XPathway 1.0

A graphical XPath 2.0 Testing & Visualization Tool

07.November 2008

Adrian Dabrowski
e0025933@student.tuwien.ac.at

Table of Contents

What is XPathway?

It is a small utility for testing XPath 2.0 Expressions and get familiar with the Saxon Xpath 2.0 Evaluator. It is written in Java 5 to ensure maximum compatibility on multiple platforms. A graphical view of the document tree and of the result set simplifies the understanding of Xpath and XML DOM. The graphical interface enables easy experimenting with XPath Expressions and therefore an easy access to the concepts behind XML and XPath.

You can distribute the application with a custom set of example XML documents embedded in the .jar file without recompiling or repackaging it. The jar file is designed to be started as easy as a double click.

XPathway utilizes the JAXP XML Parser included in Java 5 and the SAXON 9.1 library. XPathway is licensed under the GPL. SAXON is licensed under the Mozilla Public License (MPL).

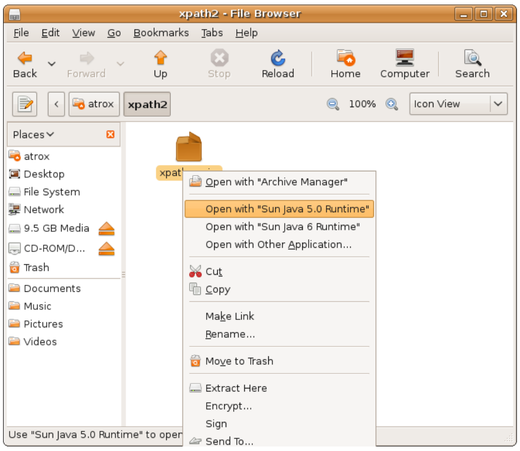
Starting Xpathway

Requirements

You need at least an installed Java Runtime Environment for JAVA5 (JRE 1.5) or JAVA6 (JRE1.6). If you also intend to develop JAVA Software, install the Java Development Kit (JDK) which includes a JRE.

Linux/Unix/OSX

Most distributions enable you to start .jar-files directly out of your graphical file browser. If its the default action, you just need to click or double-click the .jar file. Some distributions like Ubuntu have their archive manager associated with the default action. In that case use the modifier key or right mouse click to open an menu with alternative options.



Use "Sun Java 5.0 Runtime" to oper

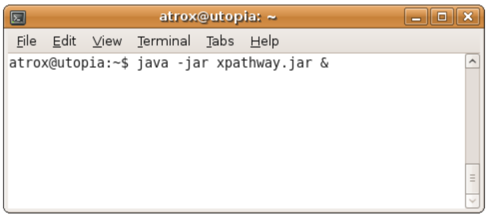
Illustration 1: Ubuntu Gnome

Windows

If you installed SUNs JRE or JDK on your System, .jar should be associated with the JAVA interpreter. Sometimes the Nokia PC Suite can interfere with that.

Command Line

On all platforms the jar file can be started using the command line in a very similar way.
Just type `java -jar xpathway.jar` and hit enter.



```
atrox@utopia:~$ java -jar xpathway.jar &
```

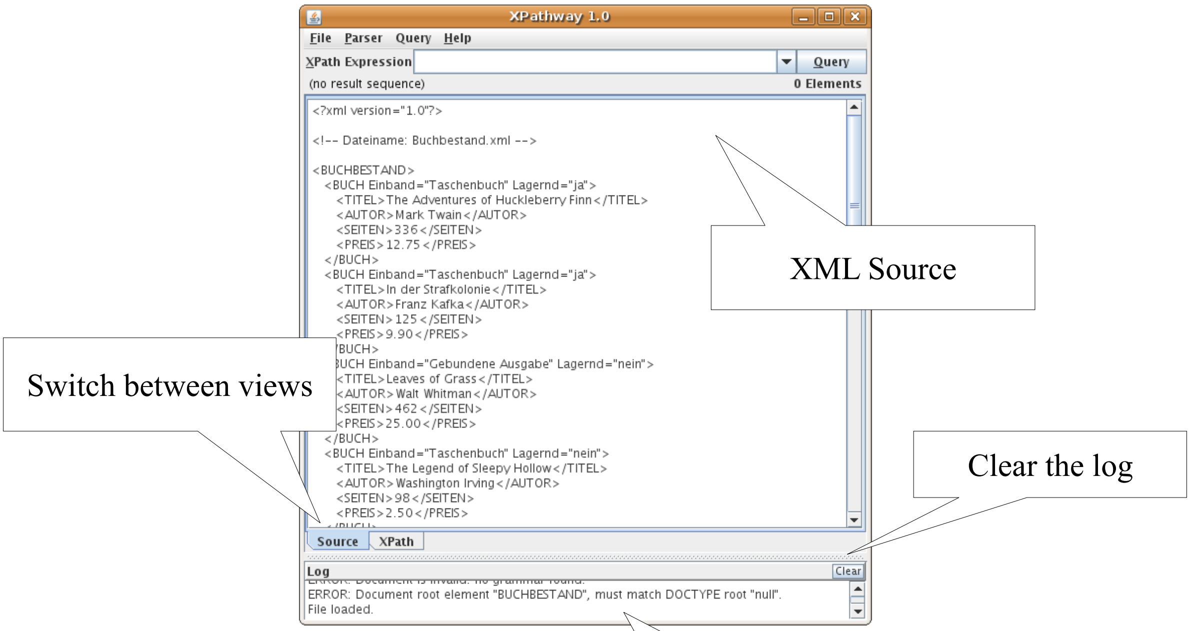
Illustration 2: Ubuntu: Shell

The GUI – basic Structure

Source View

The source view enables the user to see the original source code to see how it is transformed in a document tree.

It is not intended for editing the source. Use an external editor and the “reload”-feature in the “File”-Menu to update the document.



```
<?xml version="1.0"?>

<!-- Dateiname: Buchbestand.xml -->

<BUCHBESTAND>
  <BUCH Einband="Taschenbuch" Lagernd="ja">
    <TITEL>The Adventures of Huckleberry Finn</TITEL>
    <AUTOR>Mark Twain</AUTOR>
    <SEITEN>336</SEITEN>
    <PREIS>12.75</PREIS>
  </BUCH>
  <BUCH Einband="Taschenbuch" Lagernd="ja">
    <TITEL>In der Strafkolonie</TITEL>
    <AUTOR>Franz Kafka</AUTOR>
    <SEITEN>125</SEITEN>
    <PREIS>9.90</PREIS>
   /BUCH>
   BUCH Einband="Gebundene Ausgabe" Lagernd="nein">
    <TITEL>Leaves of Grass</TITEL>
    <AUTOR>Walt Whitman</AUTOR>
    <SEITEN>462</SEITEN>
    <PREIS>25.00</PREIS>
  </BUCH>
  <BUCH Einband="Taschenbuch" Lagernd="nein">
    <TITEL>The Legend of Sleepy Hollow</TITEL>
    <AUTOR>Washington Irving</AUTOR>
    <SEITEN>98</SEITEN>
    <PREIS>2.50</PREIS>
```

```
ERROR: Document is invalid: no grammar found.
ERROR: Document root element "BUCHBESTAND", must match DOCTYPE root "null".
File loaded.
```

XML Source

Switch between views

Clear the log

Illustration 3: Source View

Log keeps information about errors and past actions.

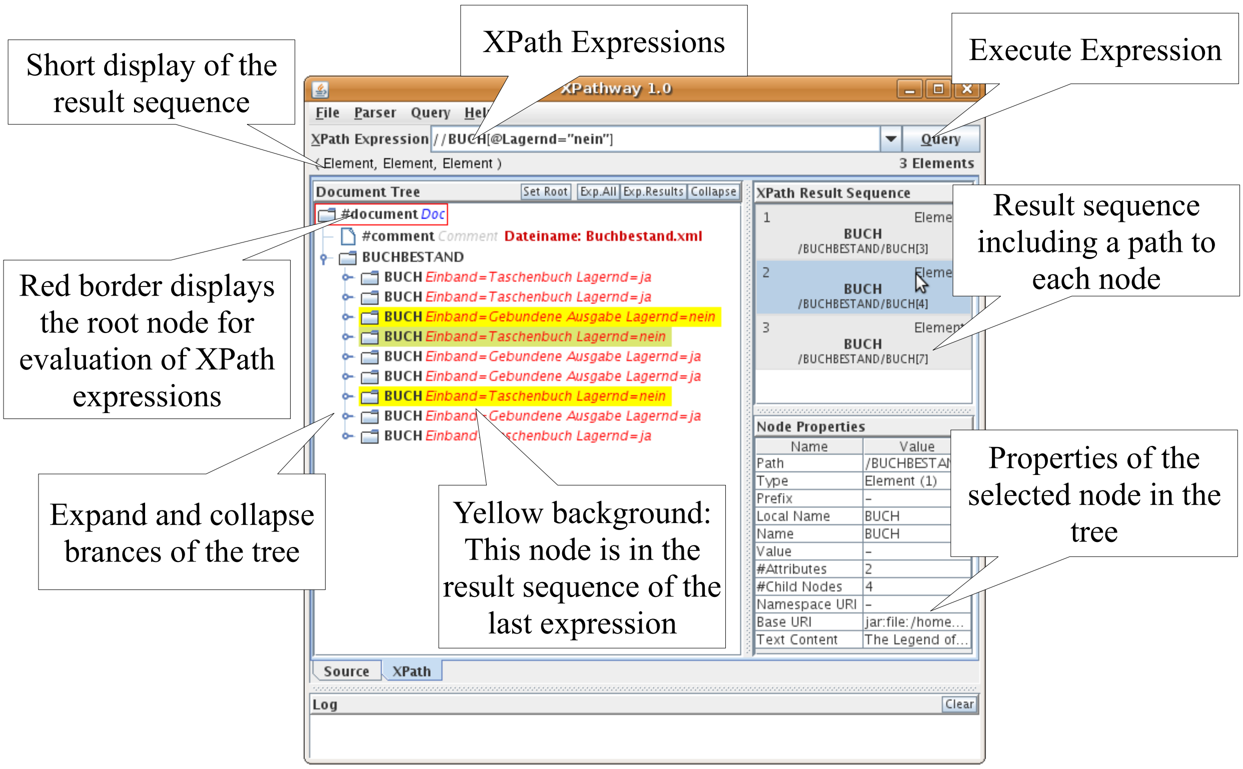
The Log Pane

The log allows you to scroll back errors and other messages. Messages about a missing grammar or missing DOCTYPE can be ignored. They are a byproduct of the validating parser used which is needed to handle namespaces correctly.

A “clear” button cleans up the log.

XPath View

The XPath View is the main working interface. It contains an hierarchical view of the data called “document tree”. Whenever a node in this tree is selected, the “Node Properties” Pane displays in-depth information about that document node. The “Result Sequence” Pane displays the results from XPath queries. Since in XPath 2.0 all Results are sequences not sets, they are numbered.



Short display of the result sequence

XPath Expressions

Execute Expression

Red border displays the root node for evaluation of XPath expressions

Result sequence including a path to each node

Properties of the selected node in the tree

Expand and collapse brances of the tree

Yellow background: This node is in the result sequence of the last expression

XPath Expression: //BUCH[@Lagernd="nein"]

(Element, Element, Element) 3 Elements

#document Doc
#comment Comment Dateiname: Buchbestand.xml
BUCHBESTAND
BUCH Einband=Taschenbuch Lagernd=ja
BUCH Einband=Taschenbuch Lagernd=ja
BUCH Einband=Gebundene Ausgabe Lagernd=nein
BUCH Einband=Taschenbuch Lagernd=nein
BUCH Einband=Gebundene Ausgabe Lagernd=ja
BUCH Einband=Gebundene Ausgabe Lagernd=ja
BUCH Einband=Taschenbuch Lagernd=nein
BUCH Einband=Gebundene Ausgabe Lagernd=ja
BUCH Einband=Taschenbuch Lagernd=ja

1 Element BUCH /BUCHBESTAND/BUCH[3]
2 Element BUCH /BUCHBESTAND/BUCH[4]
3 Element BUCH /BUCHBESTAND/BUCH[7]

Name	Value
Path	/BUCHBESTAN
Type	Element (1)
Prefix	–
Local Name	BUCH
Name	BUCH
Value	–
#Attributes	2
#Child Nodes	4
Namespace URI	–
Base URI	jar:file:/home...
Text Content	The Legend of...

Document tree

The left pane displays the data structure. Each node of the document is represented by a line in the view. Not all levels of the tree are shown at once. You can navigate it using the mouse or the keyboard. The up and down key navigates a line up or down. The left and right key enables you to expand or hide a level.

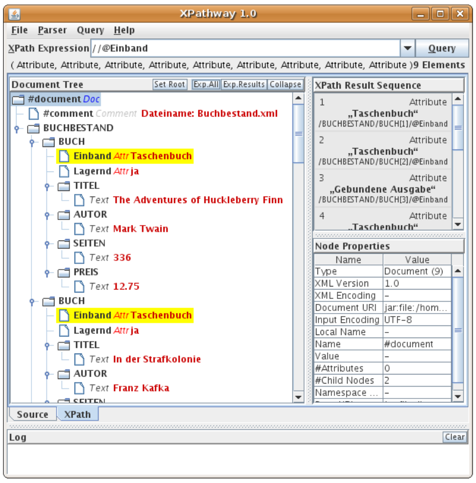
Each node is displayed using its node name and a node type. Where usefull additional information is displayed like attributes or the value of the node.

Additional information to each selected node is displayed in the properties pane on the right side.

A yellow background of a node shows that it is part of the result set. A red border displays that this node is used as starting point (root node) for expression evaluation. See “Executing Expressions” for more information.

The “**Set Root**” button changes the starting point for XPath expressions to the currently selected one.

The “**Expand All**” function shows you all levels and node of



XPath Expression: //@Einband

(Attribute, Attribute, Attribute, Attribute, Attribute, Attribute, Attribute, Attribute, Attribute)9 Elements

#document Doc
#comment Comment Dateiname: Buchbestand.xml
BUCHBESTAND
BUCH
Einband Attr Taschenbuch
Lagernd Attr ja
TITEL
Text The Adventures of Huckleberry Finn
AUTOR
Text Mark Twain
SEITEN
Text 336
PREIS
Text 12.75
BUCH
Einband Attr Taschenbuch
Lagernd Attr ja
TITEL
Text In der Strafkolonie
AUTOR
Text Franz Kafka

1 Attribute „Taschenbuch“ /BUCHBESTAND/BUCH[1]/@Einband
2 Attribute „Taschenbuch“ /BUCHBESTAND/BUCH[2]/@Einband
3 Attribute „Gebundene Ausgabe“ /BUCHBESTAND/BUCH[3]/@Einband
4 Attribute „Taschenbuch“

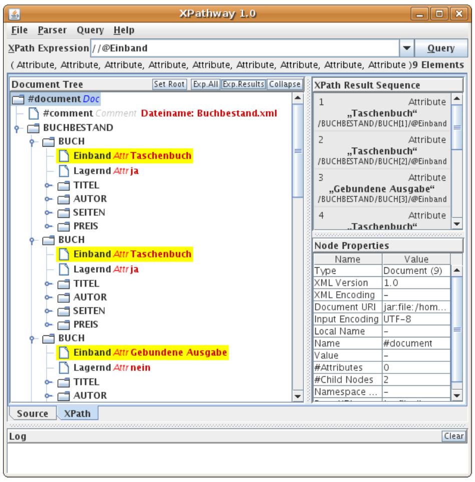
Name	Value
Type	Document (9)
XML Version	1.0
XML Encoding	–
Document URI	jar:file:/hom...
Input Encoding	UTF-8
Local Name	–
Name	#document
Value	–
#Attributes	0
#Child Nodes	2
Namespace ...	–

Illustration 4: Expand all

the tree at once. It gives you the most detailed view about the document.

"**Expand Results**" expands the parts of the document tree that are needed to display all the nodes in the result sequence. Use this, if you like to see the position of all nodes you selected by a XPath Expression.

"**Collapse All**" hides all levels but the first one.



XPath Expression //@Einband

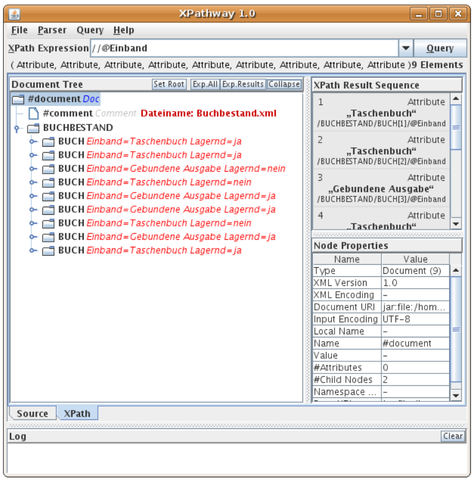
(Attribute, Attribute, Attribute, Attribute, Attribute, Attribute, Attribute, Attribute, Attribute)9 Elements

#document Doc
#comment Comment Dateiname: Buchbestand.xml
BUCHBESTAND
BUCH
Einband Attr Taschenbuch
Lagernd Attr ja
TITEL
AUTOR
SEITEN
PREIS
BUCH
Einband Attr Taschenbuch
Lagernd Attr ja
TITEL
AUTOR
SEITEN
PREIS
BUCH
Einband Attr Gebundene Ausgabe
Lagernd Attr nein
TITEL
AUTOR

1 Attribute „Taschenbuch“ /BUCHBESTAND/BUCH[1]/@Einband
2 Attribute „Taschenbuch“ /BUCHBESTAND/BUCH[2]/@Einband
3 Attribute „Gebundene Ausgabe“ /BUCHBESTAND/BUCH[3]/@Einband
4 Attribute „Taschenbuch“

Name	Value
Type	Document (9)
XML Version	1.0
XML Encoding	–
Document URI	jar:file:/hom...
Input Encoding	UTF-8
Local Name	–
Name	#document
Value	–
#Attributes	0
#Child Nodes	2
Namespace ...	–

Illustration 5: Expand results



XPath Expression //@Einband

(Attribute, Attribute, Attribute, Attribute, Attribute, Attribute, Attribute, Attribute, Attribute)9 Elements

#document Doc
#comment Comment Dateiname: Buchbestand.xml
BUCHBESTAND
BUCH Einband=Taschenbuch Lagernd=ja
BUCH Einband=Taschenbuch Lagernd=ja
BUCH Einband=Gebundene Ausgabe Lagernd=nein
BUCH Einband=Taschenbuch Lagernd=nein
BUCH Einband=Gebundene Ausgabe Lagernd=ja
BUCH Einband=Gebundene Ausgabe Lagernd=ja
BUCH Einband=Taschenbuch Lagernd=nein
BUCH Einband=Gebundene Ausgabe Lagernd=ja
BUCH Einband=Taschenbuch Lagernd=ja

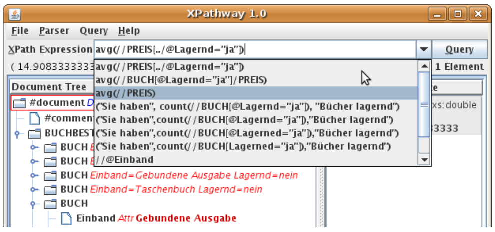
1 Attribute „Taschenbuch“ /BUCHBESTAND/BUCH[1]/@Einband
2 Attribute „Taschenbuch“ /BUCHBESTAND/BUCH[2]/@Einband
3 Attribute „Gebundene Ausgabe“ /BUCHBESTAND/BUCH[3]/@Einband
4 Attribute „Taschenbuch“

Name	Value
Type	Document (9)
XML Version	1.0
XML Encoding	–
Document URI	jar:file:/hom...
Input Encoding	UTF-8
Local Name	–
Name	#document
Value	–
#Attributes	0
#Child Nodes	2
Namespace ...	–

Illustration 6: Collapse All

History

The last 10 successful expressions are collected in the expression history. You can select old expressions by clicking on the down arrow next to the query field



XPath Expression avg(//PREIS[../@Lagernd="ja"])

(14.9083333333

1 Element

avg(//PREIS[../@Lagernd="ja"])
avg(//BUCH[@Lagernd="ja"]/PREIS)
avg(//PREIS)
("Sie haben", count(//BUCH[@Lagernd="ja"]), "Bücher lagernd")
("Sie haben",count(//BUCH[@Lagernd="ja"]),"Bücher lagernd")
("Sie haben",count(//BUCH[@Lagerned="ja"]),"Bücher lagernd")
("Sie haben",count(//BUCH[Lagerned="ja"]),"Bücher lagernd")
//@Einband

xs:double

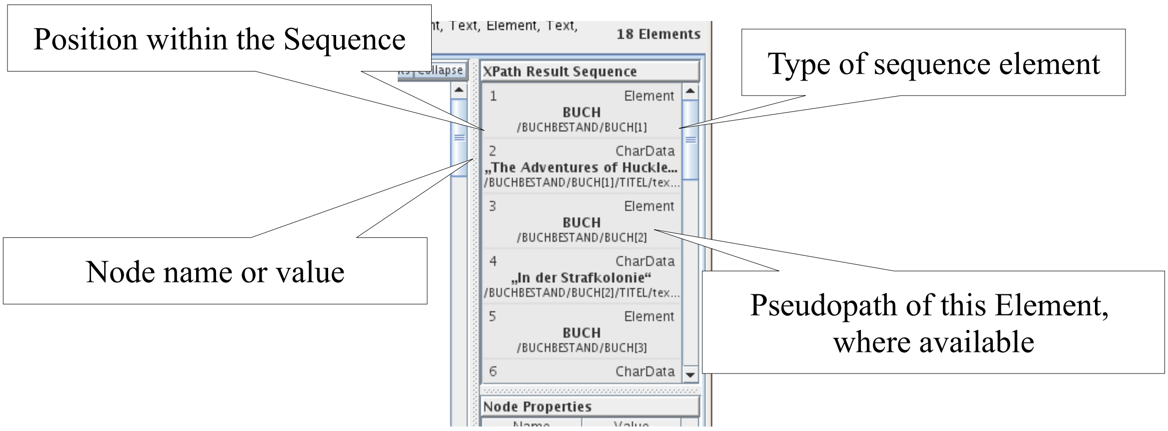
3333

#document
#commen
BUCHBEST
BUCH
BUCH
BUCH Einband=Gebundene Ausgabe Lagernd=nein
BUCH Einband=Taschenbuch Lagernd=nein
BUCH
Einband Attr Gebundene Ausgabe

Illustration 7: History function

Result Pane

The result pane displays the ordered sequence generated by the executed XPath expressions. The default behavior is to show the node name, unless it is a atomic value (string, number, etc) or a text node or similar. If you like to show all result elements as string, you can change that in the "Query" configuration menu.
As soon as you select one of the result elements, the appropriate node in the document tree is selected too.



nt, Text, Element, Text, 18 Elements

1 Element BUCH /BUCHBESTAND/BUCH[1]
2 CharData „The Adventures of Huckle... /BUCHBESTAND/BUCH[1]/TITEL/tex...
3 Element BUCH /BUCHBESTAND/BUCH[2]
4 CharData „In der Strafkolonie“ /BUCHBESTAND/BUCH[2]/TITEL/tex...
5 Element BUCH /BUCHBESTAND/BUCH[3]
6 CharData

Illustration 8: Result Pane

The pseudo path displayed with each element is a simplified XPath like path to give you an impression where to find this node. Its not a valid XPath expression in all cases.

Loading Files

The "File" menu gives you the option to open any local file or to use one of the supplied examples. The "Reopen" function allows you to reread an external XML file should you have changed it with an external editor.

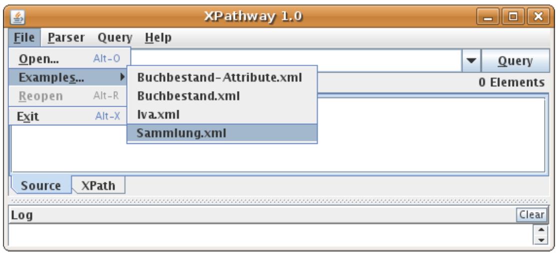


Illustration 9: File Open & Examples

Customizing the Examples

The examples are packed within the .jar file. You can open the jar archive with any archive manager that can handle ZIP-Archives and add, remove or change the files in the examples/ folder. The next time you start XPathway from that .jar file, the contents of that directory is reread and displayed in the "Examples..." menu.

That way you can prepackage XPathway with any examples you like, without recompiling or repackaging the whole application.

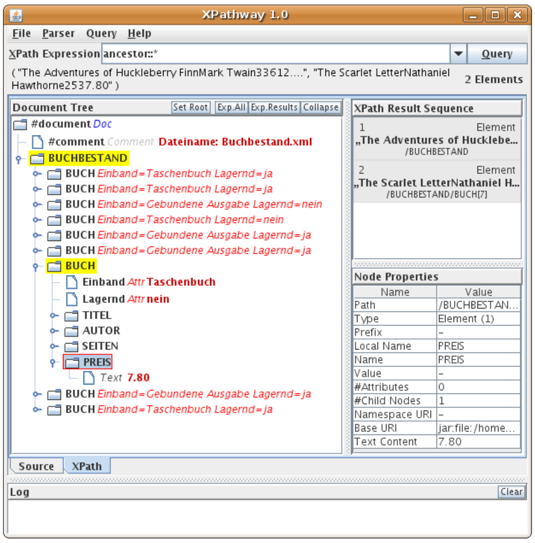
Executing Expressions

You can enter expressions in the upper XPath expression field and hit the enter key or the "Query" button on the top right. In XPath 2.0 all results are shallow sequences.
By default only atomic values and text()-nodes are displayed using their value. All document nodes are displayed using their name. You can change this to textual display for all results using the "Query" configuration menu.

Important notes

Namespaces are supported, but you might encounter unexpected behavior when dealing with default-namespaces. Since default namespaces are without a prefix, you might find it difficult to address them directly. One way to deal with this, is to define also a prefix for that namespace. This works, because during evaluation the namespaces are compared, not the prefixes.[1][2]

You can change the starting point for expressions by selecting a node and hit the "select root" button. This way, you can demonstrate backward faxing axis more easily.



XPath Expression ancestor::*

```
( "The Adventures of Huckleberry FinnMark Twain33612....", "The Scarlet LetterNathaniel
Hawthorne2537.80" )                                                            2 Elements
```

```
#document Doc
  #comment Comment Dateiname: Buchbestand.xml
  BUCHBESTAND
    BUCH Einband=Taschenbuch Lagernd=ja
    BUCH Einband=Taschenbuch Lagernd=ja
    BUCH Einband=Gebundene Ausgabe Lagernd=nein
    BUCH Einband=Taschenbuch Lagernd=nein
    BUCH Einband=Gebundene Ausgabe Lagernd=ja
    BUCH Einband=Gebundene Ausgabe Lagernd=ja
    BUCH
      Einband Attr Taschenbuch
      Lagernd Attr nein
      TITEL
      AUTOR
      SEITEN
      PREIS
        Text 7.80
    BUCH Einband=Gebundene Ausgabe Lagernd=ja
    BUCH Einband=Taschenbuch Lagernd=ja
```

```
1                          Element
„The Adventures of Hucklebe...
          /BUCHBESTAND
2                          Element
„The Scarlet LetterNathaniel H...
      /BUCHBESTAND/BUCH[7]
```

Name	Value
Path	/BUCHBESTAN...
Type	Element (1)
Prefix	-
Local Name	PREIS
Name	PREIS
Value	-
#Attributes	0
#Child Nodes	1
Namespace URI	-
Base URI	jar:file:/home...
Text Content	7.80

Illustration 10: Using a diffrent evaluation root, for demonstrating the ancestor:: axis

Parser Configuration

All changes in the parser configuration take effect with the next load of a file. You can also use the "Reload" function to load and parse the last opened file again.

The "**Namespace Aware**" option ensures all namespace information is read and incorporated into

---

[1] http://www.edankert.com/defaultnamespaces.html

[2] http://www.faqts.com/knowledge_base/view.phtml/aid/34263/fid/1400

the document tree. This only works when using a validating parser.

“**Validation Parser**” validates all XML Data while parsing. This is needed for namespace support but throws errors when documents without grammar or without matching DOCTYPEs are loaded. You can ignore that messages.

“**Ignore Element Whitespace**” instructs the parser to eliminate all the ignorable whitespace inside elements.

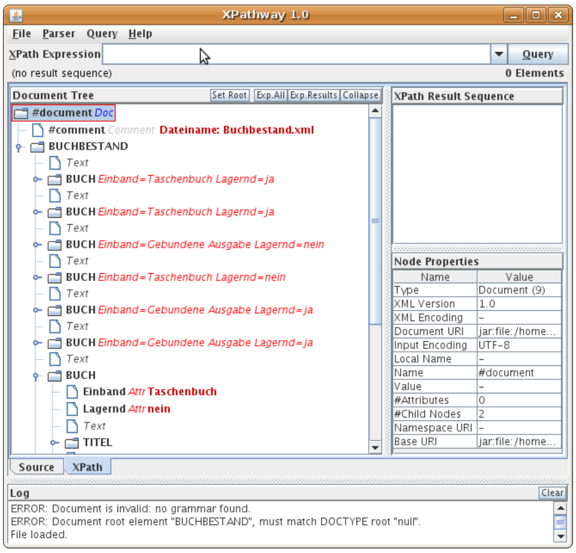
“**Remove Interim Whitespace**” removes whitespace that is often used between XML elements to make a document more readable. Its a good idea to leave this option on all the time.

“**Coalescing**” instructs the parser to convert all CDATA-Sections into Strings.

“**Xinclude Aware**” allows the usage of Include statements in the XML Source file.

“**Remove Comments**” will remove all <!-- comments --> before displaying the document tree.

The optoion “**Remove Proccess Instructions**” removes PI nodes after loading.



```
#document Doc
  #comment Comment Dateiname: Buchbestand.xml
  BUCHBESTAND
    Text
    BUCH Einband=Taschenbuch Lagernd=ja
    Text
    BUCH Einband=Taschenbuch Lagernd=ja
    Text
    BUCH Einband=Gebundene Ausgabe Lagernd=nein
    Text
    BUCH Einband=Taschenbuch Lagernd=nein
    Text
    BUCH Einband=Gebundene Ausgabe Lagernd=ja
    Text
    BUCH Einband=Gebundene Ausgabe Lagernd=ja
    Text
    BUCH
      Einband Attr Taschenbuch
      Lagernd Attr nein
      Text
      TITEL
```

Name	Value
Type	Document (9)
XML Version	1.0
XML Encoding	–
Document URI	jar:file:/home...
Input Encoding	UTF-8
Local Name	–
Name	#document
Value	–
#Attributes	0
#Child Nodes	2
Namespace URI	–
Base URI	jar:file:/home...

```
ERROR: Document is invalid: no grammar found.
ERROR: Document root element "BUCHBESTAND", must match DOCTYPE root "null".
File loaded.
```

Illustration 11: A document loaded without whitespace removal: between all elements, text nodes contain whitespace

Query Configuration

Changes in this configuration take effect the next time you execute a new query.

“**Autoexpand Results**” ensures that all nodes in the result set are shown in the document tree. It is equivalent to pressing the “Expand Results” button after each query.

The “**XPath 1.0 compatibility Mode**” option enables the SAXON parameter of the same name.

“**Always show textual representation**” shows you the string version of each element in the result sequence. This is the preferred mode, when testing XSLT embedded XPath Expressions.

Source Code

The source code is a Eclipse 3.2[3] project using JDK 1.5. For convenience you might also like to install the “Visual Editor” and the “fat-jar” extension.

Visual Editor[4] gives you a comfortable way of designing a GUI. It is not needed to compile the source, but very useful.

Fat-jar is an easy way for generating .jar files for your application which relies on other jar archives.

---

[3] Visual Editor is not available for Eclipse 3.3 in October 2008

[4] http://www.eclipse.org/vep/WebContent/main.php

The extension repacks all other jar archives into one target .jar archive.

Saxon-B[5] 9.1.0.2 (the open source variant) is used in this application.

---

[5] http://saxon.sourceforge.net/