HTML Parseing

Laboratory of Service Design and Engineering

2011/2012

Outline

- XPath Overview
- Java & XPath
- Java & HTML Parser
- Exercises

Introduction to XPath

- XPath is a query language designed for querying XML documents
- XPath uses path expressions to navigate in XML documents
- XPath contains a library of standard functions
- XPath describes paths to elements in a similar way an operating system describes paths to files
- XPath is a W3C recommendation
  - http://www.w3.org/TR/xpath20/

XML Nodes & Relationship

- library is the **parent of book; book is** the **parent of the two chapters**
- The two chapters are the **children of** book, and the section is the **child of** the second chapter
- The two chapters of the book are **siblings (they have the same parent)**
- Library, book, and the second chapter are the **ancestors of the section**
- The two chapters, the section, and the two paragraphs are the **descendents of the book**

```
<library>
  <book>

    <chapter>
    </chapter>

    <chapter>
      <section>
        <paragraph/>
        <paragraph/>
      </section>
    </chapter>

  </book>
</library>
```

XML Document

```
<bookstore>
    <book year="2000">
        <title lang="eng">Snow Crash</title>
        <author>Neal Stephenson</author>
        <publisher>Spectra</publisher>
        <isbn>0553380958</isbn>
        <price>14.95</price>
    </book>

    <book year="2005">
        <title>Burning Tower</title>
        <author>Larry Niven</author>
        <author>Jerry Pournelle</author>
        <publisher>Pocket</publisher>
        <isbn>0743416910</isbn>
        <price>5.99</price>
    </book>

    <book year="1995">
        <title>Zodiac</title>
        <author>Neal Stephenson</author>
        <publisher>Spectra</publisher>
        <isbn>0553573862</isbn>
        <price>7.50</price>
    </book>
</bookstore>
```

Node Selection

- A path that begins with a / represents an absolute path, starting from the top of the document **/bookstore/book/title**
- Note: an absolute path can select more than one element
- A / by itself means "*the whole document*"
- A path that does not begin with a / represents a path *starting from the current element*
- **book/title** Selects all title elements that are children of book

Node Selection

- A path that begins with // can start from anywhere in the document
- //title Selects every element title, no matter where it is
- bookstore//title Selects all title elements that are descendant of the bookstore element, no matter where they are under the bookstore element

Predicates

- Predicates are used to find a specific node or a node that contains a specific value.
- **/bookstore/book[1]** Selects the first book element that is the child of the bookstore element.
- **/bookstore/book[last()]** Selects the last book element that is the child of the bookstore element
- **/bookstore/book[position()<3]** Selects the
- first two book elements that are children of the bookstore element
- **/bookstore/book[price>3]** Selects all the book elements of the bookstore element that have a price element with a value greater than 3

Attributes

- You can select attributes by themselves, or elements that have certain attributes
- To choose the attribute itself, prefix the name with **@**
- //**@lang** Selects all attributes that are named lang
- To choose elements that have a given attribute, put the attribute name in square brackets
- //**title**[**@lang**='**eng**'] Selects all the title elements that have an attribute named lang with a value of 'eng'

Wildcards

- * Matches all element node at this level
- /bookstore/* Selects all the children nodes of the bookstore element
- @* Matches all attribute node
- //title[@*] Selects all title elements which have any attribute
- node() Matches any node of any kind

Selecting Several Paths

- By using the | operator in an XPath expression you can select several paths.
- //book/title | //book/price
  - Selects all the title and price elements of all book elements
- /bookstore/book/title | //price
  - Selects all the title elements of the book element of the bookstore element and all the price elements in the document

Axes

- An axis is a set of nodes relative to a given node
  - X::Y means “choose Y from the X axis”
- child::book Selects all book nodes that are children of the current node
- child::text() Selects all text child nodes of the current node
- child::node() Selects all child nodes of the current node
- ancestor::book Selects all book ancestors of the current node
- ……

Arithmetic Operators

- + add
- - subtract
- * multiply
- div (not /) divide
- mod modulo (remainder)

Boolean Operators

- = equals (Notice it's not ==)
- != not equals
- value = node-set will be true if the node-set contains any node with a value that matches value
- value != node-set will be true if the node-set contains any node with a value that does not match value
- Hence, value = node-set and value != node-set may both be true at the same time!

Boolean Operators

- And
- Or
- not()
- The following are used for numerical comparisons only:
  - <, <=, >, >=

XPath and Java

javax.xml.xpath

Class/Interface	Description
XpathFactory	Used to create an XPath object.
XPath	Provides access to the XPath evaluation environment. Provides the **evaluate** methods to evaluate XPath expressions in an DOM tree.
XPathExpression	Provides the **evaluate** methods to evaluate **compiled** XPath expressions in an XML document.

Java XPath Example

```
public class XPathTest {

       public static void main(String[] args)
        throws ParserConfigurationException, SAXException,
               IOException, XPathExpressionException {

         DocumentBuilderFactory domFactory = DocumentBuilderFactory.newInstance();
         domFactory.setNamespaceAware(true);
         DocumentBuilder builder = domFactory.newDocumentBuilder();
         Document doc = builder.parse("books.xml");

         XPathFactory factory = XPathFactory.newInstance();
         XPath xpath = factory.newXPath();
         //Compile an XPath expression for later evaluation
         XPathExpression expr = xpath.compile("/bookstore/book/title/text()");

         NodeList nodes = (NodeList) expr.evaluate(doc, XPathConstants.NODESET);
         for (int i = 0; i < nodes.getLength(); i++) {
             System.out.println(nodes.item(i).getNodeValue());
         }

       }
}
```

Java and XPath Types

- XPath and Java language do not have identical type systems
- XPath 1.0 has only four basic data types:
  - node-set
  - Number
  - Boolean
  - String
- The evaluate() method may return
  - org.w3c.dom.NodeList
  - java.lang.Double
  - java.lang.Boolean
  - java.lang.String
  - org.w3c.dom.Node
- When you **evaluate an XPath expression in Java,** the second argument specifies the return type you expect.

Exercise

- Download the code and run on your machines
- Use “Employee.xml” file from previous lab
  - Make a function which print all employees list with detail
  - A function which accepts name as parameter and print that particular employee
  - A function which accepts salary and a operator (=, > , <) as parameters and prints employee that fulfill that condition.

Java HTML Parsing

Why we need an HTML parser?

- Common scenario: we want to retrieve information from the web and use it in our application
- Ideal solution: the web site provides a web service that returns a representation of a required resource
- Common solution: we have to crawl pages and, by parsing HTML, extract the needed information
- The latter is a poor solution because:
  - If the structure of the page changes, we have to change our parser too
  - No clear structure of the data, we have to figure out that

HTML vs XML

HTML	XML
Presentation purpose	Custom data definition and exchange between applications
No semantic correlation between tags' names and content, tags define presentation (or navigation) features	Tags give us hints about the content
DTD only DOCTYPE <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" http://www.w3.org/TR/html4/loose.dtd>	External definition of entities, schema (.xsd) or DTD for validation
No customization of tags	Application-specific tags
A not well-formed document is legal	A document MUST be well-formed rules
	Extensions: XQuery, XPath, …

XML parsing API

- Low level XML API
  - SAX (Simple API for XML)
  - DOM (Document Object Model)
  - StAX (Streaming API for XML)
- High level XML API
  - JAXB (Java Architecture for XML Binding): it takes and XML document and provides it as a Java Object
- These API are included in the Java platform

HTML parsing API

- We have to use specific libraries, not included in the Java platform
- There are many open source HTML parsers
  - Cobra 0.95.1
  - HTML Parser
  - HtmlCleaner
  - HotSax
  - Java Mozilla Html Parser
  - NekoHTML
  - source:http://java-source.net/open-source/html-parsers
  - Jericho HTML Parser
  - JTidy
  - TagSoup
  - VietSpider
  - HTMLParser

HTML Parsing API

- We present JTidy
  - JTidy is simple
  - JTidy provides a DOM interface to the document that is being processed
- **JTidy can clean up malformed and faulty HTML documents.**
  - Aims to be a DOM parser for real-world HTML.

JTidy Example

```
public  void parse(String u){
    try {
        URL url = new URL(u);
        BufferedInputStream page = new BufferedInputStream(url.openStream());

        Tidy tidy = new Tidy();
        tidy.setQuiet(true);
        tidy.setShowWarnings(false);
        Document response = tidy.parseDOM(page, System.out);

        XPathFactory factory = XPathFactory.newInstance();
        XPath xPath=factory.newXPath();
        String pattern = "//h3//text()";
        NodeList nodes = (NodeList)xPath.evaluate(pattern, response,
                          XPathConstants.NODESET);
        for (int i = 0; i < nodes.getLength(); i++) {
            System.out.println("node:" + (String) nodes.item(i).getNodeValue());
        }
    } catch (MalformedURLException e) { e.printStackTrace();
    } catch (IOException e) {  e.printStackTrace();
    } catch (XPathExpressionException e) {e.printStackTrace();
    }
}
```

Proxy Settings

```
public class Main {

public static final void setProxy(){
    String PROXY_URL = "proxy.science.unitn.it";
    String PROXY_PORT = "3128";
    System.getProperties().put( "proxySet", "true" );
    System.getProperties().put( "proxyHost", PROXY_URL);
    System.getProperties().put( "proxyPort", PROXY_PORT);
}


public static void main(String[] args){
    Main.setProxy();

    JTidyTest parser = new JTidyTest();
    parser.parse("http://java-source.net/open-source/html-parsers");
}

}
```

Exercise

- Download JTidy from http://jtidy.sourceforge.net/
- Download code from course web site
- Run the code and make changes to check how it works