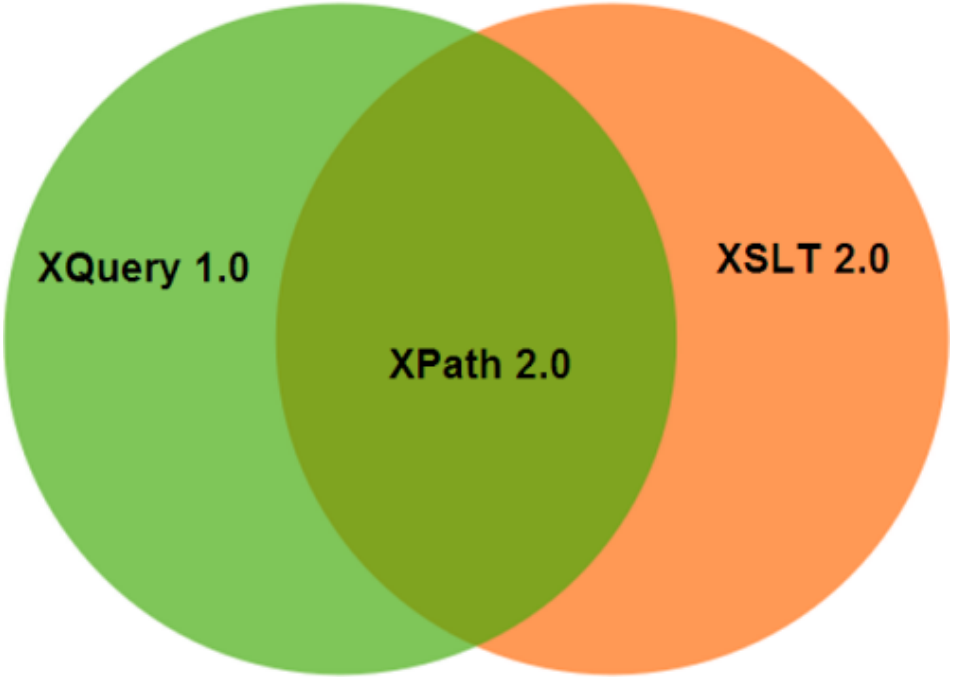
XML 原理与应用

Summer X

RUC

CH6 XML 路径语言 XPath



本章学习目标 I

- 了解 XPath 的工作原理
- 掌握 XPath 的定位路径表达式
- 熟悉 XPath 的常用函数和数据类型
- 掌握 XPath2.0 的部分新特性

目录 I

目录 II

目录 III

6.1 概述

- 考虑以下问题
  - 从存储图书信息的 XML 文档中选择指定 ISBN 号码的图书标题信息
  - 从 XML 文档中排除掉某些信息再进行处理
    - 例如：当显示员工信息时，把工资信息隐藏掉不予显示
  - 如何从 XML 文档中选择部分结点或结点集合，实现对 XML 文档中的元素、属性等信息的定位和查找，准确快速的找到 XML 文档结构树中的任意一个或一组结点

6.1.1 XPath 及其作用

- XPath
  - XML Path Language
  - 把整个 XML 文档看成是一棵由结点组成的层次树，通过结点路径进行定位
  - XPath 1.0
    - 1999 年 11 月 16 发布，作为 XSLT 和 XPointer 的配套标准，用于 XML 文档的寻址
  - XPath 2.0
    - XPath 1.0 的超集，增加了丰富的数据类型

6.1.1 XPath 及其作用

- XPath 的作用
  - XPath 为 XML 文档提供一种快捷方便并易于使用的寻址功能，实现对 XML 文档树中指定结点或结点集合的选择定位。
  - XPath 为 XML 其他相关技术提供核心支持，包括 XSLT、XQuery、XPointer、XForm 等。
  - XPath 为人们处理 XML 文档提供了一种标准通用规范，XPath 的公共 API 接口独立于特定语言，使得对 XML 文档的操纵处理更为方便

6.1.2 XPath 的工作原理

- 从内容构成来看，XML 文档与普通的文本文件相同，因此又称为序列化 XML 文档（Serialized XML Document）
- XML 更注重逻辑结构特征，目前三个从不同角度描述 XML 文档的逻辑模型
  - XPath
  - DOM
  - XML 信息集合（XML Information Set）

XPath 数据模型

- XPath 把 XML 文档看成是一个或一组文档树结点，文档的大部分内容都表示为结点
- XML 文档的特殊部分没有对应的 XPath 表示方式
  - 例如：最开始的 XML 声明语句

文档对象模型 DOM

- 同样把整个 XML 文档表示成一棵由结点组成的层次树
- DOM 和 XPath 在表示结点的方式上有所不同
  - DOM 中的结点包含有丰富的信息，更多的应用于 XML 编程处理方面
  - 例如 DOM 结点对象有类型、属性、方法等概念

XML 信息集合

- 也叫 XML 信息集
- XML 信息集是 XML 文档的纯信息表示，适用于从 XML 角度而非文本角度比较两个文件的异同
  - 信息集不区分空元素的两种形式
  - 属性所使用的引号类型也不重要，克服了严格的文本字符比较的缺点

文本不同但信息集合角度相同的两个 XML 文档 I

Code 1: ”6-1.xml”

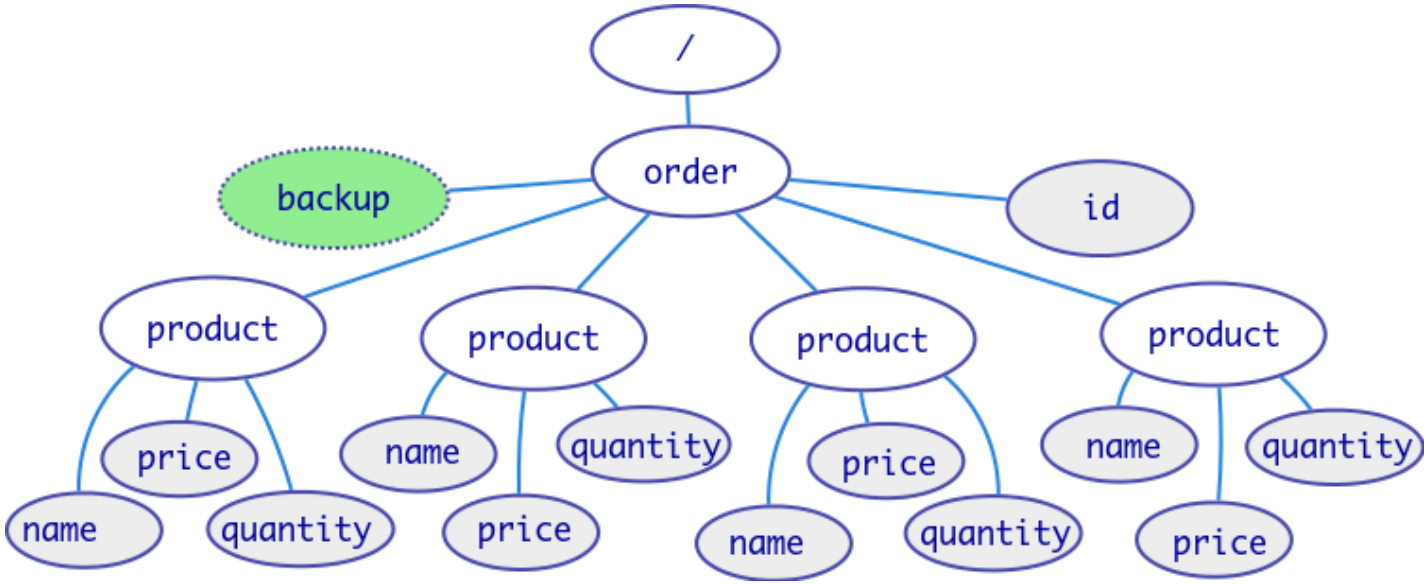
```
1 <?xml version=”1.0” encoding=”UTF−8”?>
2 <order id=”TEST−01”>
3    <?backup dbname=”order”?>
4    <product name=”打印纸” price=”30” quantity=”5”/>
5    <product name=”圆珠笔” price=”2” quantity=”10”/>
6    <product name=”铅笔” price=”1.5” quantity=”10”/>
7    <product name=”白板” price=”80” quantity=”2”/>
8 </order>
```

文本不同但信息集合角度相同的两个 XML 文档 II

Code 2: ”6-2.xml”

```
1 <order id='TEST-01' >
2   <?backup dbname="order"?>
3   <product name='打印纸' price='30' quantity='5'/>
4   <product name='圆珠笔' price='2' quantity='10'/>
5   <product name='铅笔' price='1.5' quantity='10'/>
6   <product name='白板' price='80' quantity='2'/>
7 </order>
```

XPath



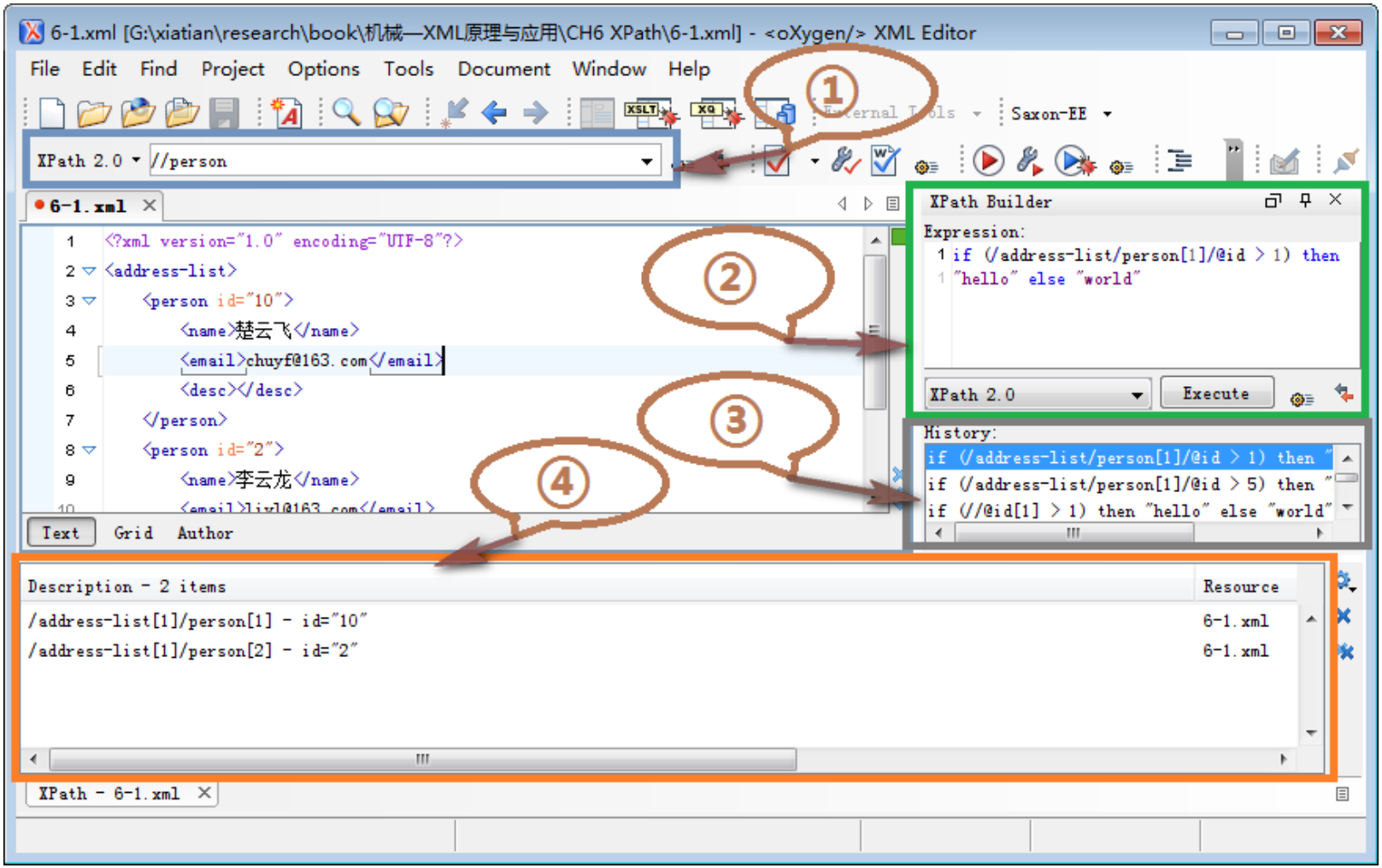
- 利用结点之间的层次关系实现定位
- 例子：

```
/order/product[2]/@name
```

6.1.3 XPath 的表达式与操作符

表达式类型	操作符
定位路径表达式	/, //, |
布尔表达式	or, and
等式表达式	=, !=
关系表达式	<=, <, >=, >
数值表达式	+, -, div, mod, *, -(unary)

6.1.4 如何测试 XPath



6.2 XPath 结点与结点集

- 结点可看成是 XPath 定位步中用来实现定位功能的离散的、逻辑性的东西
  - XML 文档中的任意一个元素、属性、处理指令、注释等，都可以是一个结点
- 结点集：结点组成的集合

6.2.1 结点的基本属性

- 结点名称
- 结点顺序
- 结点之间的家族关系

结点名称

- 大部分结点都用名称（根结点没有），三种名称：
  - 限定名称 QName（Qualified name）
    - XML 文档实例中该结点的唯一标识符，包括结点的命名空间前缀部分
    - <product> 元素的 QName 为 product
    - <common:product> 的 QName 为 common:product
  - 本地名称（Local-name）
    - QName 去除命名空间前缀后剩余的内容
    - 如果没有指定命名空间，其规范名称和本地名称完全相同
    - <common:product> 的本地名称为 product
  - 扩展名称（Expended-name）
    - 扩展名称由与命名空间关联的 URI 与本地名称共同构成
    - 扩展名称不关心命名空间前缀的具体名称

结点顺序

- 结点位置是由该结点前面的结点和后面的结点所决定的
- 通常按照先后顺序进行访问
- 借助于 XPath 坐标轴，也可以实现对 XML 文档结点的逆序访问或指定顺序访问

结点之间的家族关系

- 结点之间通过 XPath 坐标轴维持关系
- 常见关系包括：
  - 相邻关系
  - 祖先关系
  - 子孙关系
  - ...

6.2.1 结点的基本属性

- 根节点
- 元素节点
- 属性节点
- 命名空间节点
- 处理指令节点
- 注释节点
- 正文节点

6.2.3 结点集

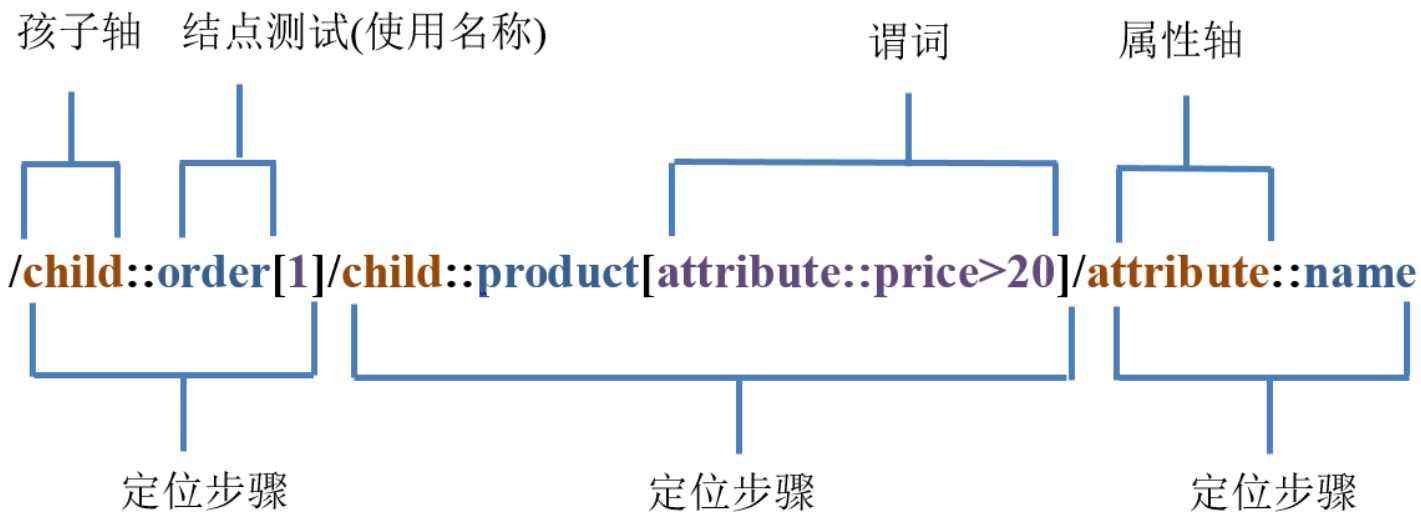
- 由结点构成的集合
- 例如“6-1.xml”中，位置路径/order/product 则返回一个包含了 4 个 product 元素的结点集

6.3 XPath 定位路径表达式

- 由一个或多个定位步骤组成，每个定位步骤之间用斜线“/”分隔
- 如表达式以“/”开始，则称为绝对路径，否则称为相对路径
- 使用“/”符号作为定位步骤之间的分隔符

6.3.1 XPath 定位步骤

- 定位步骤的组成
  - 轴：用来导航 XPath 数据模型的结点树的工具
  - 测试结点：用来确定选取轴中哪类结点
  - 谓词：可选步骤，用来过滤轴和定位方法所选取的结点集



6.3.2 XPath 轴: 在结点的哪一个方向上进行导航定位

轴	名称	方向
child	子轴	前向
parent	双亲轴	不可用
attribute	属性轴	不可用
ancestor	祖先轴	反向
ancestor-or-self	祖先自身轴	反向
descendant	子孙轴	前向
descendant-or-self	子孙自身轴	前向
following	后继轴	前向
following-sibling	后继兄弟轴	前向
preceding	前驱轴	反向
preceding-sibling	前驱兄弟轴	反向
namespace	命名空间轴	不可用
self	自身轴	不可用

XPath 轴示例 I

- 示例文档

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <abc:order id="TEST-01" xmlns:abc="http://www.abc.com"
3             xmlns:abc2="http://www.abc2.com">
4   <?backup dbname="order"?>
5   <product name="打印纸" price="30" quantity="5"/>
6   <product name="圆珠笔" price="2" quantity="10"/>
7   <product name="铅笔" price="1.5" quantity="10"/>
8   <product name="白板" price="80" quantity="2"/>
9 </abc:order>
```

XPath 轴示例 II

- 测试表达式：

```
/abc:order/namespace::abc
```

- 结果：

```
abc - http://www.abc.com
```

- 测试表达式：

```
/abc:order/namespace::node()
```

- 结果：

```
xml - http://www.w3.org/XML/1998/namespace
abc - http://www.abc.com
abc2 - http://www.abc2.com
```

6.3.3 结点测试

- 用于确定轴中的结点，只有给定结点的结点测试结果为 true，该结点才会保留在结点集中
- 使用名称进行测试，例如：
  - child::brandA:product
  - 选中 QName 为 brandA:product 的所有孩子元素结点
- 使用类型进行测试，例如：
  - child::text( )
  - 确定文本子结点
- 参考教材内容

6.3.4 谓词

- 谓词是指针对条件表达式进行求值返回真或假的过程
- 谓词置于定位步骤末端的方括号中，用于筛选一个结点集以生成新的结点集
- 针对结点集中每一个被筛选的结点，谓词表达式将此结点作为上下文结点进行求值，如为 true，则保留在结点集中，否则，从结点集中删除
- 例子：
  - descendant::product[attribute::price > 20]
  - 选择 product 的子孙元素，且 product 元素拥有大于 20 的 price 属性

6.3.5 定位路径缩写

示例	对应缩写示例
child::product	product
child::product/attribute::name	product/@name
self::node()/product	./product
parent::node()/@name	../@name
/descendant-or-self::node()/product	//product
child::product[position( ) == 3]	product[3]

6.4 XPath 基本表达式

- 布尔表达式：and or
  - person[name and age]
  - person[(age>30) or (age<50)]
- 等式表达式：=, !=
  - age = 30
  - age != 30
- 关系表达式：<, <=, >, >=
  - age <= 30
- 数值表达式：+, -, div, mod, * -(unary)
  - person[(age mod 2) =0]
  - 选择具有偶数数值 age 子元素的 person 子元素

6.5 XPath 数据类型

- 字符串类型
  - person[name=' 李云龙']
- 数值类型
  - <pages>102</pages>
  - pages + 10
- 布尔类型
  - order[@id="1"]
- 结点集类型、
  - //book

6.6 XPath 1.0 常用函数

- 结点集相关函数
- 布尔函数
- 数值函数
- 字符串函数

6.6.1 结点集函数

- last()
  - 返回上下文的大小，即给定上下文中的节点数。
- position()
  - 返回上下文结点的位置。比如，可以用表达式 position()=last() 测试处理的是否是集合中最后一个节点。
- count()
  - 参数为结点集，返回这个结点集中包含的结点个数。
- id()
  - 参数为字符串，返回一个结点集，结点集中的每一个结点的 ID 属性值等于这个参数字符串。XPath 中的另外三个与结点集相关的函数为：

6.6.2 布尔函数

- boolean()
  - 以对象为参数，返回一个布尔值。当参数是一个非 0 数值时，返回 true，为数值 0 时，返回 false；当参数为字符串时，返回 true；如参数为结点集并且非空，返回 true，否则为 false。
- not()

6.6.3 数值函数

- number ()
  - 把可选的对象参数转化成数字
- sum ()
  - 以结点集为参数，把每个结点值转换为数值，再求和返回
- ceiling ()
  - 参数为一个数值，返回比该数值大的最小整数
- floor ()
  - 参数为一个数值，返回比该数值小的最大整数
- round ()
  - 参数为一个数值，返回和参数最接近的整数

6.6.4 字符串函数

- string ()
  - 参数可以是任何类型，返回它们的字符串值。
- string-length ()
  - 参数为一个字符串，返回该字符串长度。
- substring ()
- concat ()
- contains ()
- starts-with ()

6.7 XPath 2.0 I

- 2007 年 1 月 23 日正式成为推荐标准
- 新特性
  - 支持 XML Schema 的数据类型
  - 提供了更为丰富的处理函数
    - tokenize()：拆分字符串，
    - matches()：测试字符串是否匹配
    - current-date()
  - 支持序列
    - 例子：

```
for $i in 1 to 6 return $i*$i
```

    - 该表达式会生成序列（1,4,9,16,25,36）

- 支持逻辑判断

```
if( sum(for $p in //product return $p/@price*$p/@quantity)  > 200) then
    "YES" else "NO"
```

  - 针对例 6-5 中的 XML 文档，输出结果为"YES"
- 提供了更多的结点测试函数
  - //element()：选取文档中的全部元素结点
  - //attribute()：选择所有的属性结点
- 可以调用自定义函数
  - XPath2.0 本身并不能建立自定义函数，只能在支持 XPath 的 XSLT、XQuery 等应用程序中建立自定义函数，供 XPath 调用

END