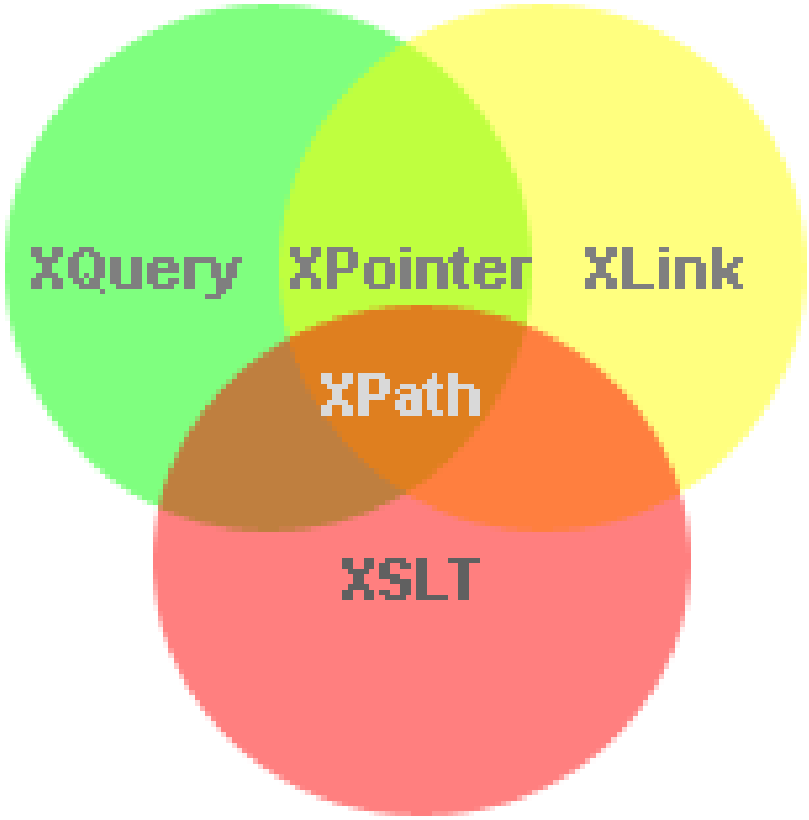
Xpath

Sources:

http://www.brics.dk/~amoeller/XML

http://www.w3schools.com

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

Opera

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

http://www.w3schools.com/xpath/xpath_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());
    }
  }
```

XML Serialization

Using XML to serialize Java Classes

```
import java.io.File;
import org.simpleframework.xml.Serializer;
import org.simpleframework.xml.load.Persister;
public class StorableAsXML implements Serializable {
    // ==== SERIALIZATION/DESERIALIZATION PRIMITIVES
    public void persist(File f){
        Serializer serializer = new Persister();
        try {
            serializer.write(this, f);
        } catch (Exception ex) {
            ex.printStackTrace();
        }
```

```
import java.io.Serializable;
public class X implements Serializable
 FileOutputStream fos=null;
        ObjectOutputStream oos=null;
        try {
            fos=new FileOutputStream(f);
            oos = new ObjectOutputStream(fos);
            oos.writeObject(this);
        } catch (IOException ex) {
              ex.printStackTrace();
        }
```

Using XML to serialize Java Classes

```
public StorableAsXML resume(File f, Class<? extends
 StorableAsXML> c){
    StorableAsXML retval = null;
    try {
        Serializer serializer = new Persister();
        retval = (StorableAsXML)serializer.read(c, f);
    } catch (Exception ex) {
        ex.printStackTrace();
    }
    return retval;
}
}
```

```
FileInputStream fis=null;
      ObjectInputStream ois=null;
      try {
          fis=new FileInputStream(f);
          ois = new ObjectInputStream(fis);
          retval=(X)ois.readObject();
      } catch (Exception ex) {
          ex.printStackTrace();
      }
```

Using XML to serialize Java Classes

```
public class Lecture  extends StorableAsXML implements Serializable {
    private Set<String> lecturers=null; //non serialized field
    @Element(name="NAME")
    public String lectureName=null;
    @Element(name="DATE")
    private Date date=null;
    @Element(name="SEQUENCE_NUMBER")
    private int sequenceNumber=-1; //-1 means not initialized
    @Element(name="COURSE_HOME")
    private String courseRef=null; //Home per il corso
    @Element(name="LECTURE_HOME")
     private String dirName=null;
    @Element(name="LECTURER",required=false)
    private String lecturer=null;
    @Element(name="VIDEO",required=false)
    private String videoFileName=null;
    @Element(name="VIDEO_LENGTH",required=false)
    private String videoLenght=null; //null = Video does not exist
    @Element(name="HAS_POST_PROCESSING")
    private boolean hasPostProcessing=false;
```

```
public Lecture(){
// needed to be a
bean
//for XMLSerialization
        …;
    }
```

Generated XML

```
<LECTURE>
  <NAME>gg</NAME>
  <DATE>2008-09-05 16:20:34.365 CEST</DATE>
  <SEQUENCE_NUMBER>1</SEQUENCE_NUMBER>
  <COURSE_HOME>/Users/ronchet/_LODE/COURSES/Hh_2008
  </COURSE_HOME>
  <LECTURE_HOME>01_Gg_2008-09-05</LECTURE_HOME>
  <LECTURER>A.B.</LECTURER>
  <HAS_POST_PROCESSING>false</HAS_POST_PROCESSING>
</LECTURE>
```

Using XML to serialize Java Classes

```
@Root(name="COURSE")
public class Course extends StorableAsXML implements Serializable
    {
    @Element(name="NAME")
    private String courseName=null;
    @Element(name="YEAR")
    private String year=null;
    @Element(name="COURSE_HOME")
    private String fullPath=null;

    @ElementList(name="LECTURES",entry="LECTURE")
    private Set<String> lectures=new TreeSet<String>();

   @ElementList(name="TEACHERS",entry="TEACHER_NAME")
    private Set<String> teachers=new TreeSet<String>();
```

```
<COURSE>
  <NAME>hh</NAME>
  <YEAR>2008</YEAR>
  <COURSE_HOME>/Hh_2008</COURSE_HOME>
  <LECTURES class="java.util.TreeSet">
    <LECTURE>01_Gg_2008-09-05</LECTURE>
  </LECTURES>
  <TEACHERS class="java.util.TreeSet">
     <TEACHER_NAME>A.B.</TEACHER_NAME>
     <TEACHER_NAME>C.D.</TEACHER_NAME>
  </TEACHERS>
</COURSE>
```

Javadoc

http://simple.sourceforge.net/download/stream/doc/javadoc/org/simpleframework/xml/package-summary.html