Xpath

http://www.w3.org/TR/xpath

By Michael Dockery
michaeld@dpslink.com

What is XPath?

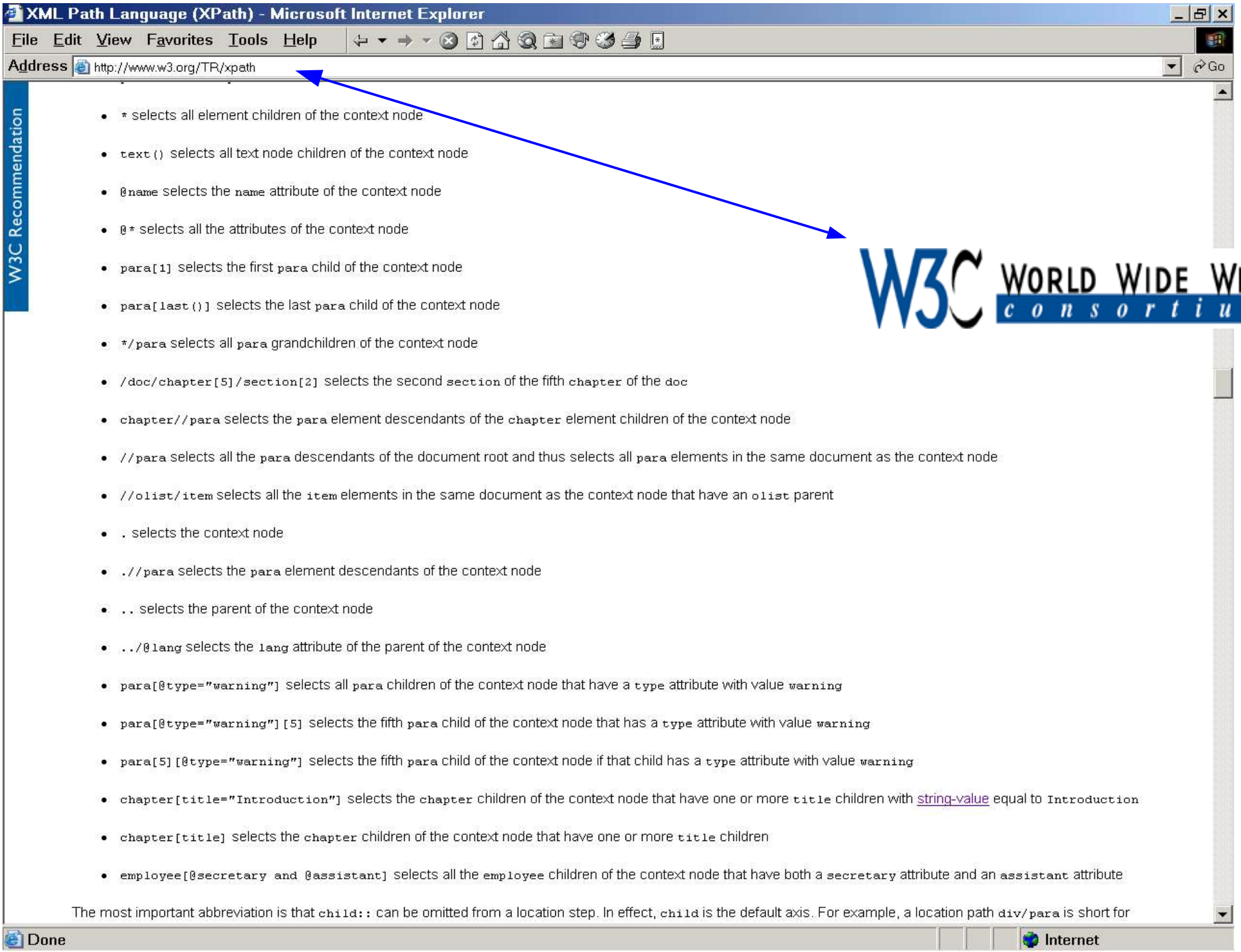
XPath is a cross-platform language for addressing parts of an XML document!

XML documents can be represented as a tree view of nodes
...very similar to tree view of PC folders.

XPath uses a pattern *expression* to identify nodes in an XML document.
An XPath pattern is a slash-separated list of child element names that describe a path through the XML document. The pattern "selects" elements that match the path.
The following XPath expression selects all the price elements of all the cd elements of the catalog element: **/catalog/cd/price**

...or for verbage from Microsoft:
XPath provides a uniform and compact syntax for addressing XML documents.
Instead of writing code to explicitly walk through a document's structure looking for nodes that match some criteria, an XPath expression can be used to encapsulate that entire process....

W3C WORLD WIDE WEB consortium



- `*` selects all element children of the context node
- `text()` selects all text node children of the context node
- `@name` selects the `name` attribute of the context node
- `@*` selects all the attributes of the context node
- `para[1]` selects the first `para` child of the context node
- `para[last()]` selects the last `para` child of the context node
- `*/para` selects all `para` grandchildren of the context node
- `/doc/chapter[5]/section[2]` selects the second `section` of the fifth `chapter` of the `doc`
- `chapter//para` selects the `para` element descendants of the `chapter` element children of the context node
- `//para` selects all the `para` descendants of the document root and thus selects all `para` elements in the same document as the context node
- `//olist/item` selects all the `item` elements in the same document as the context node that have an `olist` parent
- `.` selects the context node
- `.//para` selects the `para` element descendants of the context node
- `..` selects the parent of the context node
- `../@lang` selects the `lang` attribute of the parent of the context node
- `para[@type="warning"]` selects all `para` children of the context node that have a `type` attribute with value `warning`
- `para[@type="warning"][5]` selects the fifth `para` child of the context node that has a `type` attribute with value `warning`
- `para[5][@type="warning"]` selects the fifth `para` child of the context node if that child has a `type` attribute with value `warning`
- `chapter[title="Introduction"]` selects the `chapter` children of the context node that have one or more `title` children with string-value equal to `Introduction`
- `chapter[title]` selects the `chapter` children of the context node that have one or more `title` children
- `employee[@secretary and @assistant]` selects all the `employee` children of the context node that have both a `secretary` attribute and an `assistant` attribute

The most important abbreviation is that `child::` can be omitted from a location step. In effect, `child` is the default axis. For example, a location path `div/para` is short for

Xpath can be used with **many** other technologies!

Here are some examples:

- Java
  - -Applications
  - -Applets
  - -Servlets
  - -JSPs
- PHP
- MS .NET
- XSLT
- Xquery (Apache Xindice)
- XML Schema

Simple XML example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
  <cd country="USA">
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <price>10.90</price>
  </cd>
  <cd country="UK">
    <title>Hide your heart</title>
    <artist>Bonnie Tyler</artist>
    <price>9.90</price>
  </cd>
  <cd country="USA">
    <title>Greatest Hits</title>
    <artist>Dolly Parton</artist>
    <price>9.90</price>
  </cd>
</catalog>
```

XPath Syntax

If the path starts with a slash (/) it represents an absolute path to an element!

If the path starts with two slashes (//) then all elements in the document that fulfill the criteria will be selected (even if they are at different levels in the XML tree)!

The following XPath expression selects all the cd elements in the document:

//cd

XPath Expressions

This selects the ROOT element catalog:
/catalog

This selects all the cd elements of the catalog element:
/catalog/cd

This selects all the price elements of all the cd elements of the catalog element:
/catalog/cd/price

This selects all the cd elements that have a price element with a value larger than 10.80:
/catalog/cd[price>10.80]

XPath Expressions

Wildcards (*) can be used to select unknown XML elements.

This selects all the child elements of all the cd elements of the catalog element:

/catalog/cd/*

This selects all the price elements that are grandchild elements of the catalog element:

/catalog/*/price

This selects all price elements which have 2 ancestors:

/*/*/price

This selects all elements in the document:

//*

XPath Expressions

This selects the first cd child element of the catalog element:
/catalog/cd[1]

This selects the last cd child element of the catalog element (Note: There is no function named first()):
/catalog/cd[last()]

This selects all the cd elements of the catalog element that have a price element:
/catalog/cd[price]

This selects all the cd elements of the catalog element that have a price element with a value of 10.90:
/catalog/cd[price=10.90]

This selects all the price elements of all the cd elements of the catalog element that have a price element with a value of 10.90:
/catalog/cd[price=10.90]/price

XPath Syntax...

In XPath all attributes are specified by the @ prefix.

This XPath expression selects all attributes named country:
//@country

This XPath expression selects all cd elements which have an attribute named country:
//cd[@country]

This XPath expression selects all cd elements which have any attribute:
//cd[@*]

This XPath expression selects all cd elements which have an attribute named country with a value of 'UK':
//cd[@country='UK']

XPath Examples

```
<Basket>
   <Cherry flavor='sweet'/>
   <Cherry flavor='bitter'/>
   <Cherry/>
   <Apple color='red'/>
   <Apple color='red'/>
   <Apple color='green'/>
   …
</Basket>
```

Select all of the red apples:

//Basket/Apple[@color='red']

XPath Examples

```
<Basket>
   <Cherry flavor='sweet'/>
   <Cherry flavor='bitter'/>
   <Cherry/>
   <Apple color='red'/>
   <Apple color='red'/>
   <Apple color='green'/>
   …
</Basket>
```

Select all cherries that have some flavor:

```
//Basket/Cherry[@flavor]
```

XPath Functions

String Functions

concat() Returns the concatenation of all its arguments. string=concat(val1, val2, ..)

contains() Returns true if the second string is contained within the first string, otherwise it returns false. bool=contains(val,substr)

normalize-space() Removes leading and trailing spaces from a string. string=normalize-space(string)

starts-with() Returns true if the first string starts with the second string, otherwise it returns false. bool=starts-with(string,substr)

string() Converts the value argument to a string. string(value)

string-length() Returns the number of characters in a string. number=string-length(string)

substring() Returns a part of the string in the string argument. string=substring(string,start,length)
substring-after() Returns the part of the string in the string argument that occurs after the substring in the substr argument. string=substring-after(string,substr)
substring-before() Returns the part of the string in the string argument that occurs before the substring in the substr argument. string=substring-before(string,substr)

translate() Performs a character by character replacement. It looks in the value argument for characters contained in string1, and replaces each character for the one in the same position in the string2. string=translate(value,string1,string2)

XPath Functions

String Functions (Examples)

```
concat('The',' ','XML')              Result: 'The XML'
contains('XML','X')                  Result: true
normalize-space(' The   XML ')       Result: 'The XML'

starts-with('XML','X')               Result: true
string(314)                          Result: '314'
string-length('Beatles')             Result: 7

substring('Beatles',1,4)             Result: 'Beat'
substring-after('12/10','/')         Result: '10'
substring-before('12/10','/')        Result: '12'

translate('12:30','30','45')         Result: '12:45'
translate('12:30','03','54')         Result: '12:45'
translate('12:30','0123','abcd')     Result: 'bc:da'
```

XPath Functions

Node Set Functions

count()
Returns the number of nodes in a node-set number=count(node-set)

id()
Selects elements by their unique ID node-set=id(value)

last()
Returns the position number of the last node in the processed node list number=last()

local-name()
Returns the local part of a node. A node usually consists of a prefix, a colon, followed by the local name string=local-name(node)

name()
Returns the name of a node string=name(node)

namespace-uri()
Returns the namespace URI of a specified node uri=namespace-uri(node)

position()
Returns the position in the node list of the node that is currently being processed number=position()

XPath Functions

Number Functions

number() Converts the value argument to a number number=number(value)

Example: **number('100')** Result: 100

round() Rounds the number argument to the nearest integer integer=round(number)

Example: **round(3.14)** Result: 3

sum() Returns the total value of a set of numeric values in a node-set number=sum(nodeset)

Example: **sum(/cd/price)**

ceiling() Returns the smallest integer that is not less than the number argument number=ceiling(number)

Example: **ceiling(3.14)** Result: 4

floor() Returns the largest integer that is not greater than the number argument number=floor(number)

Example: **floor(3.14)** Result: 3

XPath Functions

Boolean Functions

boolean() Converts the value argument to Boolean and returns true or false bool=boolean(value)

false() Returns false false()

Example: number(false()) Result: 0

true() Returns true true()

Example: number(true()) Result: 1

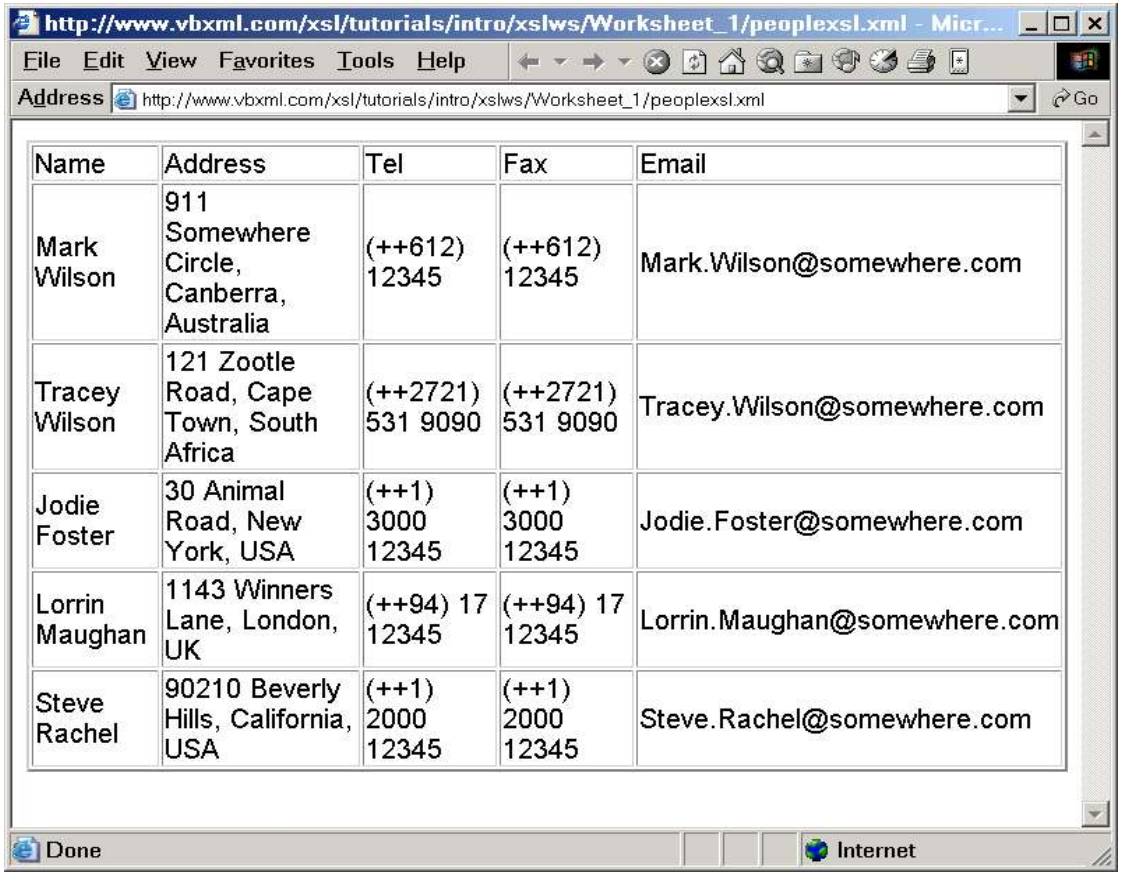
not() Returns true if the condition argument is false, and false if the condition argument is true bool=not(condition)

Example: not(false())

lang() Returns true if the language argument matches the language of the the xsl:lang element, otherwise it returns false bool=lang(language)

XPath in XSLT

```
<?xml version="1.0"?>
 <!-- ********** Resumes for People ********** -->
<PEOPLE>

 <PERSON PERSONID="p1">    <NAME>Mark Wilson</NAME>
  <ADDRESS>911 Somewhere Circle, Canberra, Australia</ADDRESS>
  <TEL>(++612) 12345</TEL>
  <FAX>(++612) 12345</FAX>
  <EMAIL>markwilson@example.com</EMAIL>
 </PERSON>

 <PERSON PERSONID="p2">
  <NAME>Tracey Wilson</NAME>
  <ADDRESS>121 Zootle Road, Cape Town, South Africa</ADDRESS>
  <TEL>(++2721) 531 9090</TEL>
  <FAX>(++2721) 531 9090</FAX>
  <EMAIL>Tracey Wilson@example.com</EMAIL>
 </PERSON>

 <PERSON PERSONID="p3">
  <NAME>Jodie Foster</NAME>
  <ADDRESS>30 Animal Road, New York, USA</ADDRESS>
  <TEL>(++1) 3000 12345</TEL>
   <FAX>(++1) 3000 12345</FAX>
  <EMAIL>Jodie Foster@example.com</EMAIL>
 </PERSON>

 <PERSON PERSONID="p4">
  <NAME>Lorrin Maughan</NAME>
  <ADDRESS>1143 Winners Lane, London, UK</ADDRESS>
  <TEL>(++94) 17 12345</TEL>    <FAX>(++94) 17 12345</FAX>
  <EMAIL>Lorrin Maughan@example.com</EMAIL>
 </PERSON>

 <PERSON PERSONID="p5">
  <NAME>Steve Rachel</NAME>
  <ADDRESS>90210 Beverly Hills, California, USA</ADDRESS>
  <TEL>(++1) 2000 12345</TEL>    <FAX>(++1) 2000 12345</FAX>
  <EMAIL>Steve Rachel@example.com</EMAIL>
 </PERSON>

</PEOPLE>
```



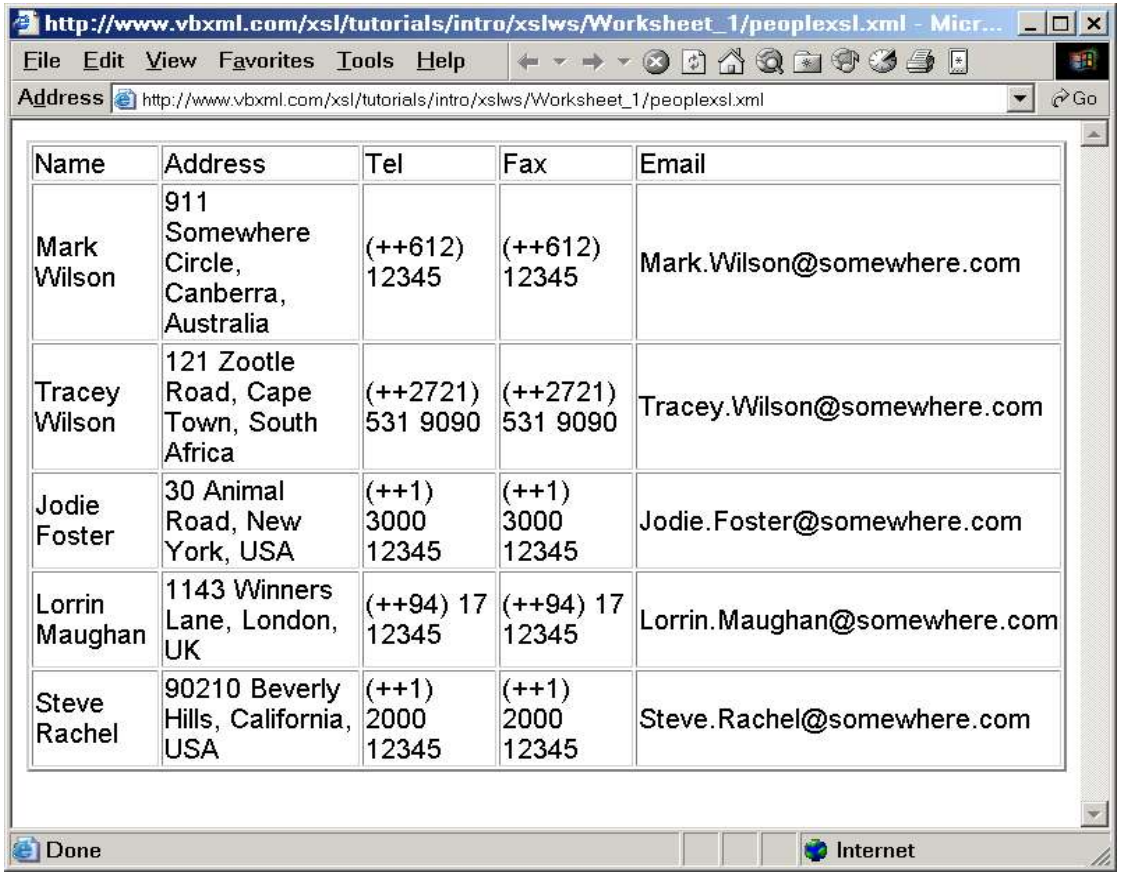
Name	Address	Tel	Fax	Email
Mark Wilson	911 Somewhere Circle, Canberra, Australia	(++612) 12345	(++612) 12345	Mark.Wilson@somewhere.com
Tracey Wilson	121 Zootle Road, Cape Town, South Africa	(++2721) 531 9090	(++2721) 531 9090	Tracey.Wilson@somewhere.com
Jodie Foster	30 Animal Road, New York, USA	(++1) 3000 12345	(++1) 3000 12345	Jodie.Foster@somewhere.com
Lorrin Maughan	1143 Winners Lane, London, UK	(++94) 17 12345	(++94) 17 12345	Lorrin.Maughan@somewhere.com
Steve Rachel	90210 Beverly Hills, California, USA	(++1) 2000 12345	(++1) 2000 12345	Steve.Rachel@somewhere.com

XPath in XSLT

eXtensible Stylesheet Language Transformation ...transform an XML file into an HTML file or another text-based format...

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<HTML>
 <BODY>
  <TABLE BORDER="2">
            <TR>
                  <TD>Name</TD>
                  <TD>Address</TD>
                  <TD>Tel</TD>
                  <TD>Fax</TD>
                  <TD>Email</TD>
            </TR>
            <xsl:for-each select="PEOPLE/PERSON">
            <TR>
                  <TD><xsl:value-of select="NAME"/></TD>
                  <TD><xsl:value-of select="ADDRESS"/></TD>
                  <TD><xsl:value-of select="TEL"/></TD>
                  <TD><xsl:value-of select="FAX"/></TD>
                  <TD><xsl:value-of select="EMAIL"/></TD>
            </TR>
            </xsl:for-each>
  </TABLE></BODY></HTML>
</xsl:template>
</xsl:stylesheet>
```

Xpath Expression



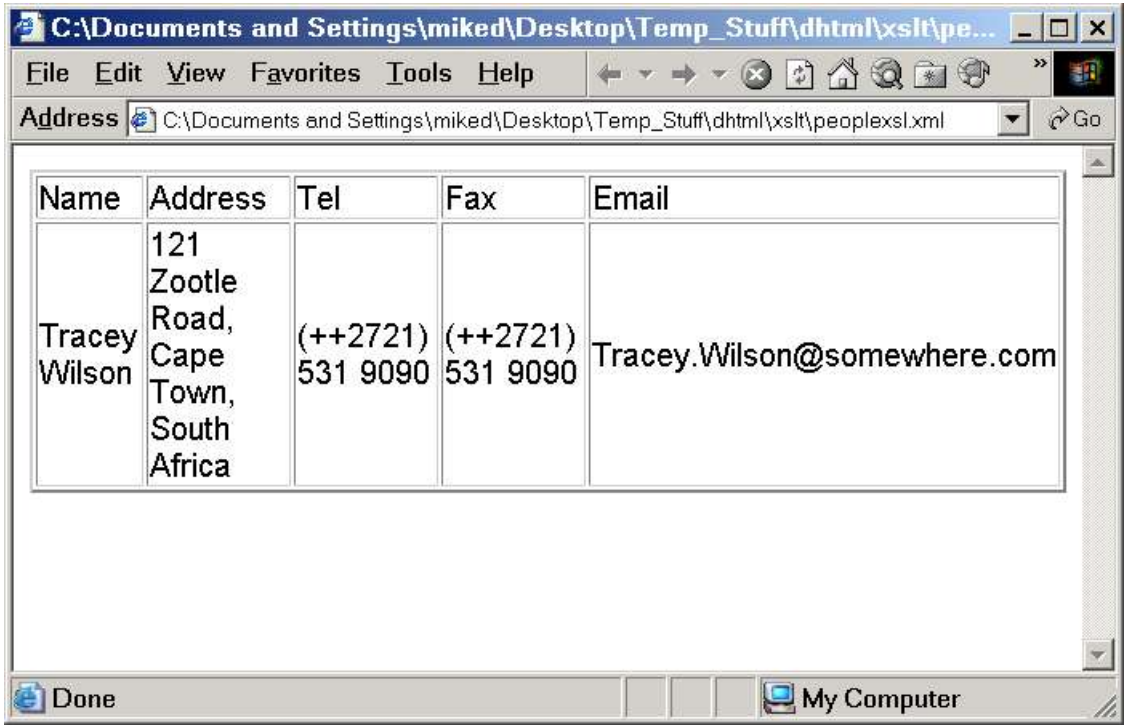
Name	Address	Tel	Fax	Email
Mark Wilson	911 Somewhere Circle, Canberra, Australia	(++612) 12345	(++612) 12345	Mark.Wilson@somewhere.com
Tracey Wilson	121 Zootle Road, Cape Town, South Africa	(++2721) 531 9090	(++2721) 531 9090	Tracey.Wilson@somewhere.com
Jodie Foster	30 Animal Road, New York, USA	(++1) 3000 12345	(++1) 3000 12345	Jodie.Foster@somewhere.com
Lorrin Maughan	1143 Winners Lane, London, UK	(++94) 17 12345	(++94) 17 12345	Lorrin.Maughan@somewhere.com
Steve Rachel	90210 Beverly Hills, California, USA	(++1) 2000 12345	(++1) 2000 12345	Steve.Rachel@somewhere.com

XPath in XSLT

eXtensible Stylesheet Language Transformation ...transform an XML file into an HTML file or another text-based format...

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<HTML>
 <BODY>
  <TABLE BORDER="2">
            <TR>
                  <TD>Name</TD>
                  <TD>Address</TD>
                  <TD>Tel</TD>
                  <TD>Fax</TD>
                  <TD>Email</TD>
            </TR>
            <xsl:for-each select="PEOPLE/PERSON[NAME='Tracey Wilson']">
            <TR>
                  <TD><xsl:value-of select="NAME"/></TD>
                  <TD><xsl:value-of select="ADDRESS"/></TD>
                  <TD><xsl:value-of select="TEL"/></TD>
                  <TD><xsl:value-of select="FAX"/></TD>
                  <TD><xsl:value-of select="EMAIL"/></TD>
            </TR>
            </xsl:for-each>
  </TABLE></BODY></HTML>
</xsl:template>
</xsl:stylesheet>
```

Xpath Expression



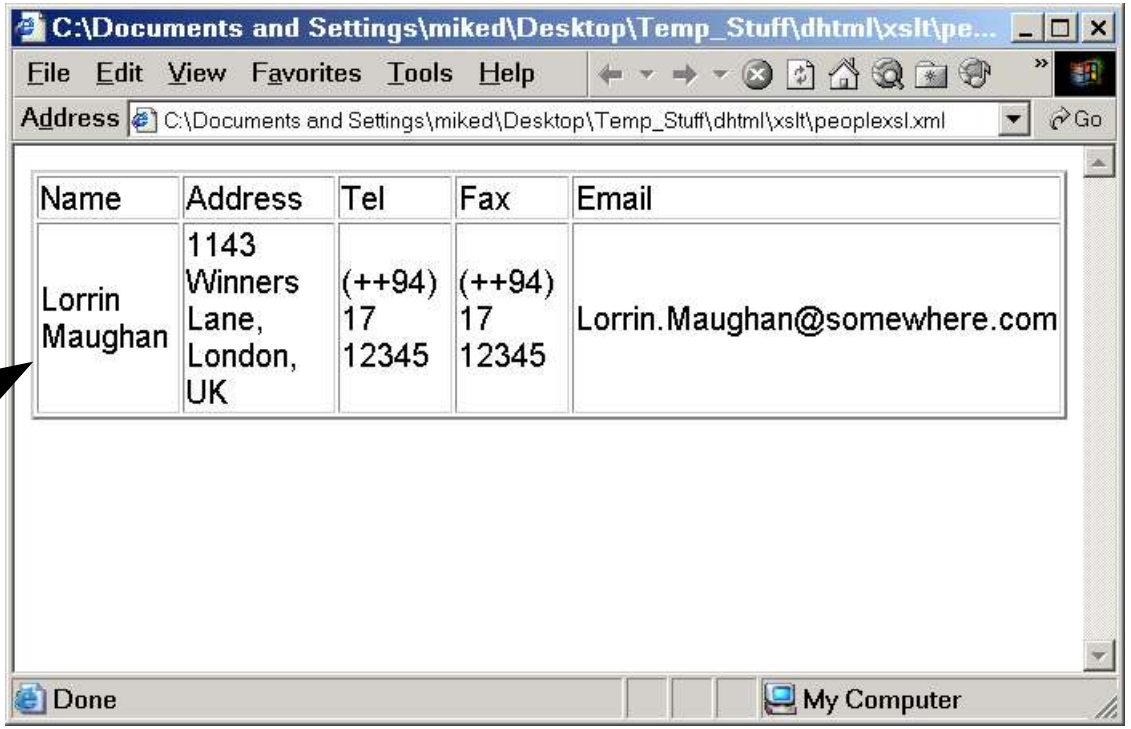
XPath in XSLT

eXtensible Stylesheet Language Transformation ...transform an XML file into an HTML file or another text-based format...

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<HTML>
 <BODY>
  <TABLE BORDER="2">
            <TR>
                  <TD>Name</TD>
                  <TD>Address</TD>
                  <TD>Tel</TD>
                  <TD>Fax</TD>
                  <TD>Email</TD>
            </TR>
            <xsl:for-each select="PEOPLE/PERSON[3]">
            <TR>
                  <TD><xsl:value-of select="NAME"/></TD>
                  <TD><xsl:value-of select="ADDRESS"/></TD>
                  <TD><xsl:value-of select="TEL"/></TD>
                  <TD><xsl:value-of select="FAX"/></TD>
                  <TD><xsl:value-of select="EMAIL"/></TD>
            </TR>
            </xsl:for-each>
  </TABLE></BODY></HTML>
</xsl:template>
</xsl:stylesheet>
```

Xpath Expression

4th person



XPath in **Java** Applications

Where are the classes?

They come with Java 1.4 (aka: “rt.jar”)

Common import statements we will use:

```
import javax.xml.parsers.*;

import org.w3c.dom.*;

import org.apache.xpath.*;
```

XPath in **Java** Applications

You first need an XML document

Convenience method

```
public static Document getDoc(String file) {
org.w3c.dom.Document doc = null;
  System.out.println("Parsing XML file " + file);
  try{
      javax.xml.parsers.DocumentBuilderFactory docBuilderFactory =
      javax.xml.parsers.DocumentBuilderFactory.newInstance();
      javax.xml.parsers.DocumentBuilder docBuilder =
          docBuilderFactory.newDocumentBuilder();
      doc = docBuilder.parse(new File( file ));
      doc.getDocumentElement().normalize();
      return doc;
  }catch (Exception e) {
  System.out.println(e); e.printStackTrace();
  return doc;
  }
}
```

Note: the **parse** method's alternative parms include :
InputSource,
InputStream, and
String uri

XPath in **Java** Applications


```
<PRODUCTS>
  <PRODUCT ACCOUNT="1111111" id="111">
      <DESCRIPTION>Alexander Apples</DESCRIPTION>
      <PREVIOUSCOUNTDATE>2003-12-31</PREVIOUSCOUNTDATE>
      <PREVIOUSCOUNT>10</PREVIOUSCOUNT>
      <THISCOUNT>7</THISCOUNT>
      <SOLD>27</SOLD>
      <ADDED>24</ADDED>
      <UOM>BAG</UOM>
      <COST>10.10</COST>
      <RETAIL>15.12</RETAIL>
      <PRODUCTLOCATION>Counter-C01R01</PRODUCTLOCATION>
      <SERIALNUMBER></SERIALNUMBER>
      <MODEL>Fruit</MODEL>
      <EXPIRATIONDATE>2003-12-31</EXPIRATIONDATE>
  </PRODUCT>
  <PRODUCT ACCOUNT="222222" id="222">
      <DESCRIPTION>Baldwin Apples</DESCRIPTION>
      <PREVIOUSCOUNTDATE>2003-12-31</PREVIOUSCOUNTDATE>
      <PREVIOUSCOUNT>20</PREVIOUSCOUNT>
      <THISCOUNT>13</THISCOUNT>
      <SOLD>37</SOLD>
      <ADDED>12</ADDED>
      <UOM>BAG</UOM>
      <COST>10.10</COST>
      <RETAIL>15.12</RETAIL>
      <PRODUCTLOCATION>Counter-C01R01</PRODUCTLOCATION>
      <SERIALNUMBER></SERIALNUMBER>
      <MODEL>Fruit</MODEL>
      <EXPIRATIONDATE>2003-12-31</EXPIRATIONDATE>
  </PRODUCT>
...
</PRODUCTS>
```

XPath in **Java** Applications

```
package com.MobileSales.test;

import java.io.*;
import org.w3c.dom.*;
import javax.xml.transform.*;
import org.apache.xpath.*;

public class XPathApp {
        static String fileToParse  = "./Data/Inventory_sample.xml";
        public static void main(String[] args) {
                String xPath = "/PRODUCTS/PRODUCT";
                try{org.w3c.dom.Document doc=
                                com.MobileSales.Util.getDoc(new FileInputStream(fileToParse), fileToParse);

                        System.out.println("Show first DOM Node xpath:"+ xPath);
                        System.out.println(XPathAPI.eval(doc, xPath).toString());

                }catch(TransformerException te){ System.out.println(te + "\n Tranformer error!");
                }catch(FileNotFoundException fne){System.out.println(fne + "\n File not found!");}
        }
}
```

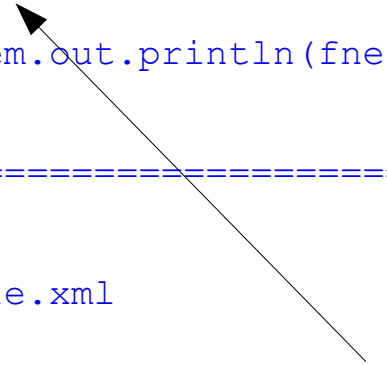
==========

Output

```
Parsing XML file ./Data/Inventory_sample.xml
Show first DOM Node xpath:/PRODUCTS/PRODUCT

        Alexander Apples
        2003-12-31
        10
        7
        27
        24
        BAG
        10.10
        15.12
        Counter-C01R01

        Fruit
        2003-12-31
```

XPath in **Java** Applications

```
public static void main(String[] args) {
 String xPath = "/PRODUCTS/PRODUCT";
 try{
  System.out.println("Parsing XML file "+ fileToParse);
  Document doc=com.MobileSales.Util.getDoc(new FileInputStream(fileToParse), fileToParse);

  org.w3c.dom.Node root = doc.getDocumentElement();
  System.out.println("Root element is " + root.getNodeName());

 }catch(FileNotFoundException fne){System.out.println(fne + "\n File not found!");}
}

=============================================================================
Output:

 Parsing XML file ./Data/Inventory_sample.xml
 Root element is PRODUCTS
```



Element org.w3c.dom.Document.getDocumentElement()

This is a convenience attribute that allows direct access to the child node that is the root element of the document. For HTML documents, this is the element with the tagName "HTML".

XPath in **Java** Applications

```
public static void main(String[] args) {
 String xPath = "/PRODUCTS/PRODUCT";
 try{
    System.out.println("Parsing XML file "+ fileToParse);
    Document doc=
        com.MobileSales.Util.getDoc(new FileInputStream(fileToParse), fileToParse);

    javax.xml.transform.Transformer serializer
        = TransformerFactory.newInstance().newTransformer();
    serializer.setOutputProperty( OutputKeys.OMIT_XML_DECLARATION, "yes");
    System.out.println("subtree "+ xPath);
    org.w3c.dom.traversal.NodeIterator nl = XPathAPI.selectNodeIterator(doc, xPath);
    Node n;
    while ((n = nl.nextNode()) != null) {
        serializer.transform(
            new javax.xml.transform.dom.DOMSource(n),
            new javax.xml.transform.stream.StreamResult(System.out));
    }
 }catch(FileNotFoundException fne){System.out.println(fne + "\n File not found!");
 }catch(javax.xml.transform.TransformerException te){System.out.println(te);}
}
```

void javax.xml.transform.Transformer.transform(
Source xmlSource, Result outputTarget)
throws TransformerException

Process the source tree to the output result.

Parameters:

xmlSource The input for the source tree.

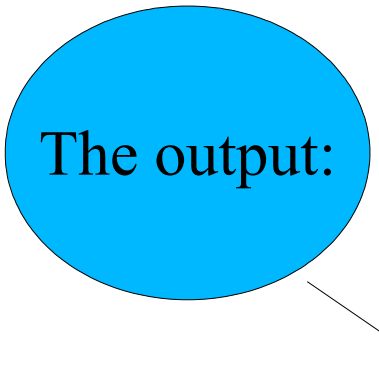
outputTarget The output target.

Throws:

TransformerException If an unrecoverable error occurs during the course of the transformation.

NodeIterators are used to step through a set of nodes in a NodeList, the document subtree governed by a particular Node, the results of a query, or any other set of nodes.

XPath in **Java** Applications

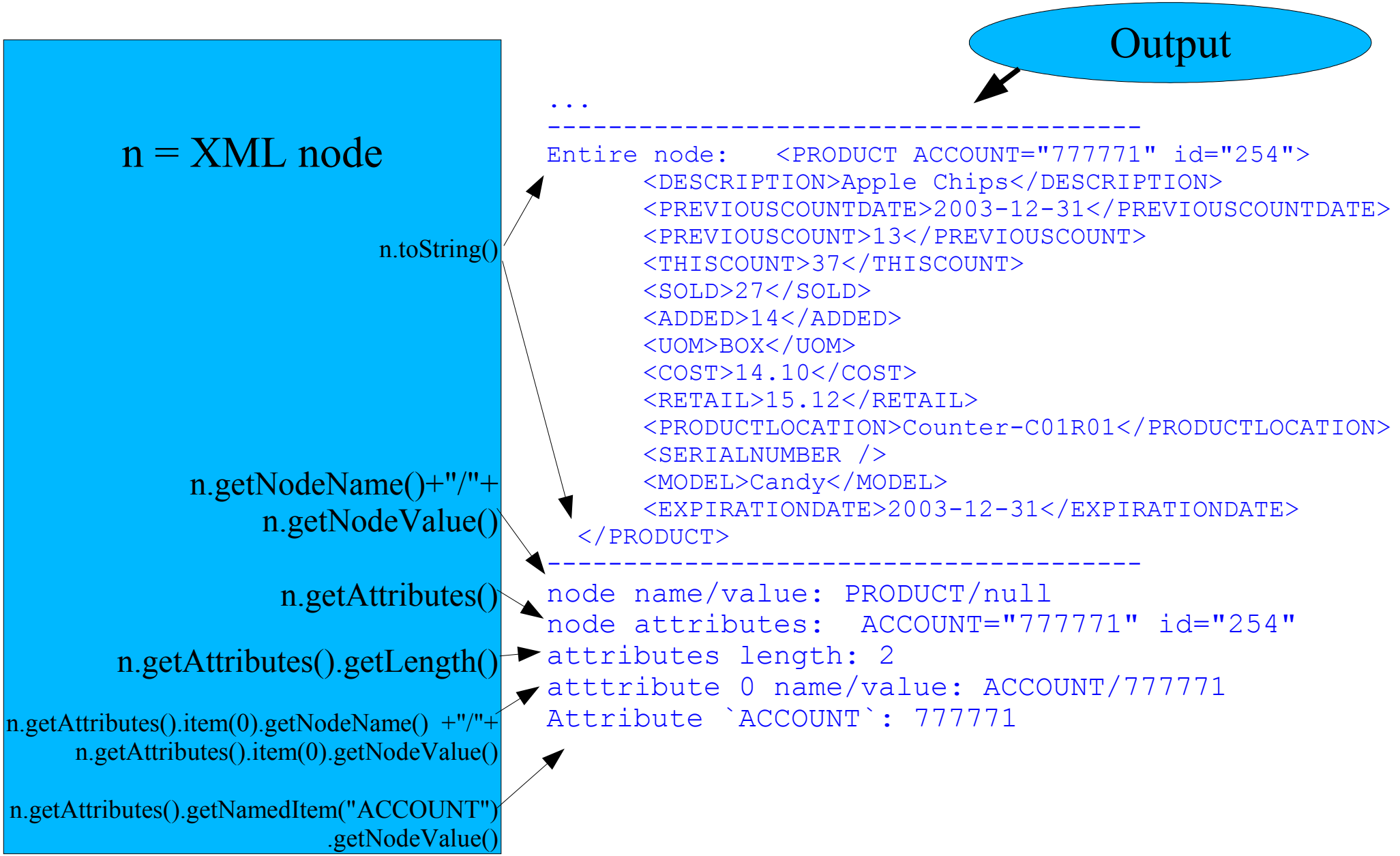



```
Parsing XML file ./Data/Inventory_sample.xml

Show all sub-tree /PRODUCTS/PRODUCT

<PRODUCT ACCOUNT="1111111" id="111">
      <DESCRIPTION>Alexander Apples</DESCRIPTION>
      <PREVIOUSCOUNTDATE>2003-12-31</PREVIOUSCOUNTDATE>
      <PREVIOUSCOUNT>10</PREVIOUSCOUNT>
      <THISCOUNT>7</THISCOUNT>
      <SOLD>27</SOLD>
      <ADDED>24</ADDED>
      <UOM>BAG</UOM>
      <COST>10.10</COST>
      <RETAIL>15.12</RETAIL>
      <PRODUCTLOCATION>Counter-C01R01</PRODUCTLOCATION>
      <SERIALNUMBER/>
      <MODEL>Fruit</MODEL>
      <EXPIRATIONDATE>2003-12-31</EXPIRATIONDATE>
  </PRODUCT><PRODUCT ACCOUNT="222222" id="222">
      <DESCRIPTION>Baldwin Apples</DESCRIPTION>.....
```

XPath in **Java** Applications

```
public static void main(String[] args) {
  String xpath="/PRODUCTS/PRODUCT";

  try{Document doc=
        com.MobileSales.Util.getDoc(new FileInputStream(fileToParse), fileToParse);

    NodeIterator ni = XPathAPI.selectNodeIterator(doc, xpath);
    Node n;

    while ((n = ni.nextNode()) != null) {
       System.out.println("---------------------------------------" );
       System.out.println("Entire node:   "+n.toString() );
       System.out.println("---------------------------------------" );
       System.out.println("node name/value:"+n.getNodeName()+"/"
            +XPathAPI.eval(n,"text()").str() );
       System.out.println("node attributes:"   +n.getAttributes() );
       System.out.println("attributes length:" +n.getAttributes().getLength() );
       System.out.println("atttribute 0 name/value: "
            +n.getAttributes().item(0).getNodeName()
            +"/"+n.getAttributes().item(0).getNodeValue() );
       System.out.println( "Attribute `ACCOUNT`: "
            +n.getAttributes().getNamedItem("ACCOUNT").getNodeValue()  );
     }

  }catch(FileNotFoundException fne){System.out.println(fne + "\n File not found!");
  }catch(javax.xml.transform.TransformerException te){System.out.println(te);}
}
```

XPath in **Java** Applications



XPath in Java Servlets

Populate database table by looping through XML

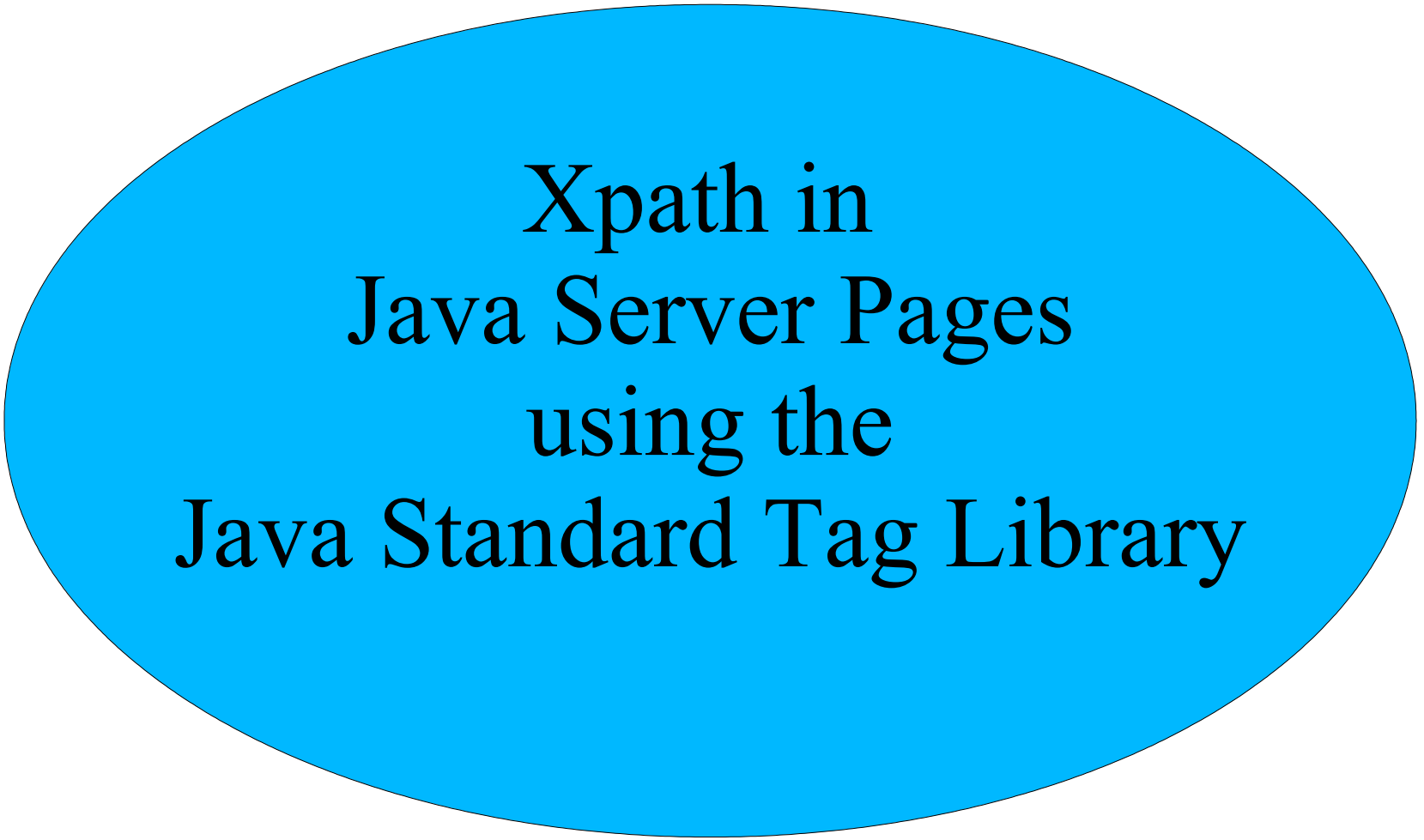
```
public class AllInventoryTable extends HttpServlet {
       String sql, dbDriver, dbURL, db2eTable="INVENTORY",
       XMLFile="Data/Inventory_sample.xml";

       Connection con; Statement stmt; PrintWriter out;
       org.w3c.dom.Node product, productDetail;
       public void doGet(HttpServletRequest req, HttpServletResponse res)
       throws javax.servlet.ServletException, java.io.IOException {
       try{out=res.getWriter();
              stmt=null; con=null;
              con=Util.getLocalDataBaseConnection(out);
              stmt = con.createStatement();

              sql="DROP table " +db2eTable;
              Util.logIt(sql, out);
              try{Util.logIt("Result=" + stmt.executeUpdate(sql), out); }catch(Exception e){}
              sql="CREATE table "+db2eTable +" ("+TableDefinitions.Inventory_FieldDefs+")";
              Util.logIt(sql, out);
              Util.logIt("rst=" +stmt.executeUpdate(sql), out);

              // insert recs from inventory.xml
              Util.logIt("insert recs from "+XMLFile, out);
              Document doc =
                     Util.getDoc(getServletContext().getResourceAsStream(XMLFile), XMLFile);
              NodeIterator products = XPathAPI.selectNodeIterator(doc, "/PRODUCTS/PRODUCT");
              Util.logIt("looping thru xml");
              while ((product = products.nextNode()) != null) {
                sql="INSERT INTO "+db2eTable+" ("+TableDefinitions.Inventory_Fields+" ) VALUES("
                     +"   '"+ product.getAttributes().getNamedItem("ACCOUNT").getNodeValue() +"'"
                     +" , CURRENT DATE "
                     +" , '"+ product.getAttributes().getNamedItem("id").getNodeValue() +"'"
                     +" , '"+ XPathAPI.eval(product,"DESCRIPTION") +"'"
                     +" ,  "+ XPathAPI.eval(product,"PREVIOUSCOUNT")
                     +" ,  "+ XPathAPI.eval(product,"THISCOUNT")
                     +" ,  "+ XPathAPI.eval(product,"SOLD")
                     +" ,  "+ XPathAPI.eval(product,"ADDED")
                     +" , '"+ XPathAPI.eval(product,"SERIALNUMBER") +"'"
                     +" , '"+ XPathAPI.eval(product,"PRODUCTLOCATION") +"'"
                     +" , '"+ XPathAPI.eval(product,"UOM") +"'"
                     +" ,  "+ XPathAPI.eval(product,"COST")
                     +" ,  "+ XPathAPI.eval(product,"RETAIL")
                     +" , CURRENT DATE "
                     +" ) ";
                     Util.logIt(sql, out);
                     Util.logIt("sql result: "+stmt.executeUpdate(sql), out);
              }
              Util.logIt("Finished loading " +db2eTable, out);
              Util.logIt("Closing DB Connection", out);
              stmt.close(); con.close();
              }catch(FileNotFoundException fnf){ Util.logIt( "Can't find "+XMLFile+"--" +fnf.getMessage(),out);
              }catch(IOException ioe){ Util.logIt( "io error -->"+ioe.getMessage(), out);
              }catch(NoClassDefFoundError cnfe){ Util.logIt( "DB driver not found! "+cnfe,out);
              }catch(SQLException se){ Util.HandleSQLError(se,out);
              }catch(UnsatisfiedLinkError ule){ Util.logIt( "UnsatisfiedLinkError-->" +ule.getMessage(), out );
              }catch(Exception e){ Util.logIt(e.getMessage(),out); e.printStackTrace();
              }finally{
                 try{Util.logIt("closing DB",out); con.close(); stmt.close(); }catch(Exception e){}
                 try{out.close();}catch(Exception e){}
              }
    }
}
```

XPath in **Java Servlet**

XPath

```
NodeIterator products = XPathAPI.selectNodeIterator(doc, "/PRODUCTS/PRODUCT");

while ((product = products.nextNode()) != null) {
 sql="INSERT INTO "+db2eTable+" ("+TableDefinitions.Inventory_Fields+" ) VALUES("
    +"   '"+ product.getAttributes().getNamedItem("ACCOUNT").getNodeValue() +"'"
    +" , CURRENT DATE "
    +" , '"+ product.getAttributes().getNamedItem("id").getNodeValue() +"'"
    +" , '"+ XPathAPI.eval(product,"DESCRIPTION") +"'"
    +" ,  "+ XPathAPI.eval(product,"PREVIOUSCOUNT")
    +" ,  "+ XPathAPI.eval(product,"THISCOUNT")
    +" ,  "+ XPathAPI.eval(product,"SOLD")
    +" ,  "+ XPathAPI.eval(product,"ADDED")
    +" , '"+ XPathAPI.eval(product,"SERIALNUMBER") +"'"
    +" , '"+ XPathAPI.eval(product,"PRODUCTLOCATION") +"'"
    +" , '"+ XPathAPI.eval(product,"UOM") +"'"
    +" ,  "+ XPathAPI.eval(product,"COST")
    +" ,  "+ XPathAPI.eval(product,"RETAIL")
    +" , CURRENT DATE "
    +" ) ";
 stmt.executeUpdate(sql);
}
```

Exploded view from last slide


Xpath in
Java Server Pages
using the
Java Standard Tag Library

JSTL (Java Standard Tag Library)

...developed by the JSR-52 expert group under the Java Community Process...

...The ultimate goal of JSTL is to help simplify JavaServer™ Pages (JSP™) page authors' lives...

JSTL version 1.1 works with JSP 2.0 and Servlet Spec 2.4

Use the **Servlet 2.4** syntax in your web.xml (web-app deployment descriptor):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app version="2.4"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee web-app_2_4.xsd
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/j2ee"> ...
```

USE JSP 2.0 taglib prefix syntax in your JSP's:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/xml" prefix="x" %>
```

==========

Do NOT use the Servlet 2.3 (or prior) deployment descriptor syntax:

```
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>...
```

Do NOT JSP 1.2 (or prior) taglib prefix syntax:

```
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
```

JSTL (Java Standard Tag Library)

JSTL allows you to use EL (Expression Language)

The EL operators

Arithmetic

+, -, *, / (or div), % (or mod)

Relational

== (or eq), != (or ne), < (or lt), > (or gt), <= (or le), >= (or ge)

Logical

&& (or and), || (or or), ! (or not)

Validation

empty

JSTL (Java Standard Tag Library)

XML Tag Library

```
<x:out select="XPathExpression" [escapeXml="{true|false}"]/>

<x:parse xml="XMLDocument" {var="var" [scope={page|request|session|application}]
                          |varDom="var" [scopeDom="scope"]}
    [systemId="systemId"] [filter="filter"]/>

<x:set select="XPathExpression" var="varName"
    [scope="{page|request|session|application}"]/>

<x:if select="XPathExpression"
    [var="varName"] [scope="{page|request|session|application}"]> body content </x:if>

<x:choose>
    <x:when select="XPathExpr"> body content </x:when>
    <x:otherwise>conditional block</x:otherwise>
</x:choose>

<x:forEach [var="varName"] select="XPathExpression"> body content </x:forEach>

<x:transform xml="XMLDocument" xslt="XSLTStylesheet"
    [xmlSystemId="XMLSystemId"]
    [xsltSystemId="XSLTSystemId"]
    [{var="varName"
        [scope="{page|request|session|application}"]|result="resultObject"}]/>


<x:param name="name" value="value"/>
```

XPath in JSP ("Java Server Page")

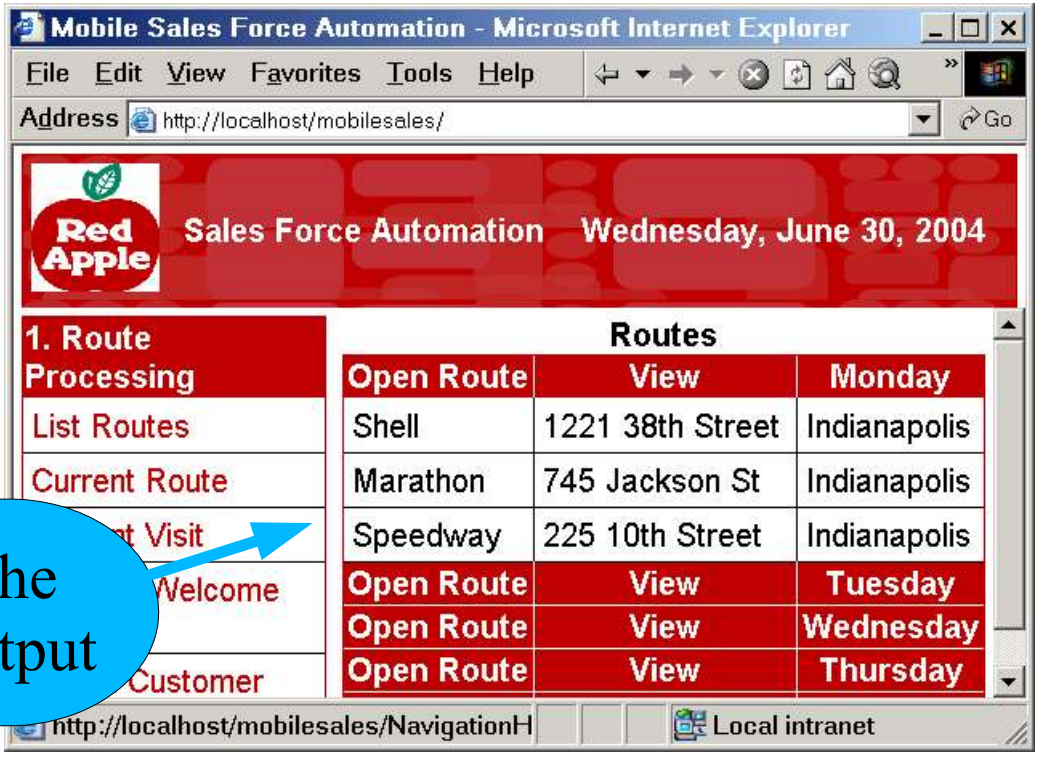
Here is a simple excerpt from a JSP (using jstl):

```
<x:forEach var="STOP" select="$route/STOP">
     <TR id='<x:out select="$route/@ID"/>'>
       <TD>  <x:out select="$STOP/NAME"/>       </TD>
       <TD>  <x:out select="$STOP/ADDRESS"/>    </TD>
       <TD>  <x:out select="$STOP/CITY"/>       </TD>
     </TR>
</x:forEach>
```

XML file

```
<ROUTES>
  <ROUTE ID="Monday">
     <TEXT>Monday</TEXT>
     <STOP ID="101" ACCOUNT="777777">
              <NAME>Shell</NAME>
              <ADDRESS>1221 38th Street</ADDRESS>
              <ADDRESS2></ADDRESS2>
              <CITY>Indianapolis</CITY>
              <STATE>IN</STATE>
              <ZIP>46222</ZIP>
              <CONTACT>Mike Cornwell</CONTACT>
              <PHONE>(317) 272-3854</PHONE>
              <TYPE>Gas Station</TYPE>
              <TAXID>325478521</TAXID>
              <MINORITY>35%</MINORITY>
     </STOP>
     <STOP ID="102" ACCOUNT="777771">
              <NAME>Marathon</NAME>
              <ADDRESS>745 Jackson St</ADDRESS>
              <ADDRESS2></ADDRESS2>
              <CITY>Indianapolis</CITY>
              <STATE>IN</STATE>
              <ZIP>46745</ZIP>
              <CONTACT>Mike Cornwell</CONTACT>
              <PHONE>(317) 272-3854</PHONE>
              <TYPE>Gas Station</TYPE>
              <TAXID>325478521</TAXID>
              <MINORITY>35%</MINORITY>
     </STOP>
     <STOP ID="103" ACCOUNT="777772">...
```

The Output



XPath in JSP (complete example)

```
<html><head> <title> List Routes </title>
 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
 <%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
 <%@ taglib uri="http://java.sun.com/jsp/jstl/xml" prefix="x" %>
 <link   href="../Styles/Global.css" type="text/css" rel="stylesheet" />
 <SCRIPT src ="../Scripts/ExpandCollapse.js" type="text/javascript"></SCRIPT>
</head>

<body onLoad="collapseAll()"> <center>
 <B>Routes</B><BR>
<c:if test="${applicationScope.routesXML==null}">
 <c:import url="../Data/Routes.xml" var="xml"/>
 <x:parse xml="${xml}" var="routesXML" scope="application"/>
</c:if>
<c:if test="${applicationScope.routesXML == null}">
  <BR> Could not find or parse the Routes.xml file <br>
  <BR> Please contact the technical support staff. <BR>
</c:if>
<c:if test="${applicationScope.routesXML != null}">
 <TABLE width="100%">
   <x:forEach var="route" select="$routesXML/ROUTES/ROUTE">
    <TR class="COLLAPSIBLE_MENU" id='<x:out select="$route/@ID" />' onclick="toggle(this.id);">
     <Th><a class="COLLAPSIBLE_MENU" href='Route.jsp?ROUTE=<x:out select="$route/TEXT"/>' target="Page">
          Open Route</a> </Th>
     <Th>View</Th>
     <Th><x:out select="$route/TEXT" /></Th> </TR>
        <x:forEach var="STOP" select="$route/STOP">
           <TR id='<x:out select="$route/@ID"/>'>
             <TD>  <x:out select="$STOP/NAME"/>       </TD>
             <TD>  <x:out select="$STOP/ADDRESS"/>    </TD>
             <TD>  <x:out select="$STOP/CITY"/>       </TD>
           </TR>
         </x:forEach>
   </x:forEach>
 </TABLE>
</c:if>
</center></body></html>
```

Lets us use the JSTL XML tags

Load the XML file into variable “routesXML”

Nested loop through STOP's

Loop through ROUTE's

XPath in JSP

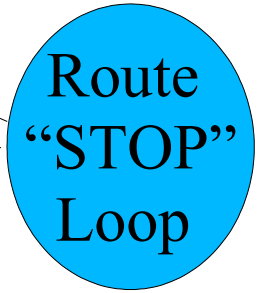
```
...

   <x:forEach var="STOP" select="$routesXML/ROUTES/ROUTE[@ID=$ROUTE_SELECTED]/STOP">
       <TR><x:set var="custNum" select="$STOP/@ACCOUNT"/>
         <TD> <a
href='../VisitOpener?STOP_PARM=<x:out select="$STOP/NAME"/>&CUST_PARM=<x:out select="$STOP/@ACCOUNT"/>' >

                <x:choose>

                  <x:when
select="$activityLogXML/LOG[WHERE=$custNum and contains(WHAT,'CLOS') and substring(WHEN,1,10)=$today]">
                     Edit/Re-open </x:when>

                  <x:when
select="$activityLogXML/LOG[WHERE=$custNum and substring(WHEN,1,10)=$today]">
                     Continue/Close </x:when>

                  <x:otherwise>
                     Start </x:otherwise>

                </x:choose>


          </a> </TD>
         <TD> <x:out select="$STOP/@ACCOUNT"/>    </TD>
         <TD> <x:out select="$STOP/NAME"/>        </TD>
         <TD> <x:out select="$STOP/ADDRESS"/>     </TD>
         <TD> <x:out select="$STOP/CITY"/>        </TD>
       </TR>
   </x:forEach>

...
```

XPath

Choose/When statement

XPath in JSP (Comlete Example)

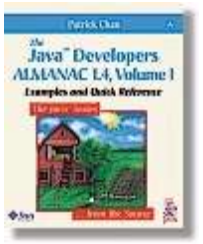
```
<html><head> <title> Route Processing </title>
 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
 <%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
 <%@ taglib uri="http://java.sun.com/jsp/jstl/xml" prefix="x" %>
 <%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
 <link   href="../Styles/Global.css"     type="text/css" rel="stylesheet" />
</head><body> <center>
<c:set var="ACTION_PAGE" value="Route.jsp" scope="session" />
<!-- <BR> sessionScope.ACTION_PAGE="${sessionScope.ACTION_PAGE}" -->
<!-- If no route selected, then list routes (List_Routes.jsp) -->
<c:if test="${param.ROUTE==null && sessionScope.ROUTE_SELECTED==null}">
        <c:redirect url="List_Routes.jsp" />
</c:if>
<c:if test="${param.ROUTE!=null}">
  <c:set var="ROUTE_SELECTED" value="${param.ROUTE}" scope="session" />
</c:if>
 <B> ${sessionScope.ROUTE_SELECTED} Route</B><BR>
<!-- Load routes.xml file -->
<c:if test="${applicationScope.routesXML==null}">
 <c:import url="../Data/Routes.xml" var="xml"/>
 <x:parse xml="${xml}" var="routesXML" scope="application"/>
</c:if>
<c:if test="${applicationScope.routesXML == null}">
  <BR> Could not find or parse the Routes.xml file <br>
  <BR> Please contact the technical support staff. <BR>
</c:if>
<!-- Load ActivityLog.xml file -->
<c:import url="../Data/ActivityLog.xml" var="xml"/>
<x:parse xml="${xml}" var="activityLogXML" scope="request"/>
<c:if test="${activityLogXML == null}">
  <BR> Could not find or parse the ActivityLog.xml file <br>
  <BR> Please contact the technical support staff. <BR>
</c:if>
<!-- assign var "today" for comparisons/searching through the activitiy log (see below on this page) -->

<jsp:useBean id="now" class="java.util.Date" />
<fmt:formatDate var="today" pattern="yyyy-MM-dd" value="${now}" />
<c:if test="${applicationScope.routesXML != null}">
 <TABLE width=100%>
         <TR>
           <TH> Action               </TH>
           <TH> Account#      </TH>
           <TH> Name          </TH>
           <TH> Address       </TH>
           <TH> City          </TH>
         </TR>
   <x:forEach var="STOP" select="$routesXML/ROUTES/ROUTE[@ID=$ROUTE_SELECTED]/STOP">
         <TR><x:set var="custNum" select="$STOP/@ACCOUNT"/>
           <TD> <a href='../VisitOpener?STOP_PARM=<x:out select="$STOP/NAME"/>&CUST_PARM=<x:out select="$STOP/@ACCOUNT"/>' >
                     <!-- when=<x:out select="$activityLogXML/LOG[substring(WHEN,1,8)]" /> -->
                     <x:choose>
                       <x:when select="$activityLogXML/LOG[WHERE=$custNum and contains(WHAT,'CLOS') and substring(WHEN,1,10)=$today]">
                            Edit/Re-open </x:when>
                       <x:when select="$activityLogXML/LOG[WHERE=$custNum and substring(WHEN,1,10)=$today]">
                            Continue/Close </x:when>
                       <x:otherwise>
                            Start </x:otherwise>
                     </x:choose>
              </a> </TD>
           <TD> <x:out select="$STOP/@ACCOUNT"/>    </TD>
           <TD> <x:out select="$STOP/NAME"/>               </TD>
           <TD> <x:out select="$STOP/ADDRESS"/>     </TD>
           <TD> <x:out select="$STOP/CITY"/>        </TD>
         </TR>
   </x:forEach>
 </TABLE>
</c:if>
</center></body></html>
```



XPath Links

www.w3.org/TR/xpath

www.w3schools.com/xpath/default.asp



javaalmanac.com/egs/org.w3c.dom/pkg.html

xml.apache.org/xalan-j/apidocs/org/apache/xpath/package-summary.html

java.sun.com/j2se/1.5.0/docs/api/javax/xml/xpath/package-summary.html