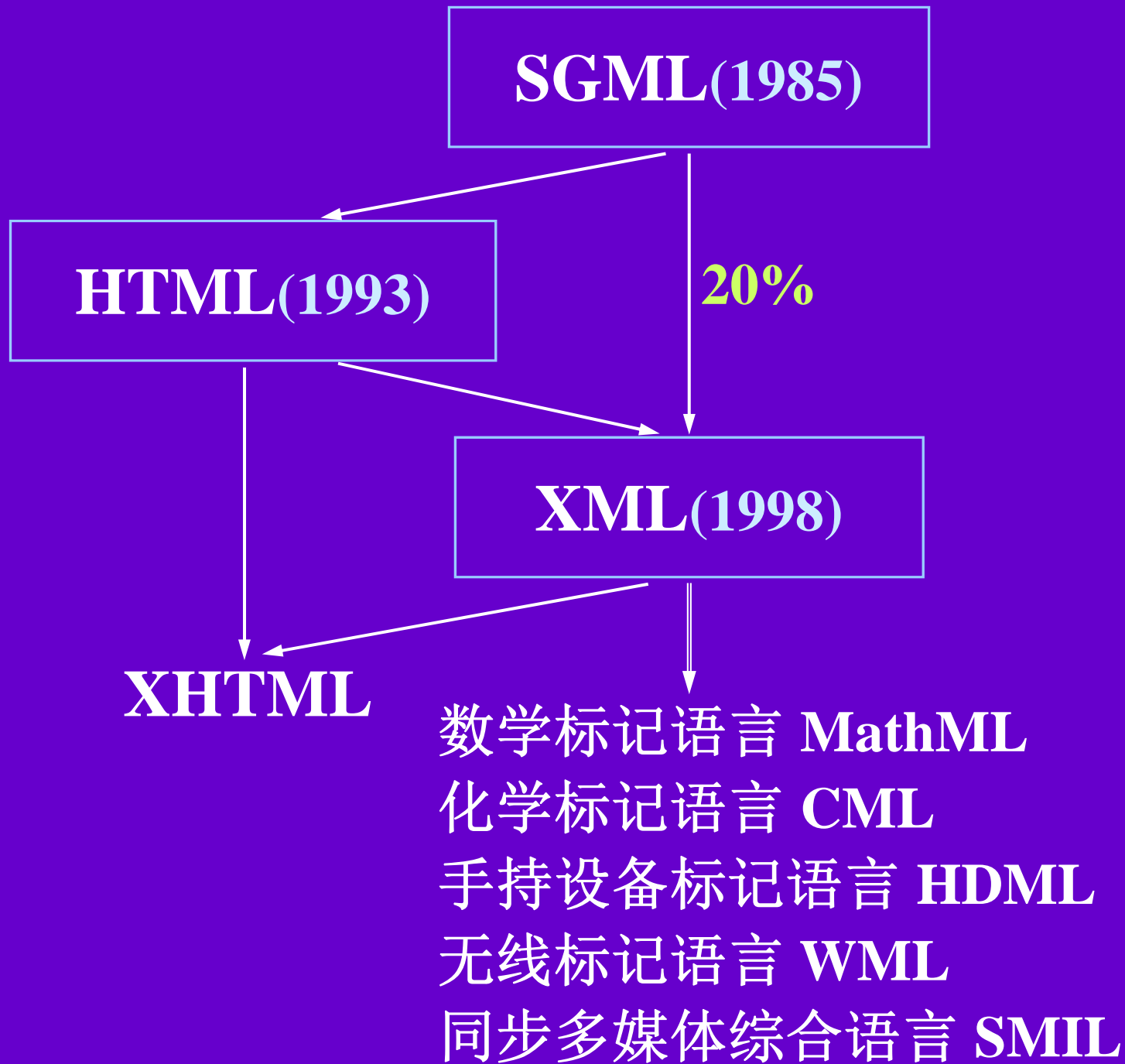
(HyperText Markup Language)

XML

(eXtensible Markup Language)

WML

(Wireless Markup Language)



元数据标准

都柏林元数据核心元素集(DC-3, 1996)

- Title
- Subject
- Description
- Source
- Language
- Relation
- Coverage
- Creator
- Publisher
- Contributor
- Rights
- Date
- Type
- Format
- Identifier

网络信息 ← 浏览器显示 显示格式

超文本标记语言 HTML

可扩展标记语言 XML

用户自定义

被标记对象的意义

数据
结构
显示格式

HTML文档 ■



数据 — **XML文档** ■

结构 — **DTD / XML Schema**

显示格式 — **XSL**

HTML文档：数据与显示格式相连

```
<html>
```

```
  <body>
```

```
    <格式1> 数据1 </格式1>
```

```
    <格式2> 数据2 </格式2>
```

```
    .....
```

```
    <格式n> 数据n </格式n>
```

```
  </body>
```

```
</html>
```


XML文档：数据具有语义、与格式无关

<语义0>

<语义1> 数据1 </语义1>

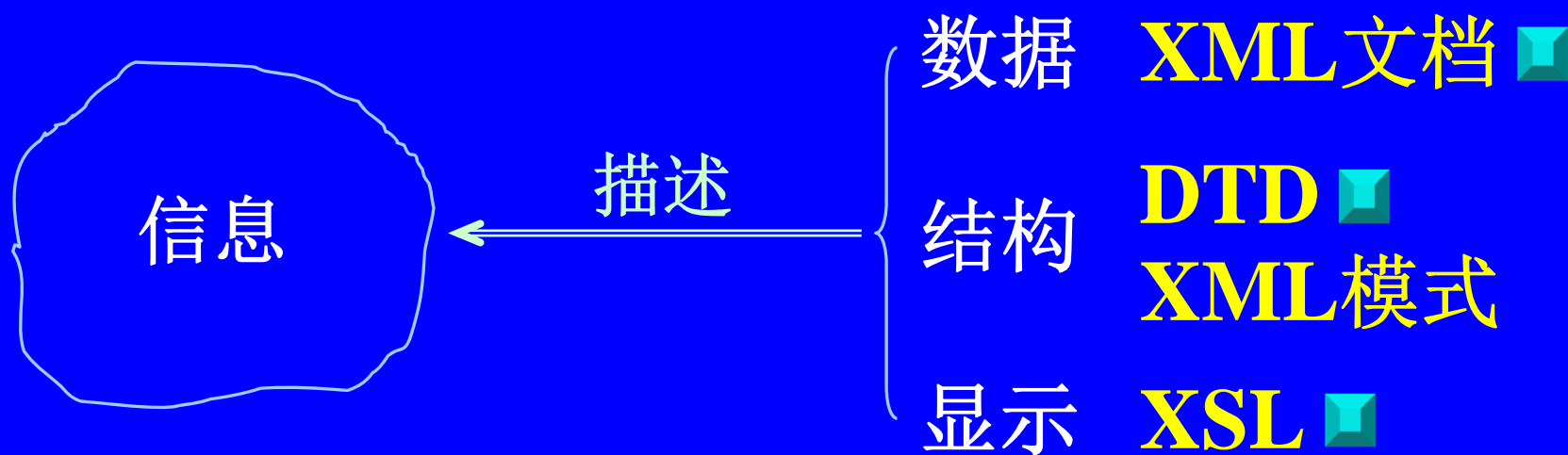
<语义2> 数据2 </语义2>

.....

<语义n> 数据n </语义n>

</语义0>

可扩展标记语言XML的应用



XML文档的结构(P39)

文档头(Prolog)

文档体(Body)

文件尾(Epilog)

XML元素集

并列
嵌套

XML元素(P41)

<标记> 内容 </标记>

<标记 属性="值"> 内容 </标记>

XML文档

文档头

< 标记 属性=“值” >

 < 标记 属性=“值” >

 < 标记 属性=“值” > 内容 < /标记 >

 < 标记 属性=“值” > 内容 < /标记 >

.....

 < /标记 >

.....

< /标记 >

文档尾

XML文档

文档头

< 语义 属性=“值” >

 < 语义 属性=“值” >

 < 语义 属性=“值” > 内容 < /语义 >

 < 语义 属性=“值” > 内容 < /语义 >

.....

 < /语义 >

.....

< /语义 >

文档尾

XML文档举例(P5)

```
< ? XML VERSION="1.0"  
      ENCODING="GB2312"  
      standalone="no" ? >
```

```
< ! DOCTYPE Book SYSTEM  
      http://db.pku.edu.cn/Book.dtd >
```

```
<书>
```

```
  <标题> 《红楼梦》 </标题>
```

```
  <作者> 曹雪芹 </作者>
```

```
  <描述> 中国四大古典名著之一 </描述>
```

```
</书>
```

XML文档举例(P40)

< 书类 >

< 书 国际标准书号=“0345374827” >

<标题> 数据结构 </标题>

<作者> 许卓群等 </作者>

< /书 >

< 书 国际标准书号=“0345374828” >

<标题> 数据挖掘 </标题>

<作者> 韩家炜等 </作者>

< /书 >

< /书类 >

XML的开发工具(P41)

XML编辑器 **XML解析器** **XML浏览器**

文本编辑器

IE

IE

FrontPage

FrontPage

FrontPage

XML Spy

XML Spy

XML Spy

处理指令(P42)

< ? xml version="1.0" encoding="GB2312" standalone="no" ? >

< ? xml-stylesheet type="text-xsl" href="contacts.xsl" ? >

\
hyperlink reference

文档头

XML声明
DTD

文档体

< 语义 属性="值" > 内容 < /语义 >

文档尾

XML文档的结构

计算机程序

主程序
数据
算法

C程序

主程序
声明
语句



```
main()  
{  
  声明  
  语句  
}
```

数据声明
函数声明

XML文档

文档头
文档体
文档尾

XML声明
元素声明

处理指令

DTD / XML Schema

元素: <语义 属性="值">内容</语义>

- 数据声明:**
- (1) 放在内存何处
 - (2) 占多少字节
 - (3) 名称
 - (4) 包括哪些成员(成员占字节, 成员名)
 - (5) 具有这种数据结构的变量名

存储类别 数据类型 类型名 {成员表列} 变量表列;

auto (缺省)
static
extern
register

int/short/long/unsigned
float/double
char
enum
struct
union

a, *a
a[n], *a[n]
(*a)[n], **a

函数声明: (1) 能否被外部文件调用
(2) 返回值类型
(3) 名称
(4) 被调用时传递的参数(参数类型, 参数名)

存储类别 返回值类型 *函数名(形参表列);

存储类别 返回值类型 函数名 (形参表列);

extern(缺省)
static

int (缺省)
short/long/unsigned
float/double
char
enum
void

基本型变量
构造型变量
指针型变量
空类型变量

处理指令举例(P42)

< ? xml version="1.0" encoding="gb2312" standalone="no" ? >

< ? xml-stylesheet type="text-xsl" href="contacts.xsl" ? >

文档头

XML 声明
DTD / XML Schema

文档体

< 语义 属性="值"> 内容 < /语义 >

文档尾

XML文档的结构

DTD / XML Schema的作用

- 定义元素(标记名、内容数据类型)
- 定义元素的属性
- 规定元素出现次数
- 规定元素出现顺序



文档类型定义DTD (document type definitions)

```
<!DOCTYPE 根元素名 [  
    <!ELEMENT 元素名 元素内容描述 >  
    <!ELEMENT 元素名 元素内容描述 >  
    .....  
>
```

EMPTY

ANY — (#PCDATA)

(子元素名|子元素名|...)

(#PCDATA|子元素名|...)

```
<!DOCTYPE 书类 [  
  <!ELEMENT 书 (#PCDATA, 标题, 作者) >  
  <!ELEMENT 标题 (#PCDATA) >  
  <!ELEMENT 作者 (#PCDATA) >  
  <!ELEMENT 书类 (书 +) >  
>
```

```
<书类>
```

```
  <书 国际标准书号="0345374827">
```

```
    <标题> 数据结构 </标题>
```

```
    <作者> 许卓群等 </作者>
```

```
  </书>
```

```
  <书 国际标准书号="0345374828">
```

```
    <标题> 数据挖掘 </标题>
```

```
    <作者> 韩家炜等 </作者>
```

```
  </书>
```

```
</书类>
```

举例

内部DTD和外部DTD (P45)

<! DOCTYPE 根元素名 SYSTEM="外部DTD文件的URL">



XSL (extensible stylesheet language)



```
<?xml version="1.0"?>
```

XML(P43)

```
<食品>
```

```
  <食品 脂肪含量="低">
```

```
    <名称> 月饼 </名称>
```

```
    <描述> 中秋节食用的传统食品 </描述>
```

```
    <价格> 2.99 </价格>
```

```
  </食品>
```

```
</食品>
```

↓
XSL

```
<html>
```

HTML(P43)

```
  <body>
```

```
    <h1> 食品 </h1>
```

```
    <ol>
```

```
      <li> 中秋节的传统食品。月饼2元 </li>
```

```
    </ol>
```

```
  </body>
```

```
</html>
```

```
<?xml version="1.0"?>
<xsl: stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl: template match="/">
    <html>
      <body>
        <h1> hamburgers </h1>
        <xsl: for-each select="hamburgers[ @lowfat="dream on"]">
          <li>
            <xsl: value-of select="name"/>,
            <xsl: value-of select="price"/>
            <xsl: value-of select="description"/>
          </li>
        </xsl: for-each>
      </body>
    </html>
  </xsl: template>
</xsl: stylesheet>
```

XSL举例(P43)

```
<?xml version="1.0"?>
```

```
<食品>
```

```
  <食品 脂肪含量="低">
```

```
    <名称> 月饼 </名称>
```

```
    <描述> 中秋节食用的传统食品 </描述>
```

```
    <价格> 2.99 </价格>
```

```
  </食品>
```

```
</食品>
```

```
<?xml version="1.0" encoding="gb2312" ?>
```

```
<食品>
```

```
  <食品 脂肪含量="低">
```

```
    <名称> 月饼 </名称>
```

```
    <描述> 中秋节食用的传统食品 </描述>
```

```
    <价格> 2.99 </价格>
```

```
  </食品>
```

```
</食品>
```

改正

```
<?xml version="1.0"?>
<xsl: stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl: template match="/">
    <html>
      <body>
        <h1> hamburgers </h1>
        <xsl: for-each select="hamburgers[ @lowfat="dream on"]">
          <li>
            <xsl: value-of select="name"/>,
            <xsl: value-of select="price"/>
            <xsl: value-of select="description"/>
          </li>
        </xsl: for-each>
      </body>
    </html>
  </xsl: template>
</xsl: stylesheet>
```

XSL举例(英文)

```
<?xml version="1.0" encoding="gb2312" ?>
<xsl: stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl: template match="/">
    <html>
      <body>
        <h1> 食品 </h1>
        <xsl: for-each select="食品/食品[@脂肪含量='低']">
          <li>
            <xsl: value-of select="名称"/>,
            <xsl: value-of select="价格"/>,
            <xsl: value-of select="描述"/>
          </li>
        </xsl: for-each>
      </body>
    </html>
  </xsl: template>
</xsl: stylesheet>
```

XSL举例(中文)

```
<?xml version="1.0" encoding="gb2312" ?>
<?xml-stylesheet type="text/xsl" href="p43.xsl"?>
<食品>
  <食品 脂肪含量="低">
    <名称> 月饼 </名称>
    <描述> 中秋节食用的传统食品 </描述>
    <价格> 2.99 </价格>
  </食品>
</食品>
```

XML(P43)

↓ 显示

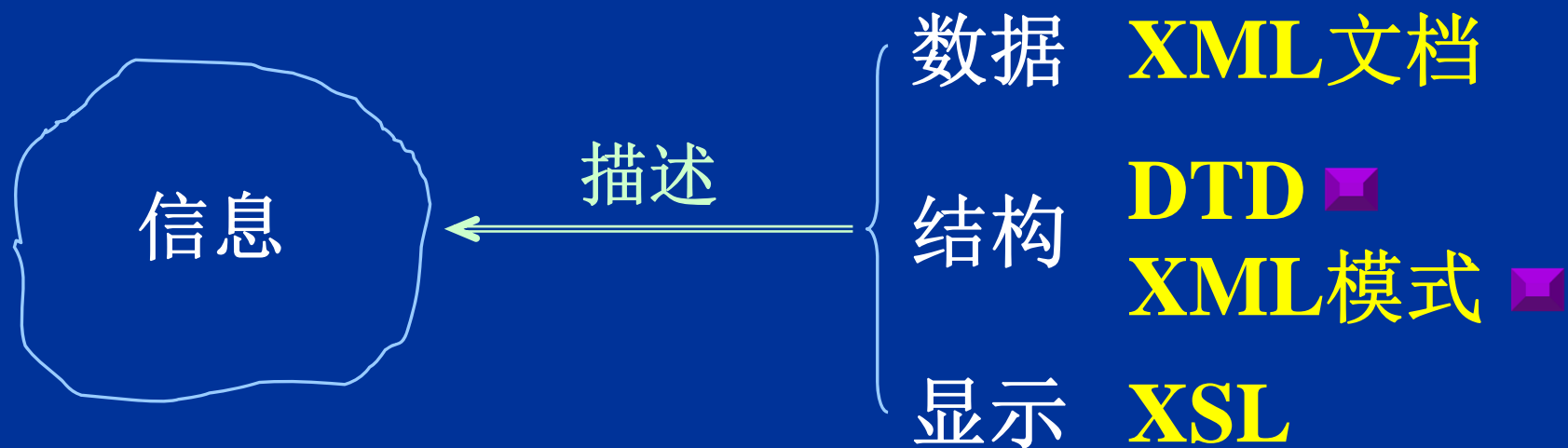
食品

- 月饼, 2.99, 中秋节食用的传统食品


```
<?xml version="1.0" encoding="gb2312" ?>
<xsl: stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl: template match="/">
    <html>
      <body>
        <h1> 食品 </h1>
        <xsl: for-each select="食品/食品[@脂肪含量='低']">
          <li>
            <xsl: value-of select="名称"/>,
            <xsl: value-of select="价格"/>,
            <xsl: value-of select="描述"/>
          </li>
        </xsl: for-each>
      </body>
    </html>
  </xsl: template>
</xsl: stylesheet>
```

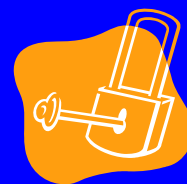
p43.xsl

可扩展标记语言XML的应用



DTD / XML Schema的作用

- 定义元素(标记名、内容数据类型)
- 定义元素的属性
- 规定元素出现次数
- 规定元素出现顺序



文档类型定义DTD (document type definitions)

```
<!DOCTYPE 根元素名 [  
    <!ELEMENT 元素名 元素内容描述 >  
    <!ELEMENT 元素名 元素内容描述 >  
    .....  
>
```

EMPTY

ANY—— (#PCDATA)

(子元素名|子元素名|...)

(#PCDATA|子元素名|...)

```
<!DOCTYPE 书类 [  
  <!ELEMENT 书 (#PCDATA, 标题, 作者) >  
  <!ELEMENT 标题 (#PCDATA) >  
  <!ELEMENT 作者 (#PCDATA) >  
  <!ELEMENT 书类 (书 +) >  

```

```
<书类>
```

```
  <书 国际标准书号="0345374827">
```

```
    <标题> 数据结构 </标题>
```

```
    <作者> 许卓群等 </作者>
```

```
  </书>
```

```
  <书 国际标准书号="0345374828">
```

```
    <标题> 数据挖掘 </标题>
```

```
    <作者> 韩家炜等 </作者>
```

```
  </书>
```

```
</书类>
```

举例

文档类型定义DTD的缺陷

- 非XML的语法规则
- 数据类型主要是字符串
- 不支持命名空间
- 未规定元素的次数和顺序

XML模式 (P46)

- 一致性 使用XML语法 ■
- 扩展性 支持数据类型 ■
支持命名空间 ■
- 互换性 不同模式可转换
- 规范性 约束标记的使用 ■

<ElementType

content = “{ empty | textOnly | eltOnly | mixed }”

dt: type = “datatype”

model = “{ open | closed }”

name = “idref”

order = “{ one | seq | many }”

</ElementType>

<AttributeType

default = “default-value”

dt: type = “datatype”

dt: values = “enumerated-values”

name = “idref”

required = “{ yes | no }”

</AttributeType>



XML模式主要构成

XML文档

XML元素集

↑ 并列/嵌套

XML元素

<标记 属性="值"> 内容 </标记>

XML模式的主要构成

```

<ElementType
  content = “ ”
  dt:type = “ ”
  model = “ ”
  name = “ ”
  order = “ ” />

```

```

<AttributeType
  default = “ ”
  dt:type = “ ”
  dt:values = “ ”
  name = “ ”
  required = “ ” />

```

兼容DTD

基本数据类型

entity
 entities
 enumeration
 id
 idref
 idrefs
 nmtoken
 nmtokens
 notation
 string

扩展数据类型

boolean
 char
 date
 Date time
 Float
 Int
 Number
 Time
 uri
 uuid
 ...

XML的优势：可扩展的标记/自定义的标记

引发的问题

标记的冲突

举例

张三	定义	<联系方式>	信址	</联系方式>
李四	定义	<联系方式>	电话	</联系方式>

张三 定义 <联系方式> 信址 </联系方式>

李四 定义 <联系方式> 电话 </联系方式>

↓ 解决: 增加前缀

<张三: 联系方式> 信址 </张三: 联系方式>

<李四: 联系方式> 电话 </李四: 联系方式>

前缀不惟一

↓ 解决: 前缀绑定URI

张三 = <http://www.emp.org/dltu/zs.dtd>

李四 = <http://www.emp.org/dlmu/ls.xml>

命名空间前缀

命名空间名称 惟一性

张三 = `http://www.emp.org/dltu/zs.dtd`

李四 = `http://www.emp.org/dlmu/ls.xml`

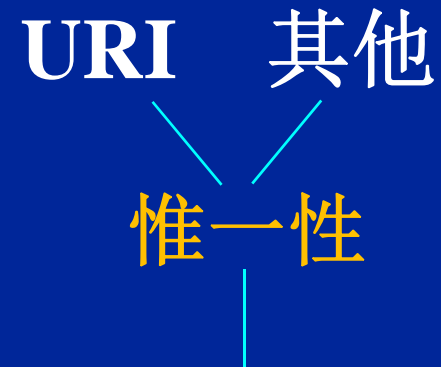
命名空间：标记的集合。具有惟一性。

Name Space

标记集合



命名空间的定义



xmlns: <命名空间前缀> = <命名空间名称>

xmlns = <命名空间名称>

合法标记

命名空间前缀: 标记

举例：XML元素的定义与约束 (P50)

```
< element  
  name = "name"  
  type = "string"  
  minOccurs = "1"  
  maxOccurs = "1"  
/>
```

```
< element  
  name = "birthday"  
  type = "date"  
  minOccurs = "1"  
  maxOccurs = "1"  
/>
```

```
< simpleType name="GendaType" >  
  < restriction base = "string" >  
    < enumeration value = "男" / >  
    < enumeration value = "女" / >  
  < /restriction >  
< /simpleType >
```

P50

```
< element  
  name = "genda"  
  type = "GendaType"  
  minOccurs = "1"  
  maxOccurs = "1"  
 / >
```


XML模式文档的结构

<Schema name = “...” xmlns = “...”>

(ElementType)

(AttributeType)

</Schema>

<ElementType

content = “{ empty | textOnly | eltOnly | mixed }”

dt: type = “ datatype ”

model = “{ open | closed }”

name = “ idref ”

order = “{ one | seq | many }” >

</ElementType>

<AttributeType

default = “ default-value ”

dt: type = “ datatype ”

dt: values = “ enumerated-values ”

name = “ idref ”

required = “{ yes | no }” >

</AttributeType>

XML模式主要构成

XML模式文档 p43.xsd

↓ 定义结构

```
<?xml version = "1.0" encoding = "gb2312" ?>
<?xml-stylesheet type = "text/xsl" href = "p43.xsl"?>
<食品>
  <食品 脂肪含量 = "低">
    <名称> 月饼 </名称>
    <描述> 中秋节食用的传统食品 </描述>
    <价格> 2.99 </价格>
  </食品>
</食品>
```

p43.xml

```
<?xml version = "1.0" encoding = "gb2312"?>
<xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema">
    <xs:element name = "食品" type = "foodType"/>
    <xs:complexType name = "foodType">
        <xs:all>
            <xs:element name="食品" type="spType"/>
        </xs:all>
    </xs:complexType>
    <xs:complexType name = "spType">
        ...
    </xs:complexType>
</xs:schema>
```



p43.xsd

```
<xs:complexType name = "spType">
  <xs:sequence>
    <xs:element name = "名称" type = "xs:string"/>
    <xs:element name = "描述" type = "xs:string"/>
    <xs:element name = "价格" type = "xs:decimal"/>
  </xs:sequence>
  <xs:attribute name = "脂肪含量">
    <xs:simpleType>
      <xs:restriction base = "xs:string">
        <xs:enumeration value = "低"/>
        <xs:enumeration value = "高"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:complexType>
```

食品的类型

XML文档与XML Schema文档的关联

XML文档的根元素上添加属性

- XML文档未用命名空间
noNamespaceSchemaLocation
= “模式文档 . xsd”
- XML文档已用命名空间
schemaLocation
= “命名空间URI 模式文档 . xsd”

XML模式文档 p43.xsd

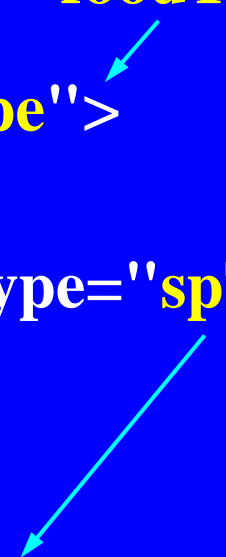


↓ 定义结构

```
<?xml version = "1.0" encoding = "gb2312" ?>
<?xml-stylesheet type = "text/xsl" href = "p43.xsl"?>
<食品 xmlns: xsi =
    "http://www.w3.org/2001/XMLSchema-instance"
    xsi: noNamespaceSchemaLocation = "p43.xsd">
    <食品 脂肪含量 = "低">
        <名称> 月饼 </名称>
        <描述> 中秋节食用的传统食品 </描述>
        <价格> 2.99 </价格>
    </食品>
</食品>
```

p43.xml

```
<?xml version = "1.0" encoding = "gb2312"?>
<xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema">
    <xs:element name = "食品" type = "foodType"/>
    <xs:complexType name = "foodType">
        <xs:all>
            <xs:element name="食品" type="spType"/>
        </xs:all>
    </xs:complexType>
    <xs:complexType name = "spType">
        ...
    </xs:complexType>
</xs:schema>
```



p43.xsd


```
<xs:complexType name = "spType">
  <xs:sequence>
    <xs:element name = "名称" type = "xs:string"/>
    <xs:element name = "描述" type = "xs:string"/>
    <xs:element name = "价格" type = "xs:decimal"/>
  </xs:sequence>
  <xs:attribute name = "脂肪含量">
    <xs:simpleType>
      <xs:restriction base = "xs:string">
        <xs:enumeration value = "低"/>
        <xs:enumeration value = "高"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:complexType>
```

食品的类型

XML文档的验证

格式良好验证 符合XML语法

使用IE、XML Spy 等

有效性验证 符合XML模式

使用XML Spy 等

课间休息



再见

