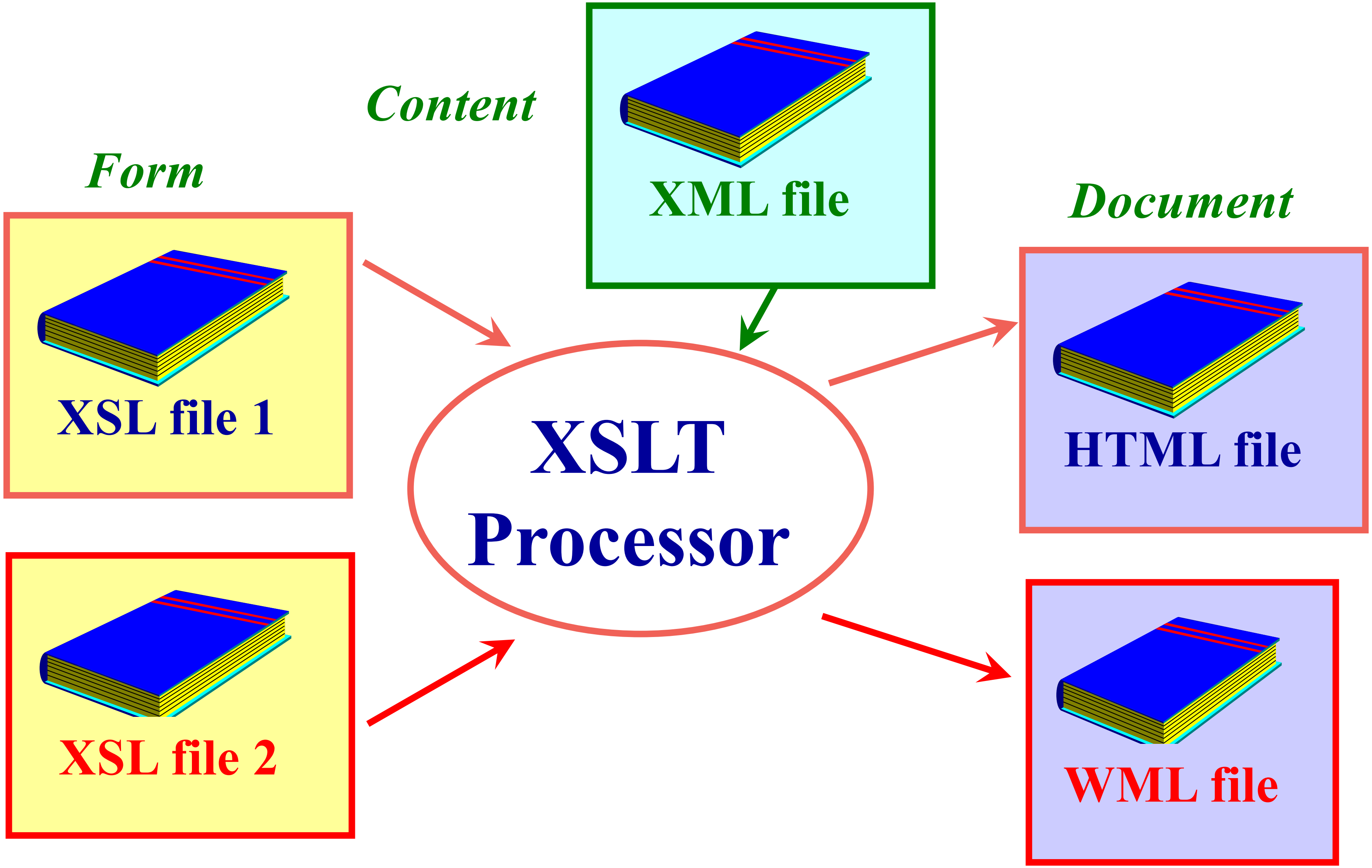
Hands on XSL

XSL-basic elements

Transforming XML



See also https://www.w3schools.com/xml/xsl_intro.asp

HANDS ON! - Esempio1 XML

```xml
<?xml version="1.0"?>
<?xml-stylesheet href="hello.xsl" type="text/xsl"?>

<!-- Here is a sample XML file -->
<page>
    <title>Test Page</title>
    <content>
        <paragraph>What you see is what you get!</paragraph>
    </content>
</page>
```

HANDS ON! - Esempio1 XSL a

```
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="page">
    <html>
      <head>
        <title>
          <xsl:value-of select="title"/>
        </title>
      </head>
      <body bgcolor="#ffffff">
        <xsl:apply-templates/>
      </body>
    </html>
  </xsl:template>
```

HANDS ON! - Esempio1 XSL b

```
  <xsl:template match="paragraph">
    <p align="center">
      <i>
        <xsl:apply-templates/>
      </i>
    </p>
  </xsl:template>
</xsl:stylesheet>
```

HANDS ON! - Esempio1 Xalan

Let us use the Apache XSLT processor: Xalan.

1) Get Xalan from xml.apache.org/xalan-j/index.html

```
2 )Set CLASSPATH=%CLASSPATH%;.../xalan.jar;
          .../xerces.jar
```

```
3) java org.apache.xalan.xslt.Process
–IN testPage.xml –XSL testPage.xsl –O out.html
```

(see https://xml.apache.org/xalan-j/commandline.html)

HANDS ON! - Esempio1 Output HTML

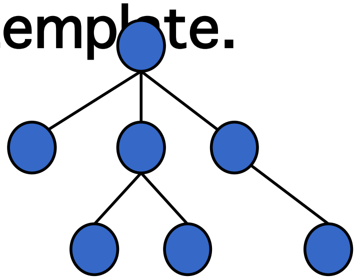
```
<html>
    <head>
        <title>
            Test Page
        </title>
    </head>
    <body bgcolor="#ffffff">
        <p align="center">
            <i>
                What you see is what you get!
            </i>
        </p>
    </body>
</html>
```

The process

- The process starts by traversing the document tree, attempting to find a single matching rule for each visited node.

- Once the rule is found, the body of the rule is istantiated

- Further processing is specified with the *<xsl:apply-templates>*. The nodes to process are specified in the *match* attribute. If the attribute is omitted, it continues with the next element that has a matching template.



Implicit rules

```
<template match=“/|*”>
  <apply-templates/>
</template>

<template match=“text()”>
  <value-of select=“.”/>
</template>
```

Selective processing - example

```
<?xml version="1.0"?>
<?xml-stylesheet href="IgnoraParte4.xsl" type="text/xsl" ?>
<ROOT>
<SECRET>
SEZIONE RISERVATA:
   <TAG1>Testo Privato</TAG1>
</SECRET>
<PUBLIC>
SEZIONE PUBBLICA
  <TAG1>Testo Pubblico</TAG1>
</PUBLIC>
</ROOT>
```

Selective processing - example

```
<xsl:stylesheet
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0"
>
  <xsl:template match="SECRET">A private part
   exists</xsl:template>
  <xsl:template match="PUBLIC">A public part exists/xsl:template>
  <xsl:template match="PUBLIC">The public part contains:
   <xsl:apply-templates/></xsl:template>
</xsl:stylesheet>
```

OUTPUT

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

A private part exists
The public part contains:
SEZIONE PUBBLICA
Testo Pubblico
```

Pattern Matching - nodes

/ matches the root node

A matches any <A> element
* matches any element
A|B matches any <A> or <B> element
A/B matches any <B> element within a <A> element
A//B matches any <B> element with a <A> ancestor

text() matches any text node

Pattern Matching

id(“pippo”) matches the element with unique ID pippo

A[1] matches any <A> element that is the first <A> child of its parent

A[last()=1] matches any <A> element that is the last <A> child of its parent

B/A[position() mod 2 = 1] matches any <A> element that is an odd-numbered <A> child of its B parent

Pattern Matching - attributes

@A matches any A attribute

@* matches any attribute

B[@A="v"]//C matches any <C> element that has a <B> ancestor with a A attribute with v value

processing-instruction()
node()

Imports, priorities and spaces

IMPORT

`<import  href=“…”>`

PRIORITIES

`<template  match=“…” priority=“2” >` (default 1)

When priorities are equal, the last definition wins

STRIPPING SPACES

```
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">
<xsl:strip-space elements="*"/>
...
</xsl:stylesheet>
```

Variables, templates and parameters

```
<variable name=“colore”>rosso</variable>
...
Il colore e’ : <xsl:value-of select=“$colore”>
```

Once a value has been assigned to a variable, it cannot be changed

```
<template name=“header”>
Sequence of text and tags
</template>
...
<call-template name=“header”/>
```

```
<template name=“header”><param name=“P”>default</param>
Sequence of text and tags, including <value-of select=“$P”/>
</template>
...
<call-template name=“header”>
<with-param name=“P”>3</with-param></call-template>
```

conditions

```
< xsl: if test”position() mod 2 =0”>
    <B><apply_templates/></B>
</ xsl: if>
```

```
<xsl:choose>
<xsl: when test”position() mod 2 =0”>
    <B><apply_templates/></B>
</xsl: when>
<xsl: otherwise>
    <I><apply_templates/></I>
</xsl: otherwise >
</xsl: choose >
```

for-each

```
<xsl:for-each select=“expression”>
    some rule
</xsl:for-each>
```

Sorting

```
<list>
    <item sortcode=“C”> Pluto</item>
    <item sortcode=“A”> Topolino </item>
    <item sortcode=“B”>Pippo</item>
</list>
```

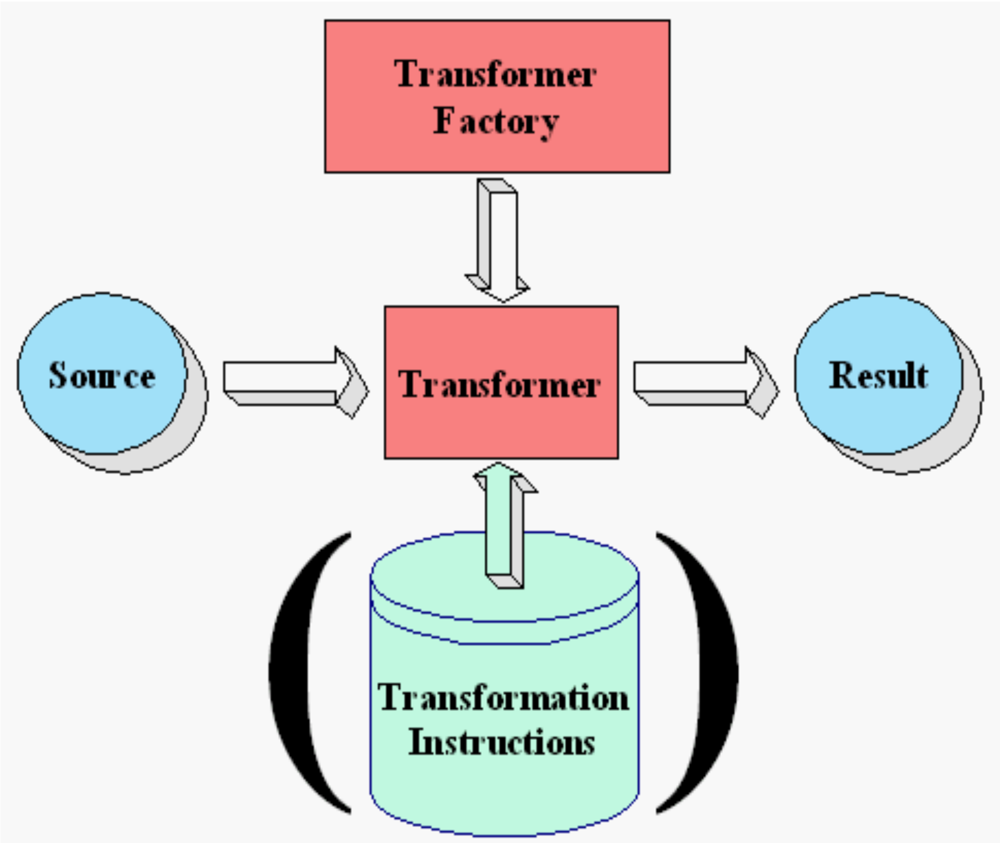
```
<template match=“list”>
 <apply-templates><sort/></apply-templates>
</template>
```

```
<template match=“list”>
 <apply-templates>
 <sort select=“@sortcode” order=descending/>
</apply-templates>
</template>
```

Transformations

- ***Using XSLT from Java***

TrAX



```
TransformerFactory tf = TransformerFactory .newInstance();
 StreamSource xslSS=new StreamSource(“source.xsl”);
 StreamSource xmlSS=new StreamSource(“source.xml”);
 Transformer t=tf.newTrasformer(xslSS);
 t.transform(xmlSS,new StreamResult(new
          FileOutputStream(“out.html”);
```

```
java –Djavax.xml.transform.TransformerFactory=
org.apache.xalan.processor.TrasformerFactoryImpl MyClass
```

xml.transform packages

Package	***Description***
javax.xml.transform	Defines the TransformerFactory and Transformer classes, which you use to get a object capable of doing transformations. After creating a transformer object, you invoke its transform() method, providing it with an input (source) and output (result).
javax.xml.transform.dom	Classes to create input (source) and output (result) objects from a DOM.
javax.xml.transform.sax	Classes to create input (source) from a SAX parser and output (result) objects from a SAX event handler.
javax.xml.transform.stream	Classes to create input (source) and output (result) objects from an I/O stream.

TrAX main classes

- javax.xml.transform.Transformer
- transform(Source xmls, Result output)

- javax.xml.transform.sax.SAXResult implements Result
- javax.xml.transform.sax.SAXSource implements Source

- javax.xml.transform.stream.StreamResult implements Result
- javax.xml.transform.stream.StreamSource implements Source

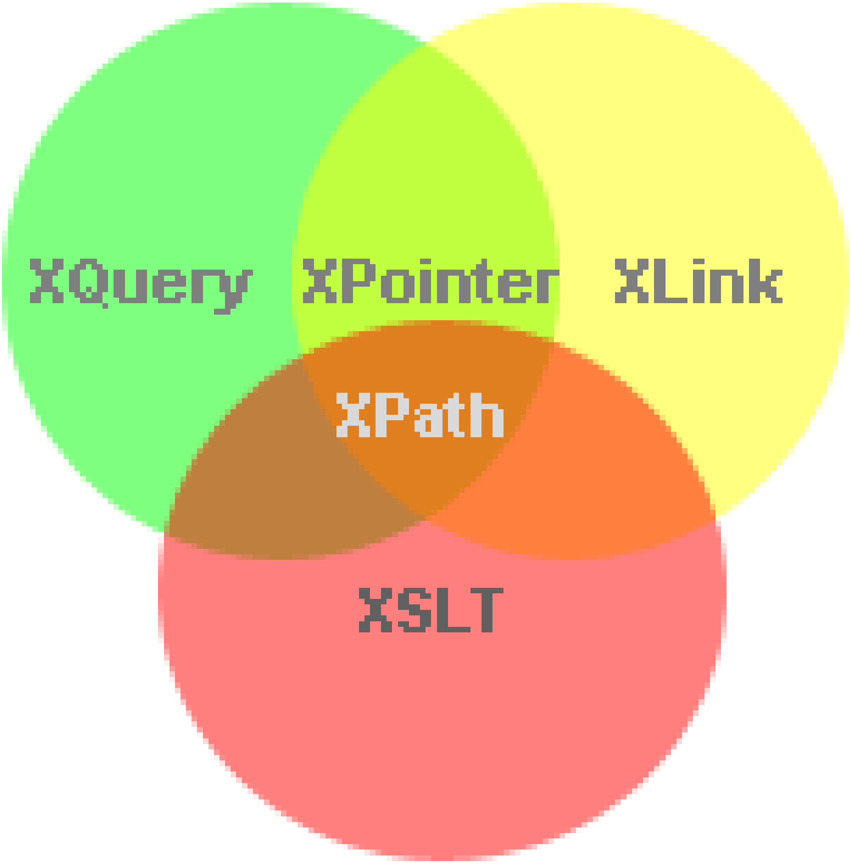
- javax.xml.transform.dom.DOMResult implements Result
- javax.xml.transform. dom.DOMSource implements Source

Xpath

See also

https://www.w3schools.com/xml/xpath_intro.asp

Overlapping domains



XPath

- XPath is a syntax for defining parts of an XML document
- XPath uses path expressions to navigate in XML documents
- XPath contains a library of standard functions
- XPath is a major element in XSLT
- XPath is a W3C Standard

Terminology

- Element
- Attribute
- text,
- namespace,
- processing-instruction,
- comment,
- document (root) nodes

expressions

The most useful path expressions:

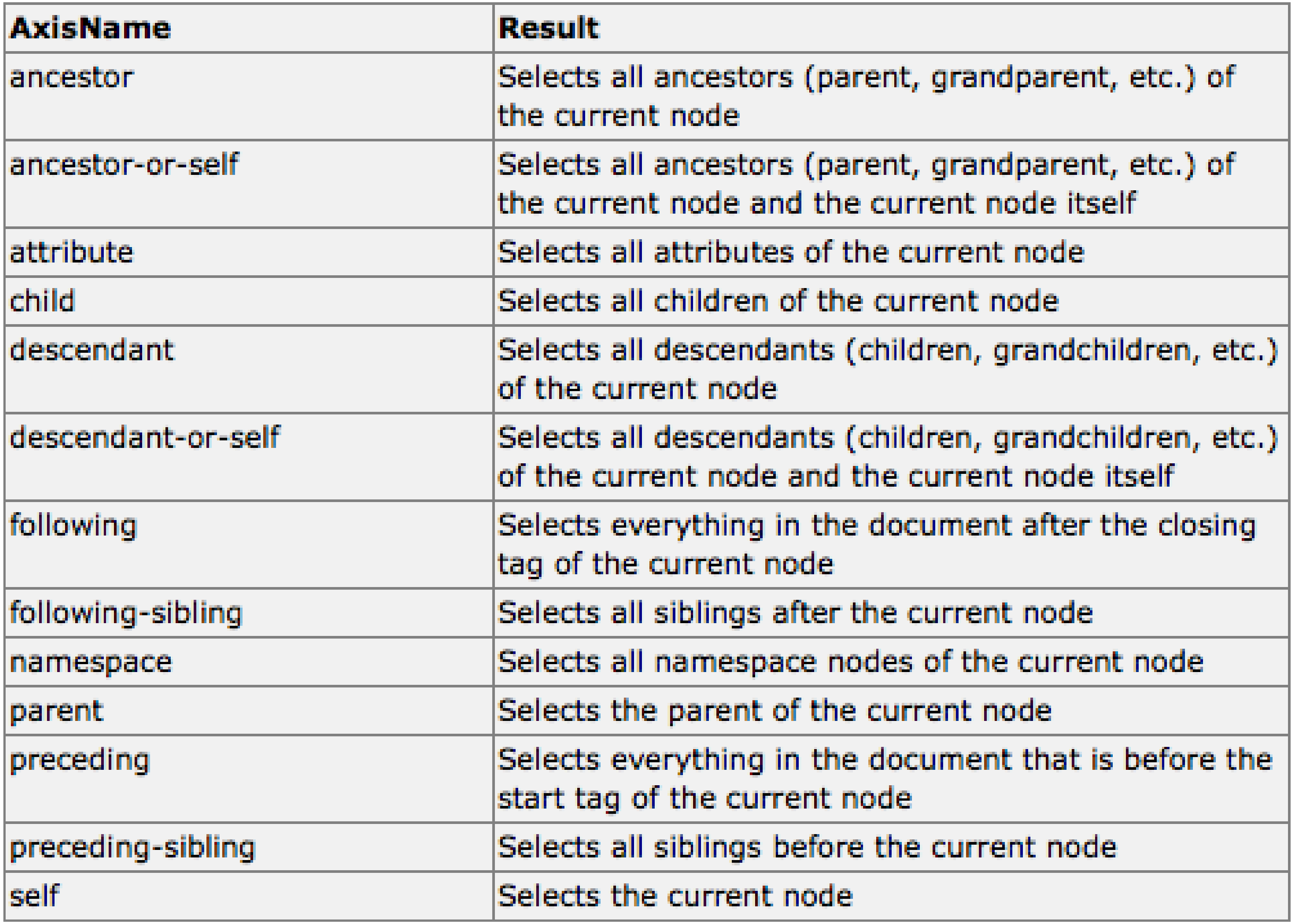
- nodename Selects all child nodes of the named node
- / Selects from the root node
- // Selects nodes in the document from the current node that match the selection no matter where they are
- . Selects the current node
- .. Selects the parent of the current node
- @ Selects attributes

Wildcards

Path wildcards can be used to select unknown XML elements.

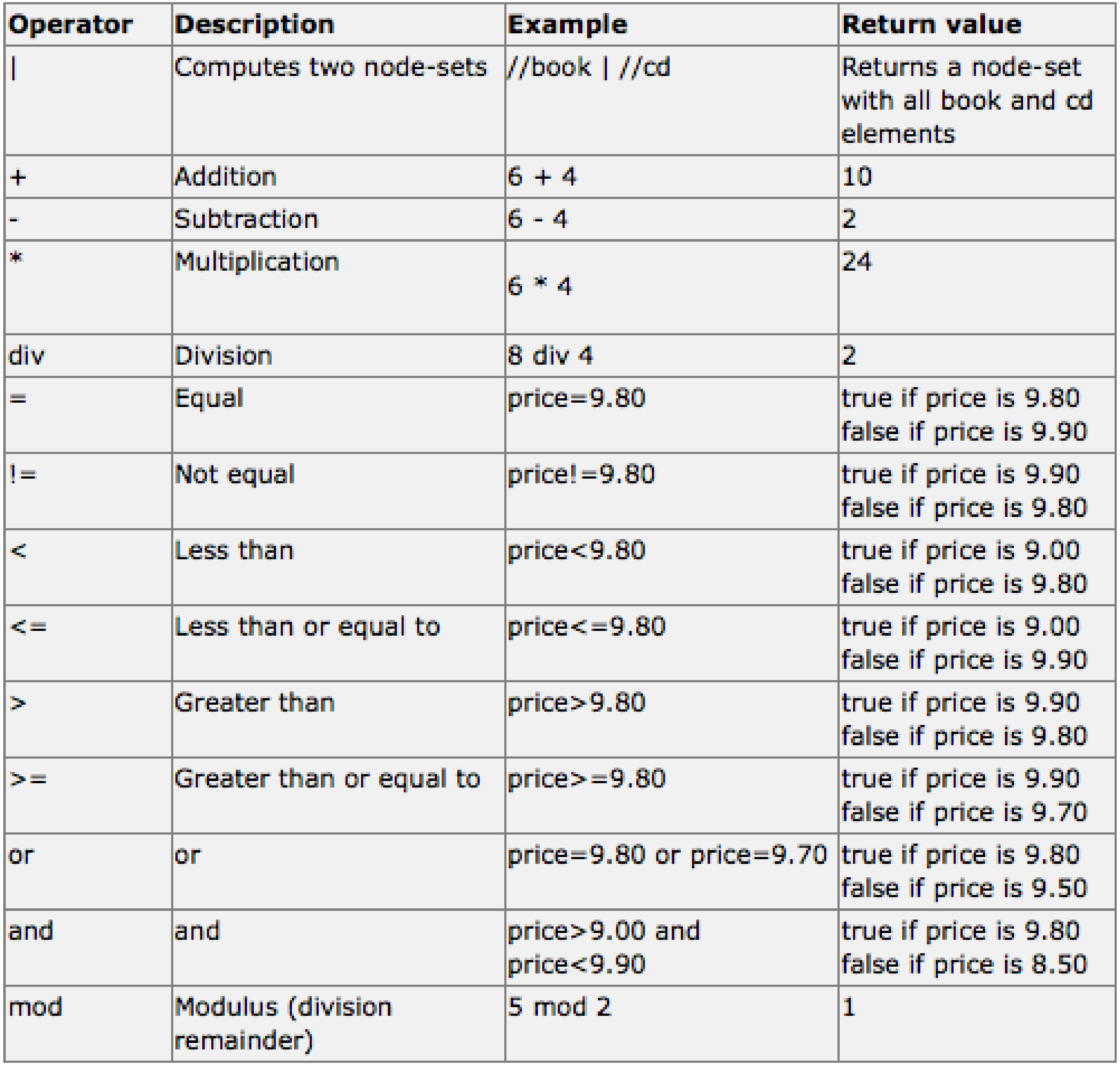
*	Matches any element node
@*	Matches any attribute node
node()	Matches any node of any kind

Axis: a node-set relative to the current node.



AxisName	Result
ancestor	Selects all ancestors (parent, grandparent, etc.) of the current node
ancestor-or-self	Selects all ancestors (parent, grandparent, etc.) of the current node and the current node itself
attribute	Selects all attributes of the current node
child	Selects all children of the current node
descendant	Selects all descendants (children, grandchildren, etc.) of the current node
descendant-or-self	Selects all descendants (children, grandchildren, etc.) of the current node and the current node itself
following	Selects everything in the document after the closing tag of the current node
following-sibling	Selects all siblings after the current node
namespace	Selects all namespace nodes of the current node
parent	Selects the parent of the current node
preceding	Selects everything in the document that is before the start tag of the current node
preceding-sibling	Selects all siblings before the current node
self	Selects the current node

Operators



Operator	Description	Example	Return value
|	Computes two node-sets	//book | //cd	Returns a node-set with all book and cd elements
+	Addition	6 + 4	10
-	Subtraction	6 - 4	2
*	Multiplication	6 * 4	24
div	Division	8 div 4	2
=	Equal	price=9.80	true if price is 9.80 false if price is 9.90
!=	Not equal	price!=9.80	true if price is 9.90 false if price is 9.80
<	Less than	price<9.80	true if price is 9.00 false if price is 9.80
<=	Less than or equal to	price<=9.80	true if price is 9.00 false if price is 9.90
>	Greater than	price>9.80	true if price is 9.90 false if price is 9.80
>=	Greater than or equal to	price>=9.80	true if price is 9.90 false if price is 9.70
or	or	price=9.80 or price=9.70	true if price is 9.80 false if price is 9.50
and	and	price>9.00 and price<9.90	true if price is 9.80 false if price is 8.50
mod	Modulus (division remainder)	5 mod 2	1

Xpath functions

- See https://www.w3schools.com/xml/xsl_functions.asp

Pattern Matching - nodes

/ matches the root node

A matches any <A> element

* matches any element

A|B matches any <A> or <B> element

A/B matches any <B> element within a <A> element

A//B matches any <B> element with a <A> ancestor

text() matches any text node

Pattern Matching

id("pippo") matches the element with unique ID pippo

A[1] matches any <A> element that is the first <A> child of its parent

A[last()=1] matches any <A> element that is the last <A> child of its parent

B/A[position() mod 2 = 1] matches any <A> element that is an odd-numbered <A> child of its B parent

Pattern Matching - attributes

@A matches any A attribute

@* matches any attribute

B[@A=“v”]//C matches any <C> element that has a <B> ancestor with a A attribute with v value

processing-instruction()
node()

Using Xpath from java

XPath expressions are much easier to write than detailed (DOM) navigation code.

When you need to extract information from an XML document, the quickest and simplest way is to embed an XPath expression inside your Java program.

Java 5 introduces the javax.xml.xpath package, an XML object-model independent library for querying documents with XPath.

Example

Find all the books by Dante Alighieri

- //book[author="Dante Alighieri"]/title

assuming a suitable data structure:

```
...
    <book author="someone">
     ...
    <title>Title of the book</title>
    ...
    </book>
...
```

Java code

```
import java.io.IOException;
import org.w3c.dom.*;
import org.xml.sax.SAXException;
import javax.xml.parsers.*;
import javax.xml.xpath.*;
public class XPathExample {
 public static void main(String[] args)
  throws ParserConfigurationException, SAXException,
      IOException, XPathExpressionException {
   //read an XML file into a DOM Document
   DocumentBuilderFactory domFactory=
        DocumentBuilderFactory.newInstance();
   domFactory.setNamespaceAware(true); // never forget this!
   DocumentBuilder builder =
   domFactory.newDocumentBuilder();Document doc =
        builder.parse("books.xml");
```

Java code

```
// prepare the XPath expression
   XPathFactory factory = XPathFactory.newInstance();
   XPath xpath = factory.newXPath();
   XPathExpression expr
    = xpath.compile("//book[author='Dante Alighieri']/title/text()");
// evaluate the expression on a Node
   Object result = expr.evaluate(doc, XPathConstants.NODESET);
// examine the results
   NodeList nodes = (NodeList) result;
   for (int i = 0; i < nodes.getLength(); i++) {
    System.out.println(nodes.item(i).getNodeValue());
   }
  }
```