XPath

Sang Shin
www.JPassion.com
“Learn with Passion!”

Agenda

- XPath Overview
- Node
- Node set
- Location path
- Wild cards
- Multiple matches
- Compound location paths
- Predicates
- Functions

XPath Overview

XPath Overview

- Expression language for referencing particular parts of XML documents
- Examples that can be expressed with XPath
  - First *person* element
  - Seventh child element of the third *person* element
  - *ID* attribute of the first *person* element whose content is the string "*JavaPassion class*"
  - All *xml-stylesheet* processing instructions

XPath Expression Criteria

- Position
- Relative position
- Type
- Content
- Numbers
- Strings
- Booleans
- Functions

XPath Usages

- XSLT Stylesheet
  - To *match* and *select* elements and attributes of input XML document
- XPointer
  - To identify the particular point in or part of an XML document that an XLink links to

Lab:

Exercise 1: Install XPath Checker
Exercise 7: Display XPath of a Selected Element
4345_ws_xml_xpath.zip

Node Types

XPath Node

- XML document is a tree of nodes
- 7 kinds of nodes
  - The root node
  - Element nodes
  - Text nodes
  - Attribute nodes
  - Comment nodes
  - Processing instruction nodes
  - Namespace nodes

XPath Node

- Root node
  - Is not the same as root element
  - Contains entire document including
    - root element
    - processing instructions
    - comments

Expression Result Datatypes

- XPath expression evaluates to one of four types
  - Node set
  - Boolean
  - Number
  - String

Node Set

Node Set

- Collection of zero or more nodes from an XML document
- Returned from location path expressions
- Things that cannot be in node set (because XPath operates on an XML document after these items are resolved)
  - CDATA sections
  - Entity references
  - Document type declaration

Example XML document: people.xml (We will use this to learn XPath)

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="people.xsl"?>
<!DOCTYPE people [
 <!ATTLIST homepage xlink:type CDATA #FIXED "simple"
                  xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink">
 <!ATTLIST person id ID #IMPLIED>
]>
<people>
  <person born="1912" died="1954" id="p342">
    <name>
      <first_name>Alan</first_name>
      <last_name>Turing</last_name>
    </name>
    <!-- Did the word computer scientist exist in Turing's day? -->
    <profession>computer scientist</profession>
    <profession>mathematician</profession>
    <profession>cryptographer</profession>
    <homepage xlink:href="http://www.turing.org.uk/"/>
  </person>

  <person born="1918" died="1988" id="p4567">
    <name>
      <first_name>Richard</first_name>
      <middle_initial>&#x4D;</middle_initial>
      <last_name>Feynman</last_name>
    </name>
    <profession>physicist</profession>
    <hobby>Playing the bongoes</hobby>
  </person>
</people>
```

Location Path

Location Path

- A Node set is returned by location path expression
- A location path is made of location steps
- A location step contains an axis and a node test separated by double colon
  - axis::node-test
- A location step
  - abbreviated form - axis is assumed (focus of this presentation)
  - unabbreviated form - axis is specified

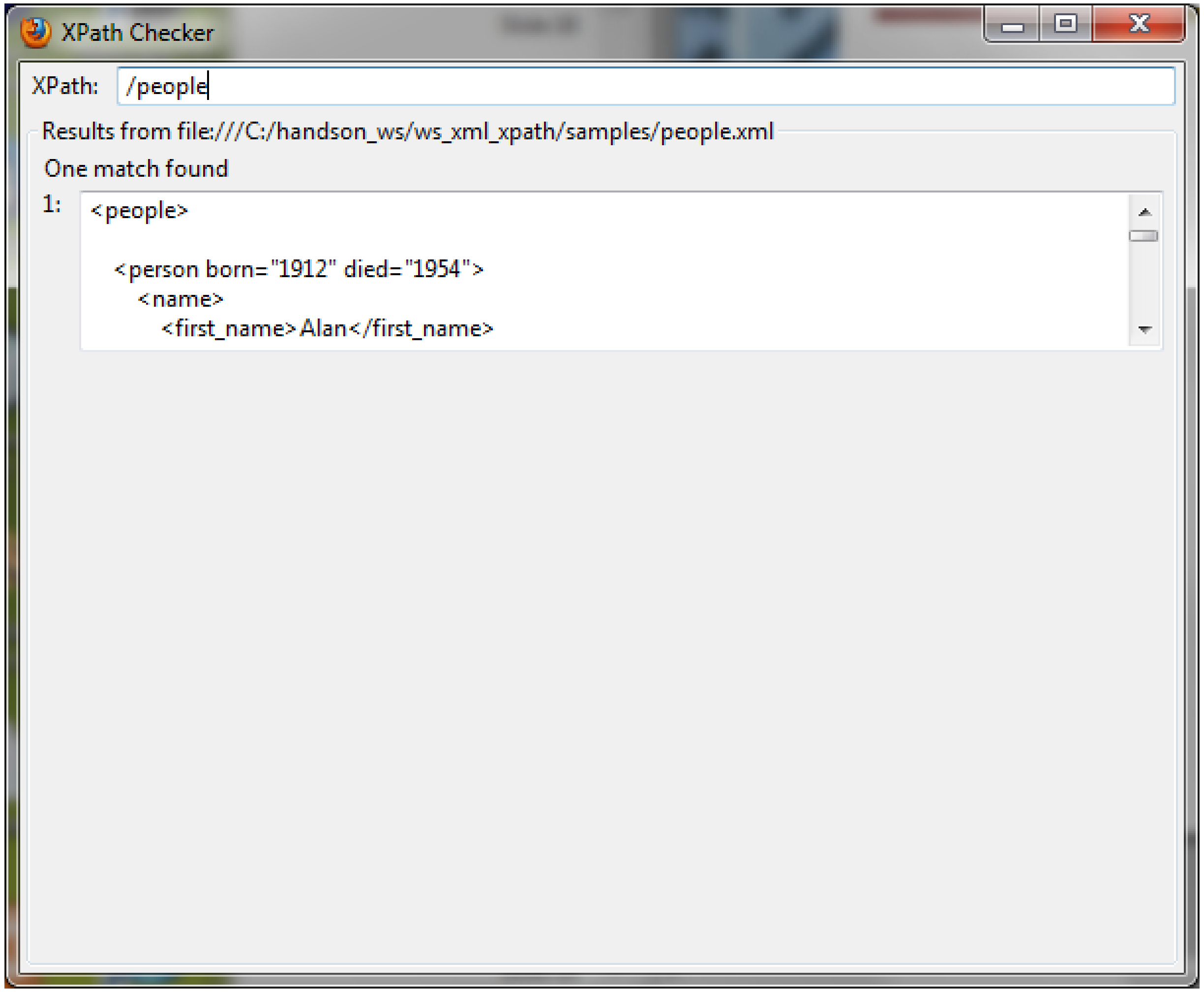
Location Path can include

- Root
- Element
- Attribute
- comment(), text(), processing-instruction()
- Wild cards
- Multiple matches with “|”
- Compound location paths

Root Location Path

- Selects document’s root node
- Represented by “/”
- Absolute location regardless of what the context node is

/people



XPath Checker

XPath: /people

Results from file:///C:/handson_ws/ws_xml_xpath/samples/people.xml

One match found

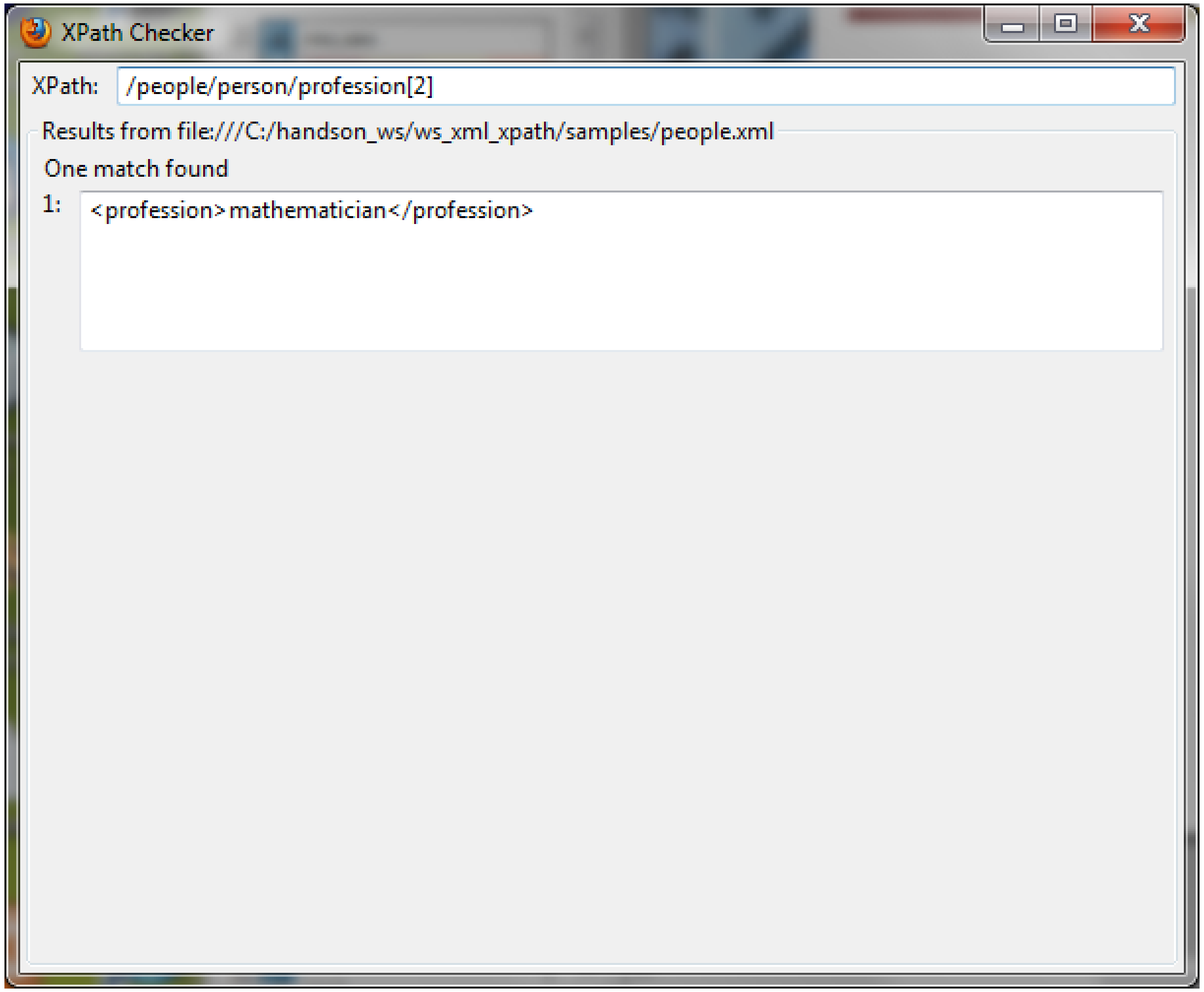
```
1:  <people>

      <person born="1912" died="1954">
        <name>
          <first_name>Alan</first_name>
```

Get all immediate child “<people>” nodes under the root /

Child Element Location Step

- Expression is child element name
- Selects all child elements with the specified name of the context node
- Context node
  - in XSLT
    - Specified in *match* attribute of *xsl:template* element
  - in Xpointer
    - Other means of determining context node are provided

/people/person/profession[2]



XPath: /people/person/profession[2]

Results from file:///C:/handson_ws/ws_xml_xpath/samples/people.xml

One match found

1: <profession>mathematician</profession>

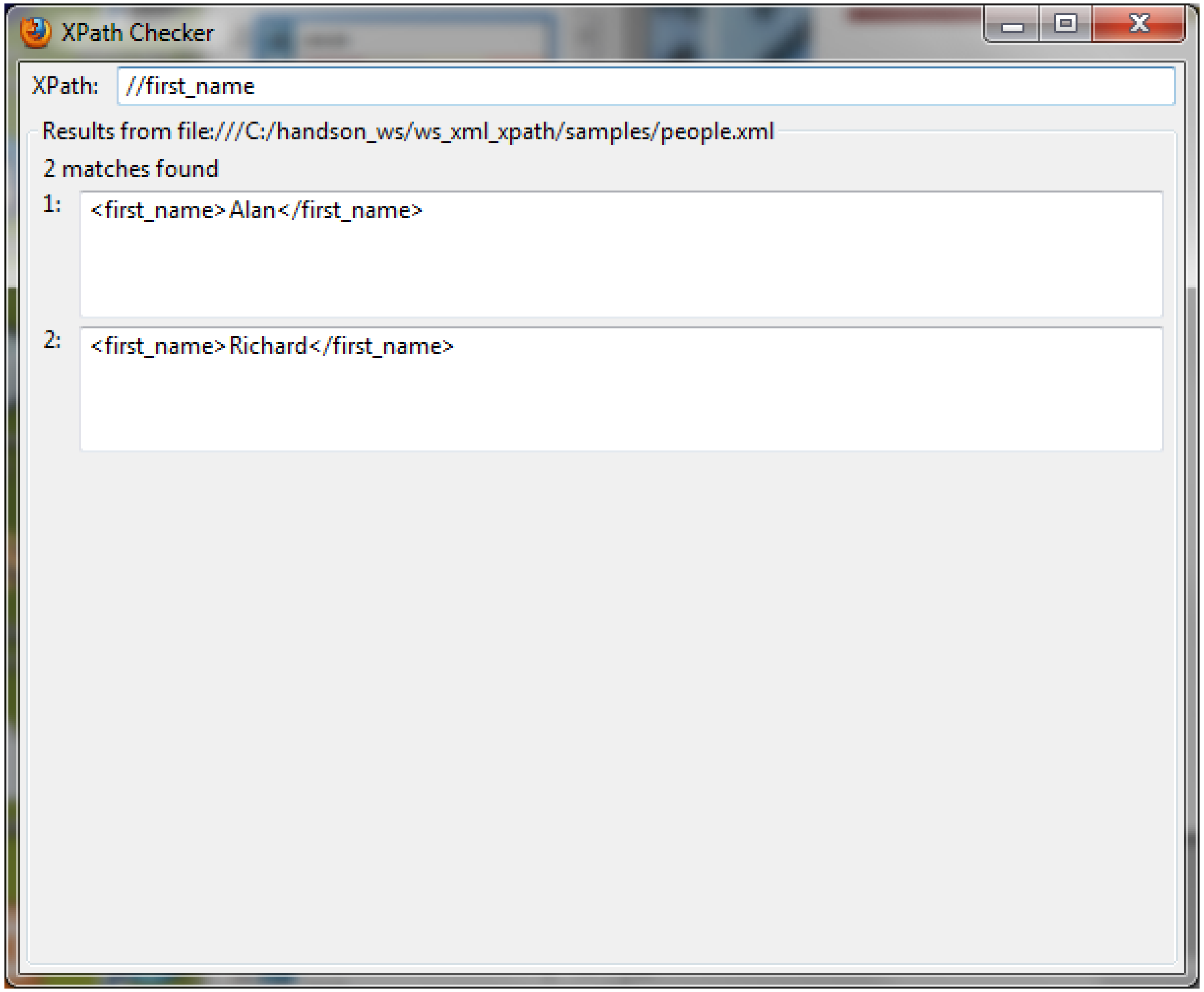
Attribute Location Steps

- Expression: @attribute-name

Other Location Steps

- text node
  - *text()*
  - select all immediate text nodes of context node
- processing-instruction node
  - *processing-instruction()*
- comment node
  - *comment()*
- //xyz
  - All xyz nodes under the root

//first_name



XPath: //first_name

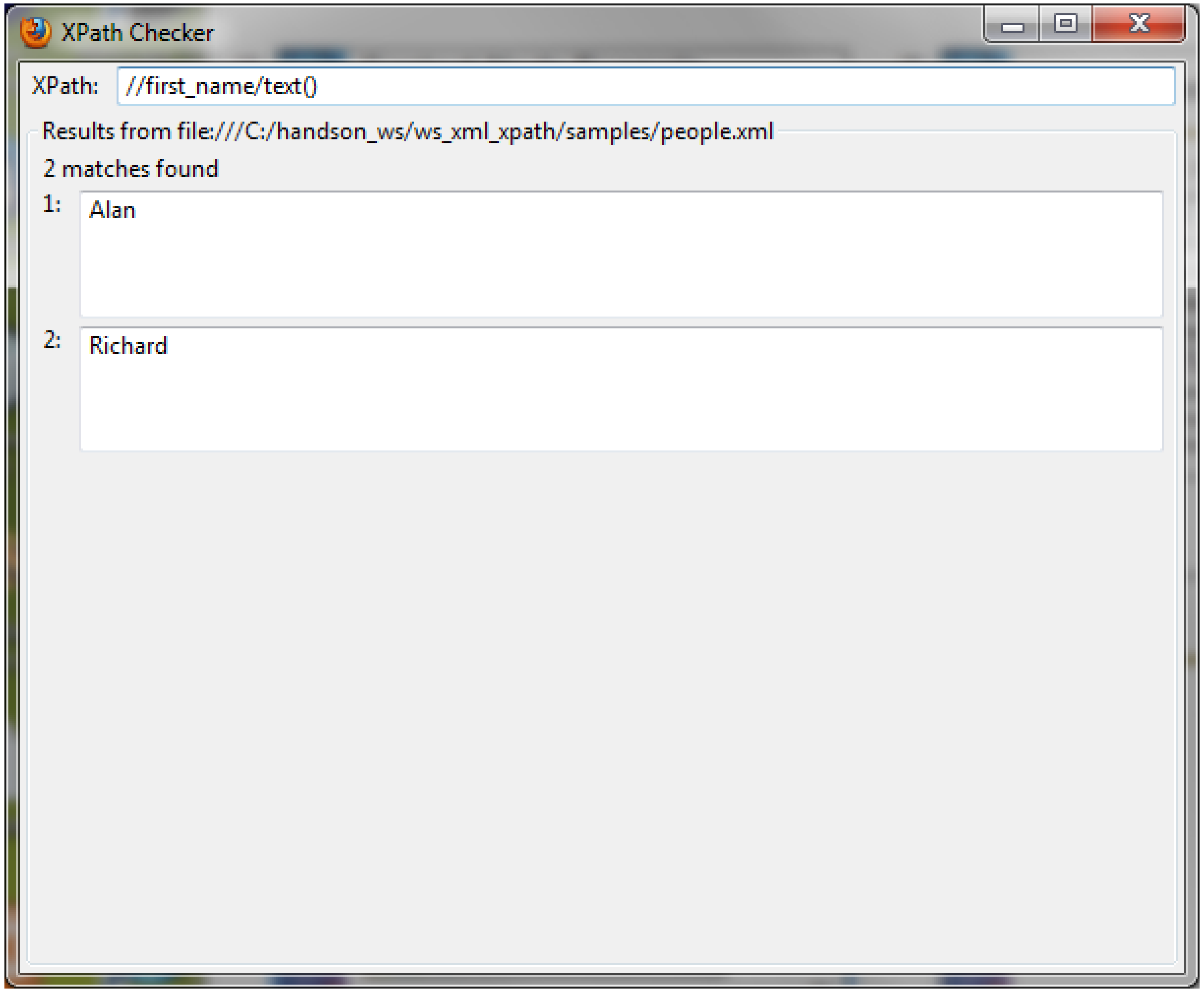
Results from file:///C:/handson_ws/ws_xml_xpath/samples/people.xml

2 matches found

1: <first_name>Alan</first_name>

2: <first_name>Richard</first_name>

//first_name/text()



XPath: //first_name/text()

Results from file:///C:/handson_ws/ws_xml_xpath/samples/people.xml

2 matches found

1: Alan

2: Richard

comment()

- Replace each comment with the text

```
<xsl:template match=“comment()”>
   <i>Comment deleted</I>
<xsl:template>
```

Wild Cards

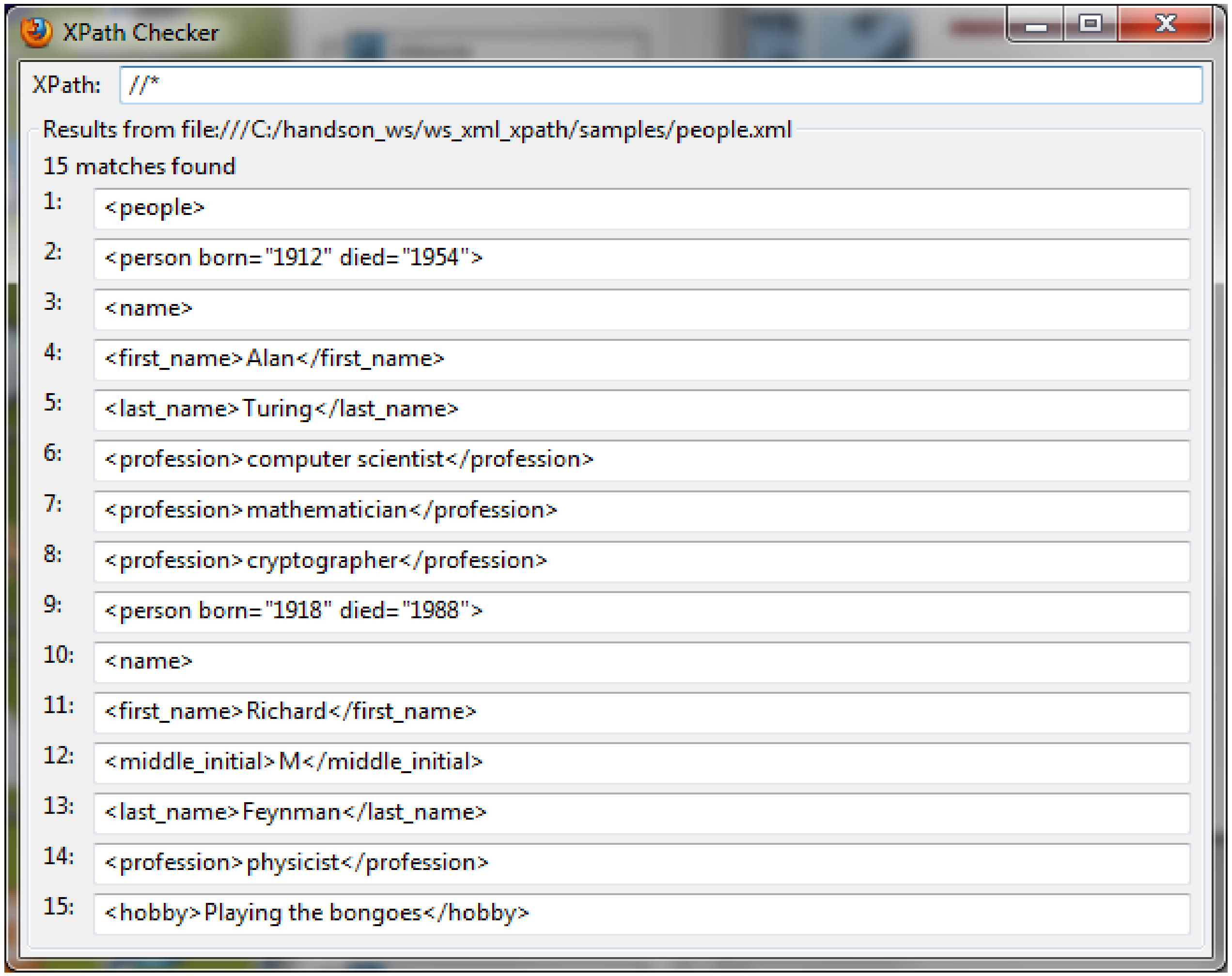
Wild Cards

- Match different element and node types at the same time
- Three wild cards
  - node()
  - *
  - @*

Wild Cards

- node()
  - Matches all nodes including element, text, attribute, processing instruction, namespace, and comment nodes
- Expression: *
  - Matches any element node regardless of type
  - Does not match attributes, text nodes, comments, processing instruction nodes
- @*
  - Matches all attribute nodes

//* - all element nodes



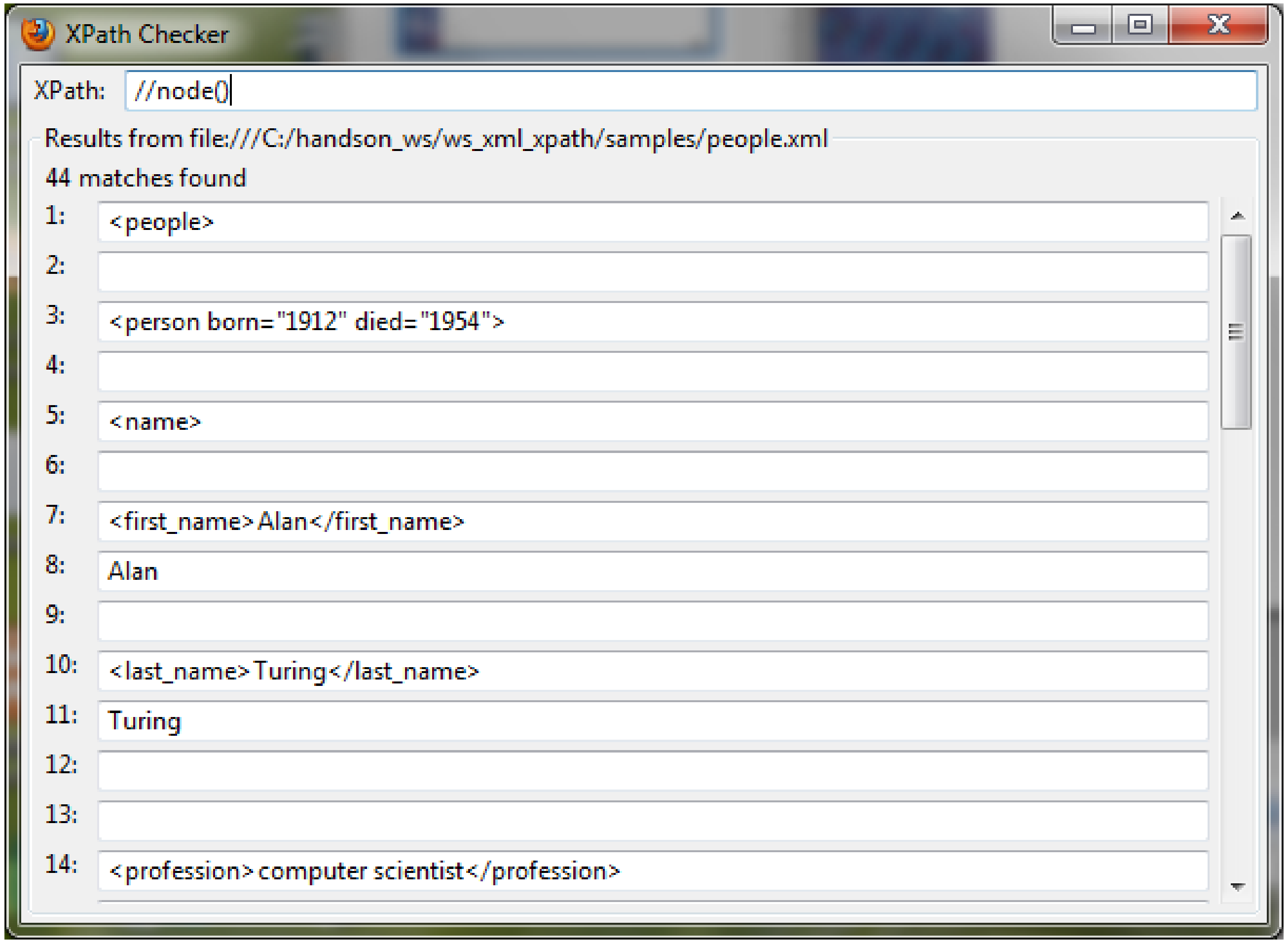
XPath: //*

Results from file:///C:/handson_ws/ws_xml_xpath/samples/people.xml

15 matches found

```
1:  <people>
2:  <person born="1912" died="1954">
3:  <name>
4:  <first_name>Alan</first_name>
5:  <last_name>Turing</last_name>
6:  <profession>computer scientist</profession>
7:  <profession>mathematician</profession>
8:  <profession>cryptographer</profession>
9:  <person born="1918" died="1988">
10: <name>
11: <first_name>Richard</first_name>
12: <middle_initial>M</middle_initial>
13: <last_name>Feynman</last_name>
14: <profession>physicist</profession>
15: <hobby>Playing the bongoes</hobby>
```

//node()   - all nodes



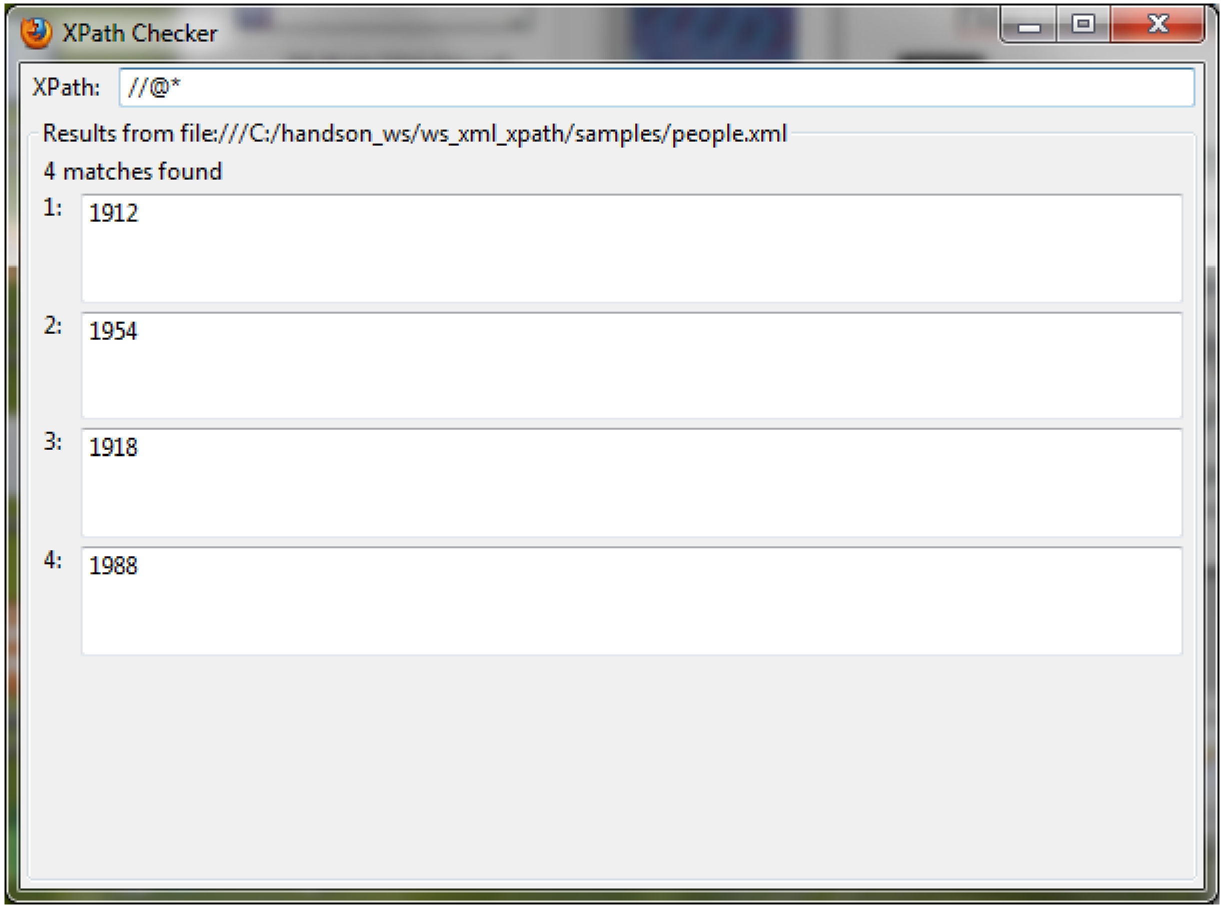
XPath: //node()

Results from file:///C:/handson_ws/ws_xml_xpath/samples/people.xml

44 matches found

```
1:  <people>
2:  
3:  <person born="1912" died="1954">
4:  
5:  <name>
6:  
7:  <first_name>Alan</first_name>
8:  Alan
9:  
10: <last_name>Turing</last_name>
11: Turing
12: 
13: 
14: <profession>computer scientist</profession>
```

//@* - all attribute nodes



XPath: //@*

Results from file:///C:/handson_ws/ws_xml_xpath/samples/people.xml

4 matches found

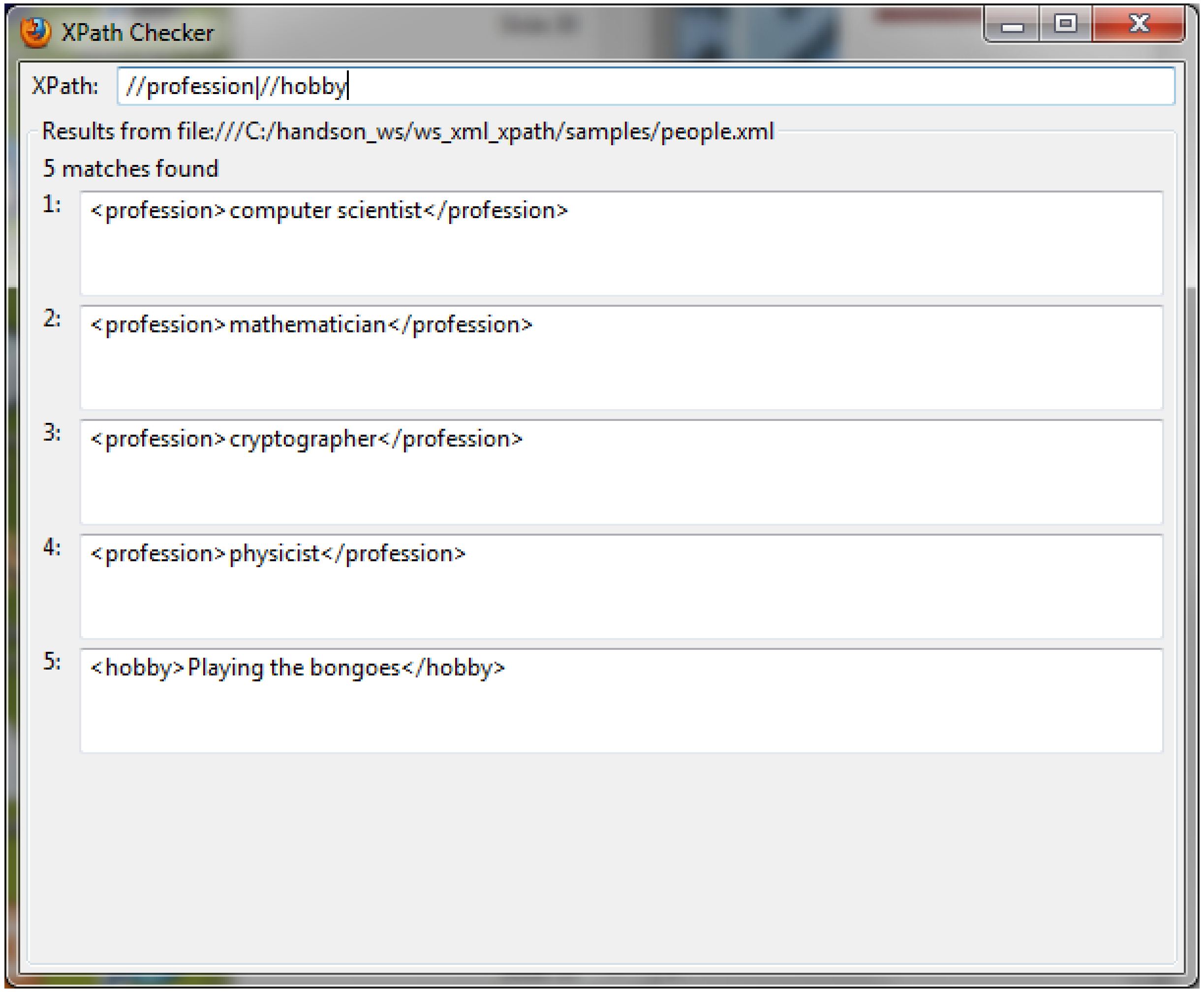
```
1: 1912
2: 1954
3: 1918
4: 1988
```

Multiple Matches

Multiple Matches with “|”

- OR operation
- Examples
  - *profession|hobby*
  - *first_name|last_name|profession|hobby*
  - *@id|@xlink:type*
  - *|@*

//profession|//hobby



XPath: //profession|//hobby

Results from file:///C:/handson_ws/ws_xml_xpath/samples/people.xml

5 matches found

1: <profession>computer scientist</profession>

2: <profession>mathematician</profession>

3: <profession>cryptographer</profession>

4: <profession>physicist</profession>

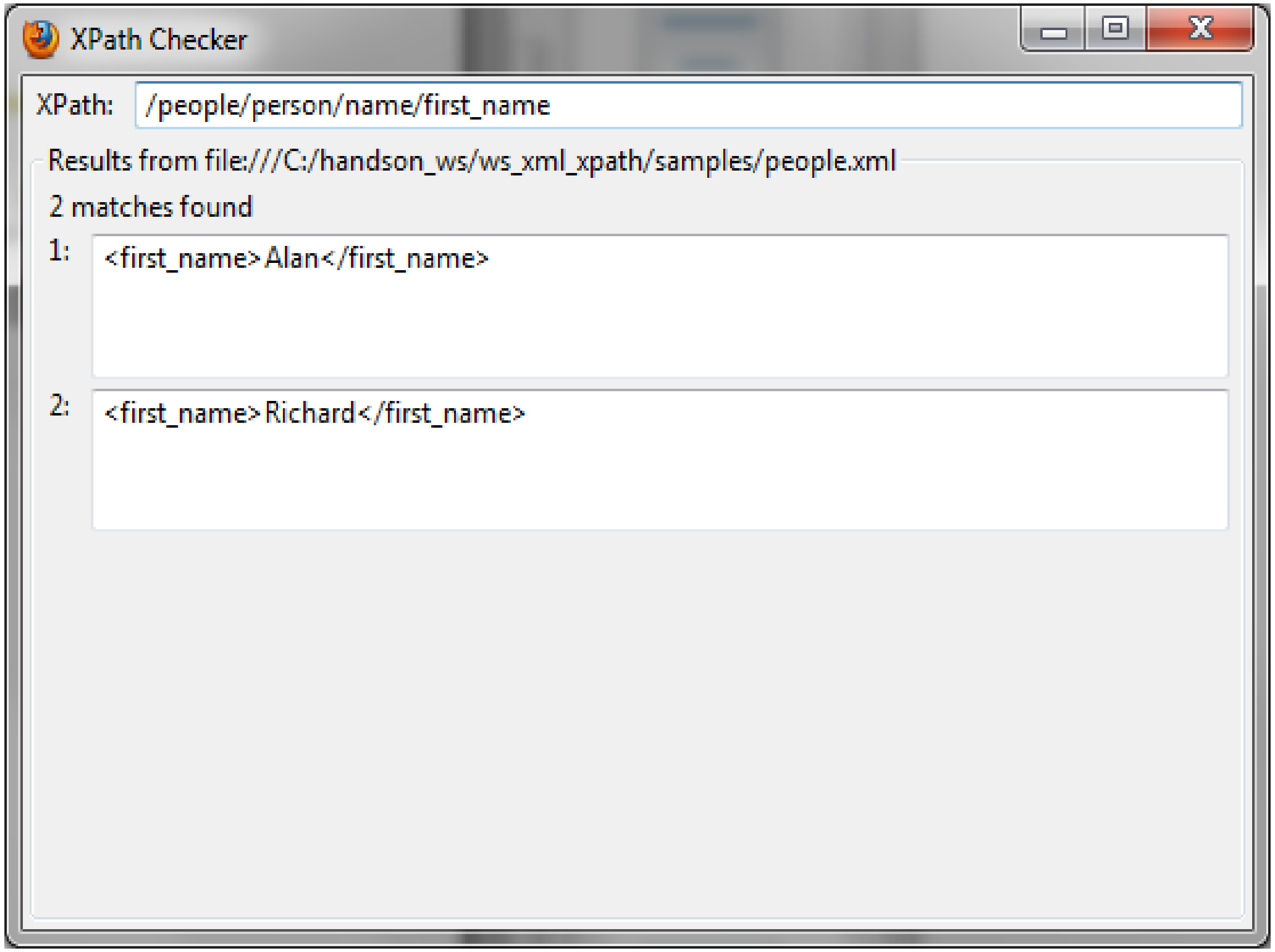
5: <hobby>Playing the bongoes</hobby>

Compound Location Paths

Compound Location Paths

- Combine single location steps with / (forward slash)
- Move down the hierarchy from the matched node to other nodes
- “.” (period) refers to current node
- “..” (double period) refers to parent node
- “//” refers to descendants of the context node

/people/person/name/first_name



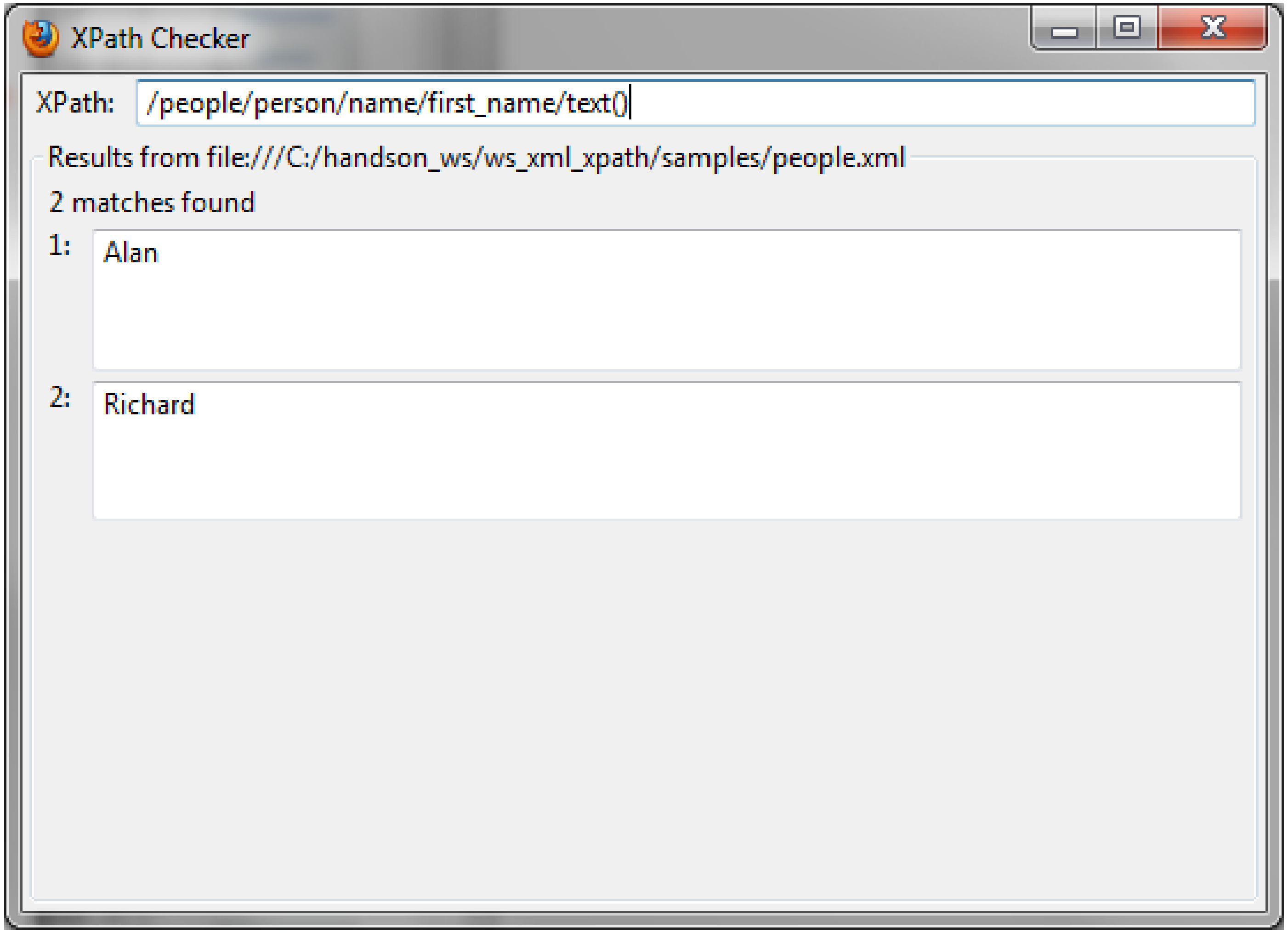
XPath: /people/person/name/first_name

Results from file:///C:/handson_ws/ws_xml_xpath/samples/people.xml

2 matches found

```
1: <first_name>Alan</first_name>
2: <first_name>Richard</first_name>
```

/people/person/name/first_name/text()



XPath: /people/person/name/first_name/text()

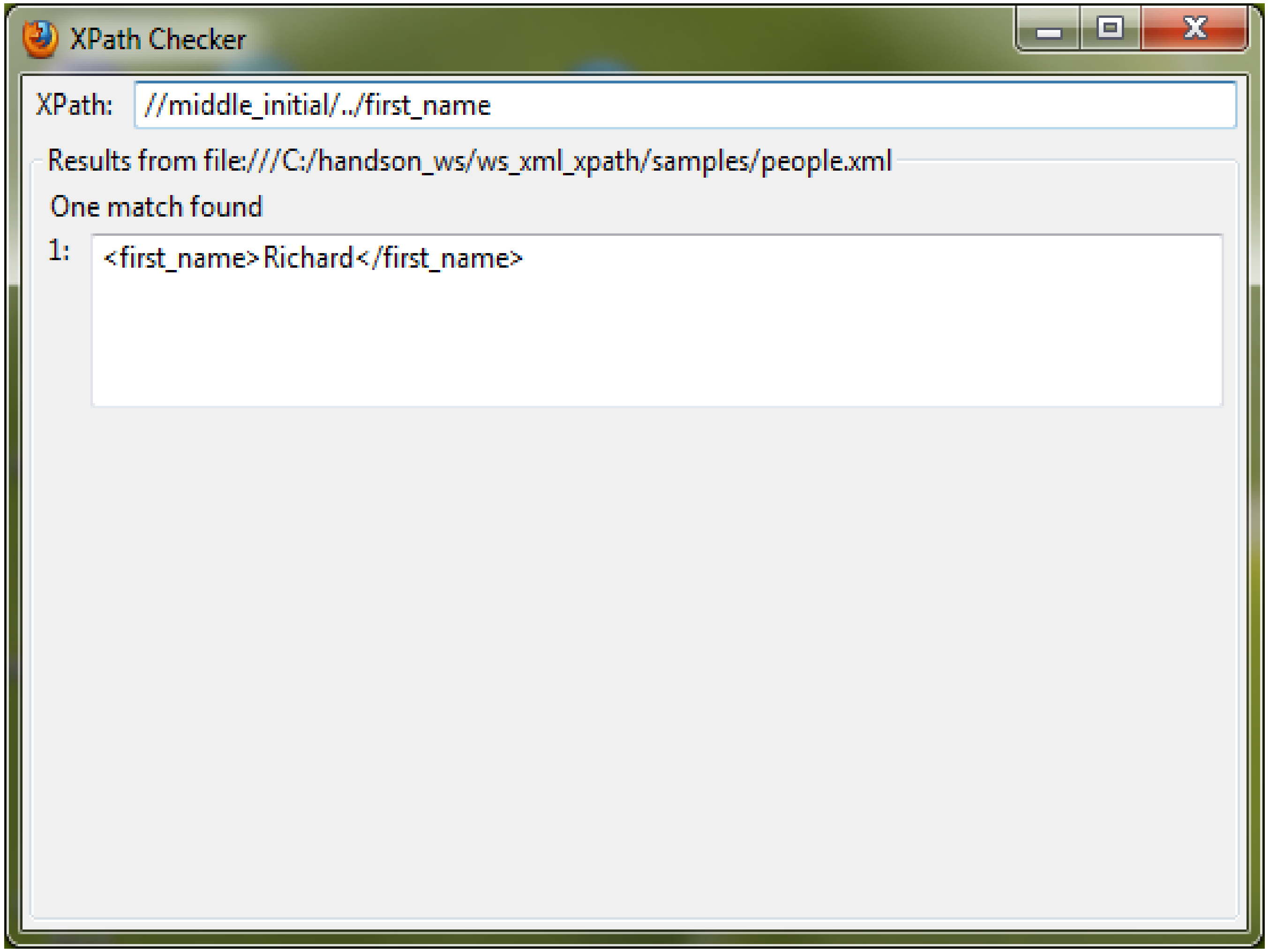
Results from file:///C:/handson_ws/ws_xml_xpath/samples/people.xml

2 matches found

1: Alan

2: Richard

//middle_initial/../first_name



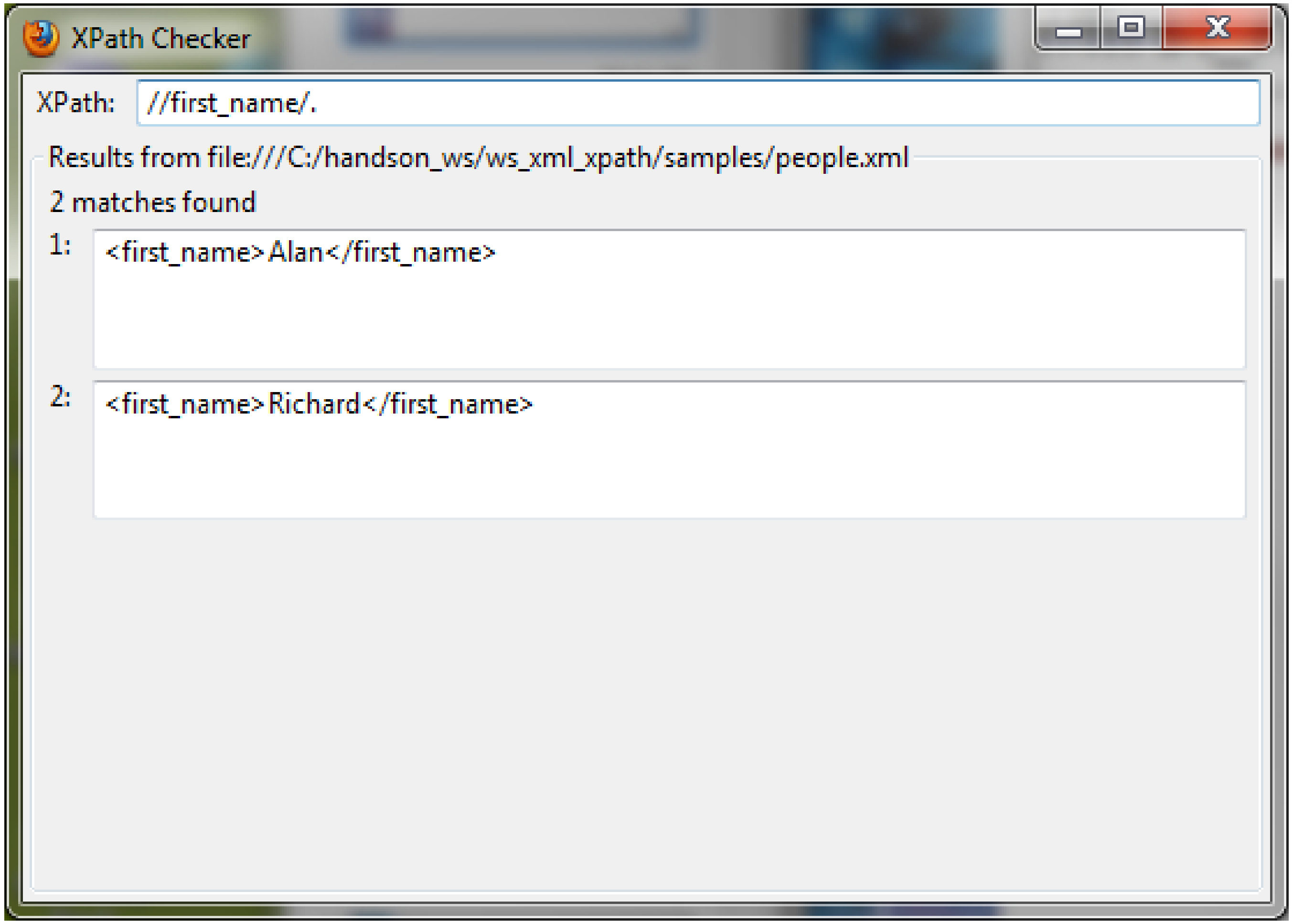
XPath: //middle_initial/../first_name

Results from file:///C:/handson_ws/ws_xml_xpath/samples/people.xml

One match found

```
1: <first_name>Richard</first_name>
```

//first_name/.

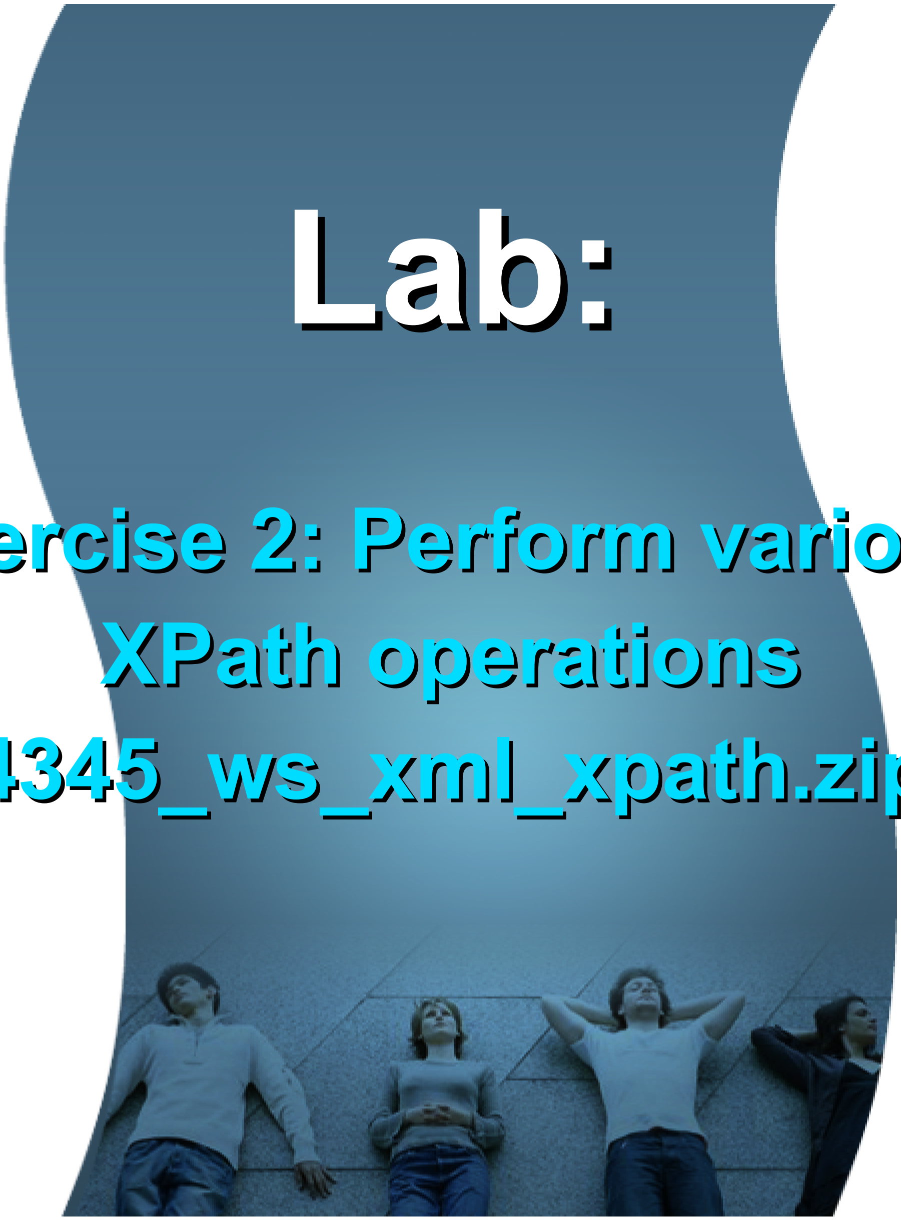


XPath: //first_name/.

Results from file:///C:/handson_ws/ws_xml_xpath/samples/people.xml

2 matches found

```
1: <first_name>Alan</first_name>

2: <first_name>Richard</first_name>
```



Lab:

Exercise 2: Perform various XPath operations

4345_ws_xml_xpath.zip

Predicates

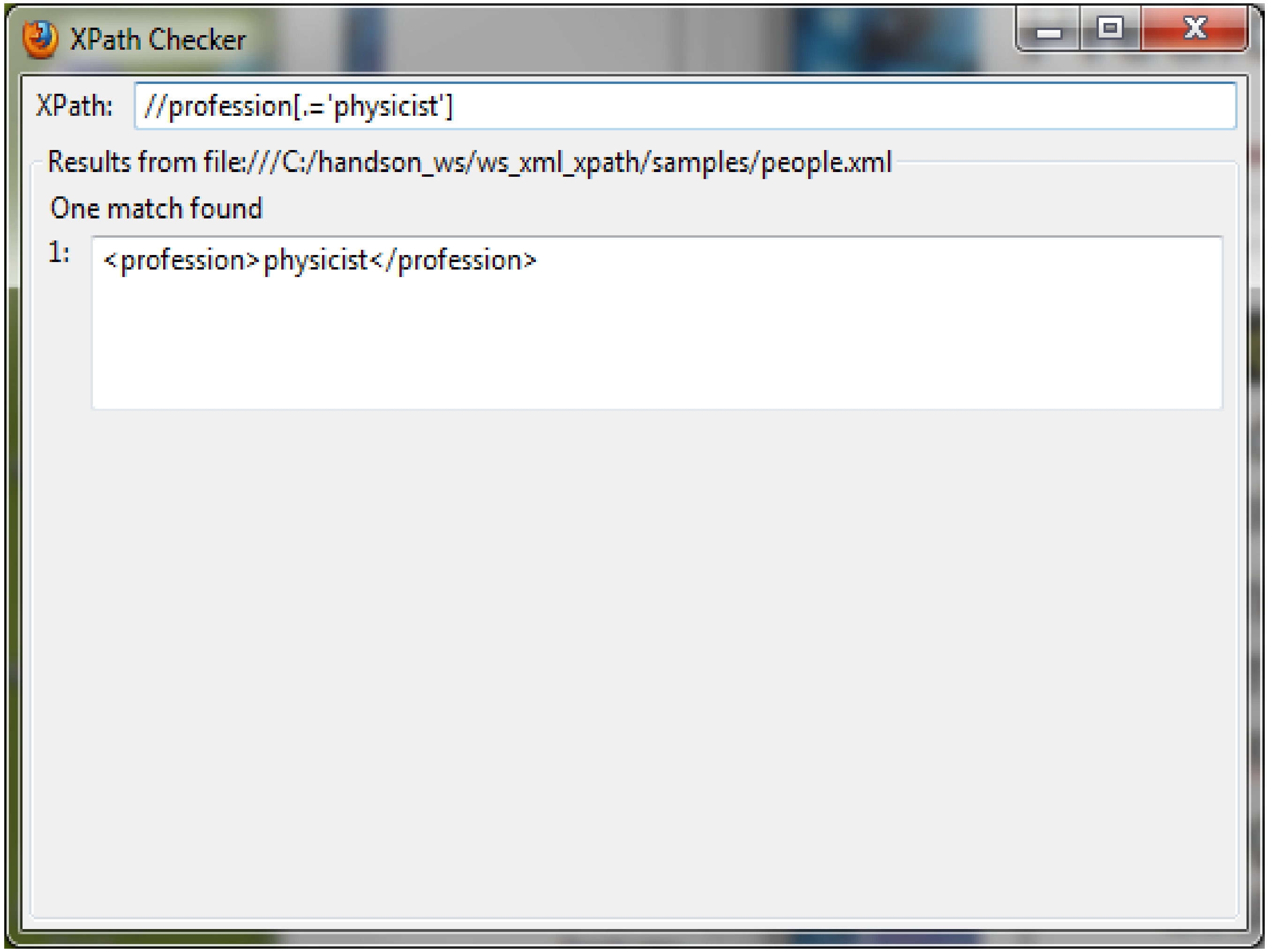
Predicates

- Used to select a subset from a node set
- Can be applied to each location step in a location path
- Boolean expression applied to each node in the node set

Predicates – value comparison

- Example 1
  - > //profession[.='physicist']
  - > Find all *profession* elements whose value is *physicist*
  - > . (Period) stands for string value of the current node (same as the value returned by *xsl:value-of*)
  - > Both single quote and double quote can be used

//profession[.='physicist']



XPath: //profession[.='physicist']

Results from file:///C:/handson_ws/ws_xml_xpath/samples/people.xml

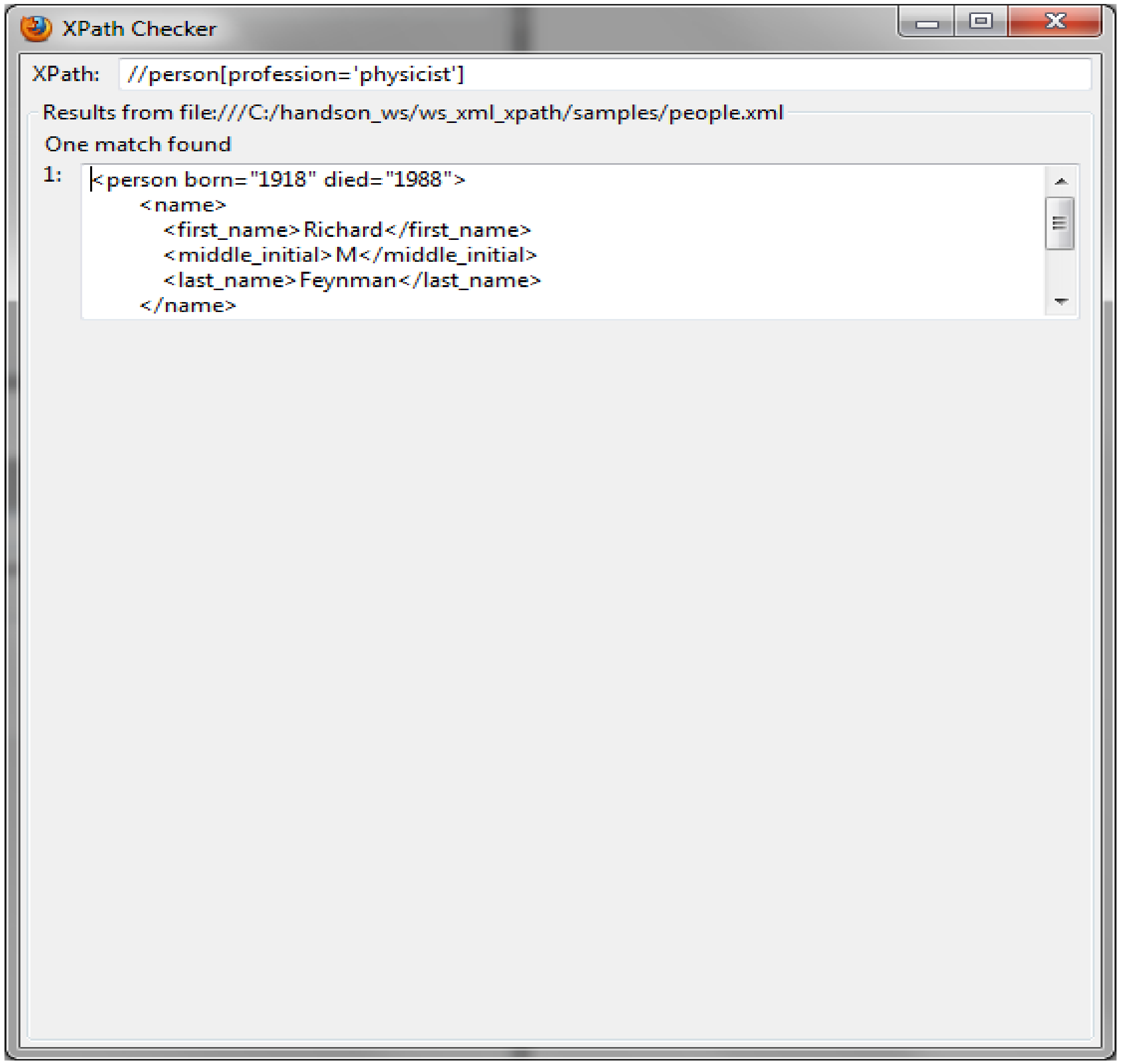
One match found

1: <profession>physicist</profession>

Predicates

- Example 2
  - *//person[profession="physicist"]*
  - Find person elements that have profession child element with the value "physicist"
- Example 3
  - *//person[@id="p4567"]*
  - Find a person element whose ID attributes is *p4567*

//person[profession="physicist"]



XPath: //person[profession='physicist']

Results from file:///C:/handson_ws/ws_xml_xpath/samples/people.xml

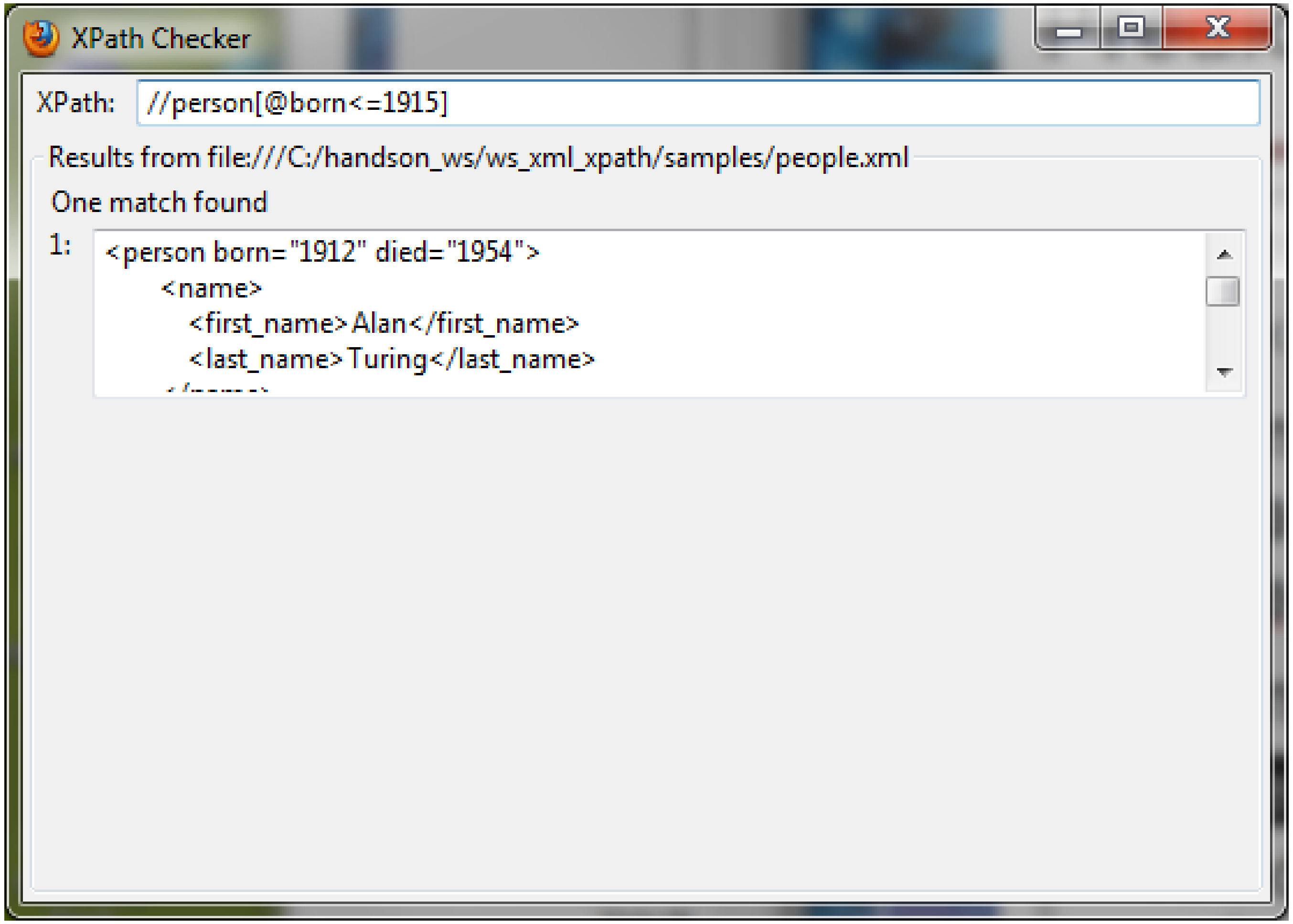
One match found

1:
```
<person born="1918" died="1988">
        <name>
           <first_name>Richard</first_name>
           <middle_initial>M</middle_initial>
           <last_name>Feynman</last_name>
        </name>
```

Predicates – relational operators

- Supports all relational operators
  - =, <, >, <=, >=, !=
- Example 4
  - //person[@born<=1915]
  - Find *person* elements with *born* attribute's numeric value is less than or equal to 1915

//person[@born<=1915]



XPath: //person[@born<=1915]

Results from file:///C:/handson_ws/ws_xml_xpath/samples/people.xml

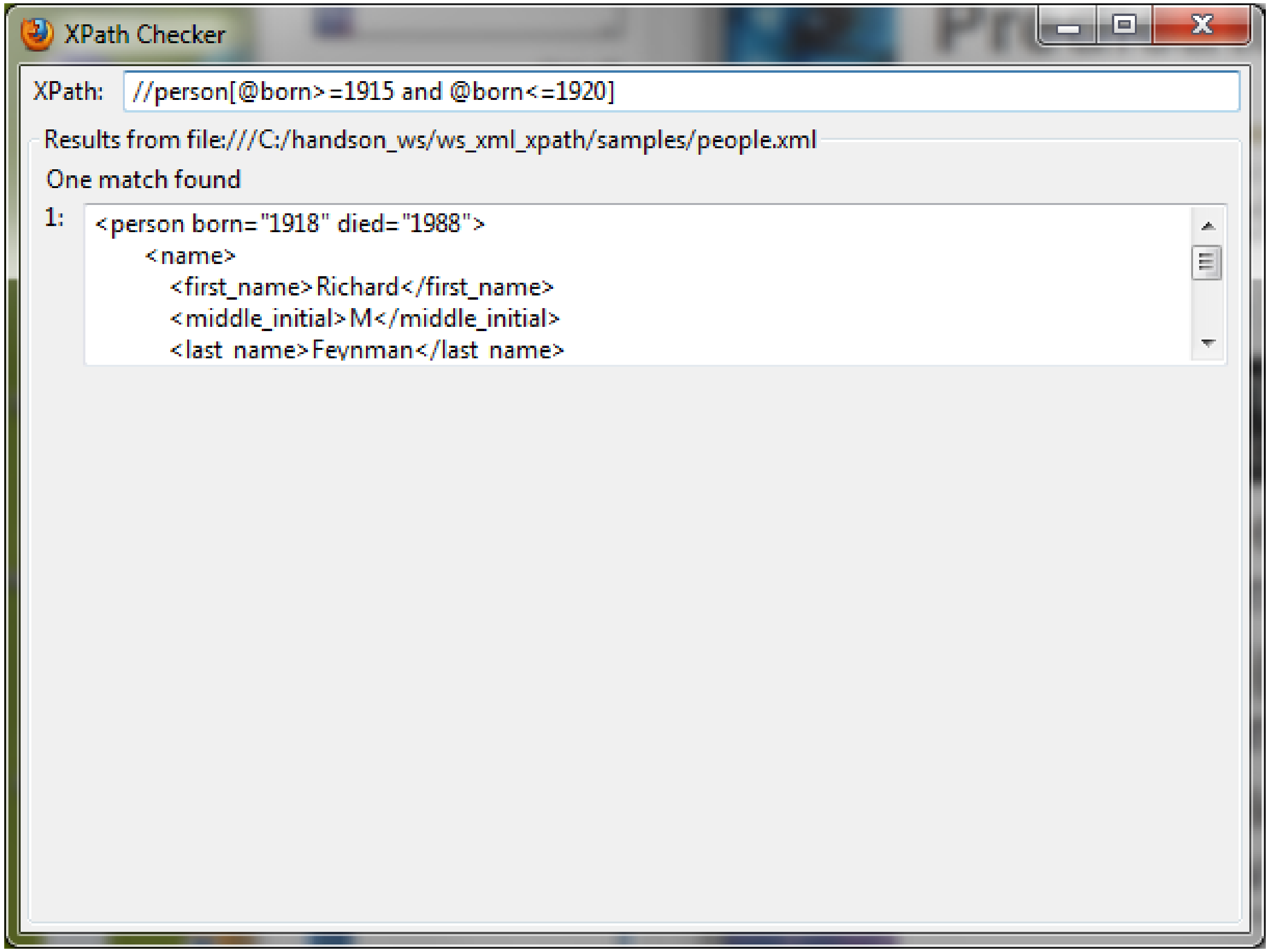
One match found

1:
```
<person born="1912" died="1954">
      <name>
         <first_name>Alan</first_name>
         <last_name>Turing</last_name>
```

Predicates - “and” & “or”

- Example 5
  - //person[@born>=1915 and @born<=1920]
  - person elements with *born* attribute value between 1915 and 1920, inclusive
  - //name[first_name=“Dick” or first_name=“Sang”]
  - name elements that have first_name child whose value is “Dick” or “Sang”

//person[@born>=1915 and @born<=1920]



XPath: //person[@born>=1915 and @born<=1920]

Results from file:///C:/handson_ws/ws_xml_xpath/samples/people.xml

One match found

1:
```
<person born="1918" died="1988">
      <name>
         <first_name>Richard</first_name>
         <middle_initial>M</middle_initial>
         <last_name>Feynman</last_name>
```

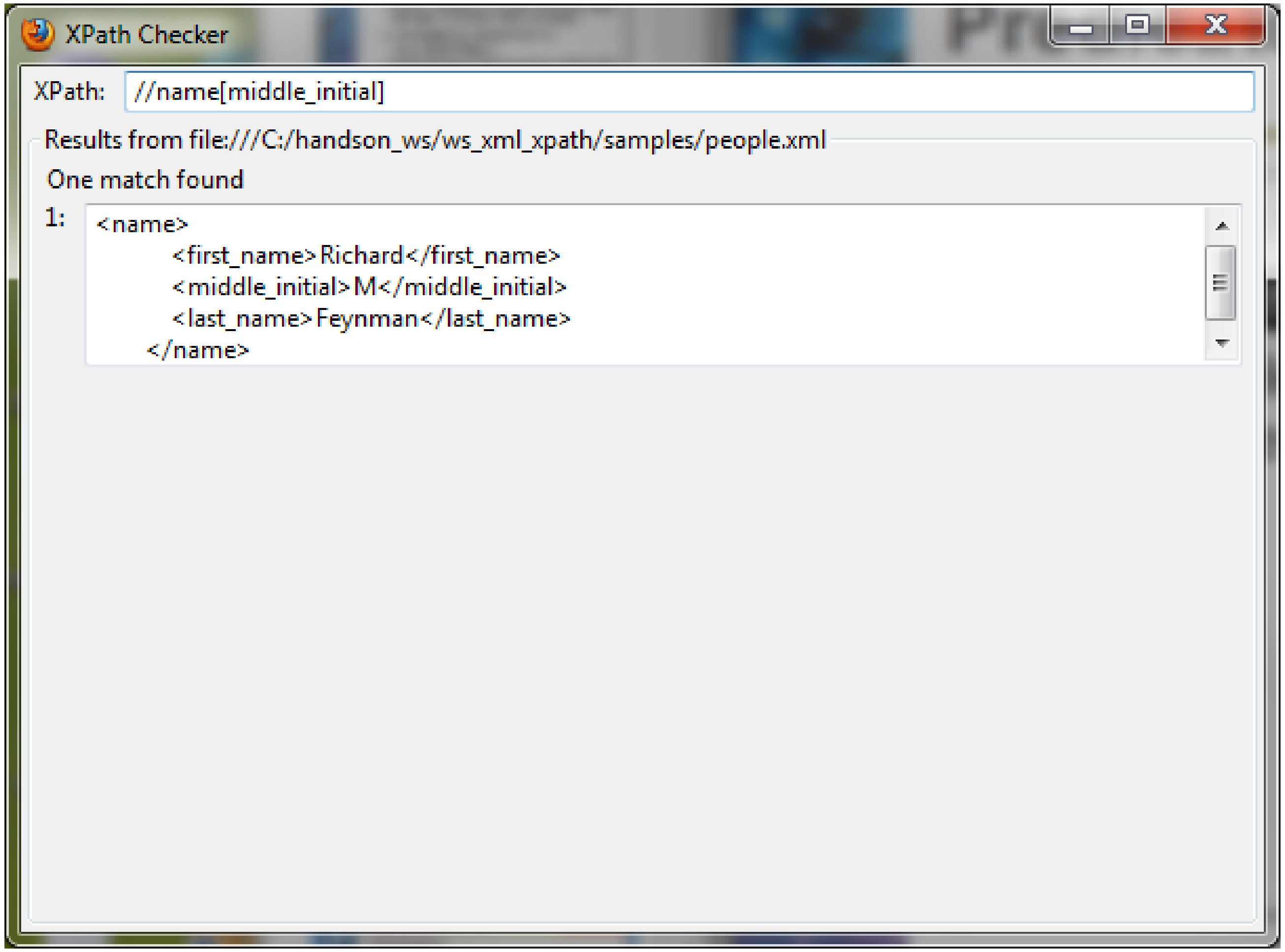
Predicates – non-boolean expression

- Predicates could be non-boolean expression
  - They will be converted into boolean
- Examples
  - Number
    - True if not 0
  - Node set
    - True if node set is non-empty
  - String
    - True if non-empty string

Predicates

- Example 6
  - > //name[middle_initial]
  - > *name* elements which have *middle_initial* child element

//name[middle_initial]



XPath: //name[middle_initial]

Results from file:///C:/handson_ws/ws_xml_xpath/samples/people.xml

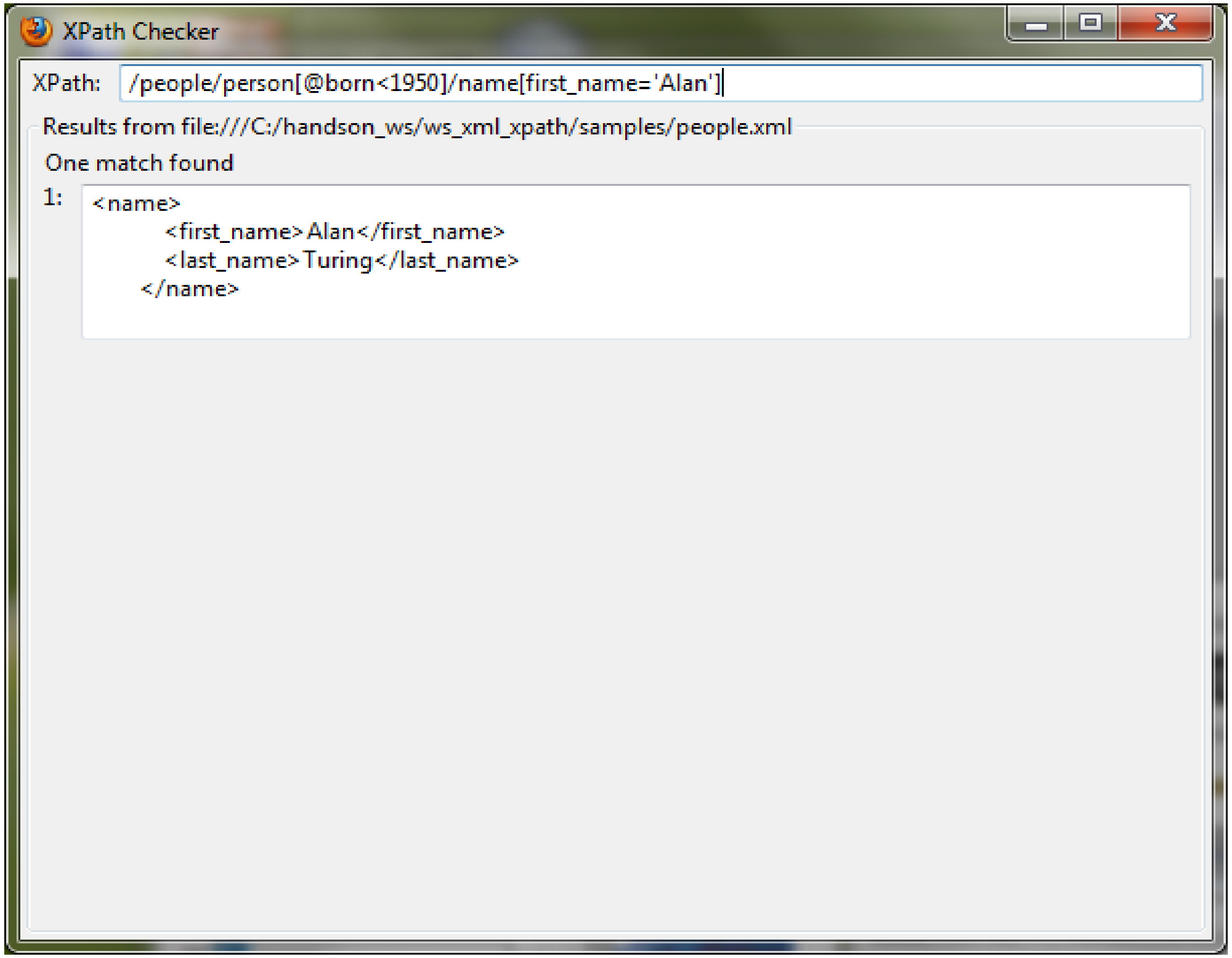
One match found

1:
```
<name>
        <first_name>Richard</first_name>
        <middle_initial>M</middle_initial>
        <last_name>Feynman</last_name>
    </name>
```

Predicates

- Can be applied to each step in a location path
- Example 7
  - /people/person[@born<1950]/name[first_name="Alan"]
  - Select all people child elements of the root element, then select all person elements whose *born* attribute has a value numerically less than 1950, then select all *name* child elements that have a *first_name* child element whose value is "*Alan*"

/people/person[@born<1950]/name[first_name='Alan']

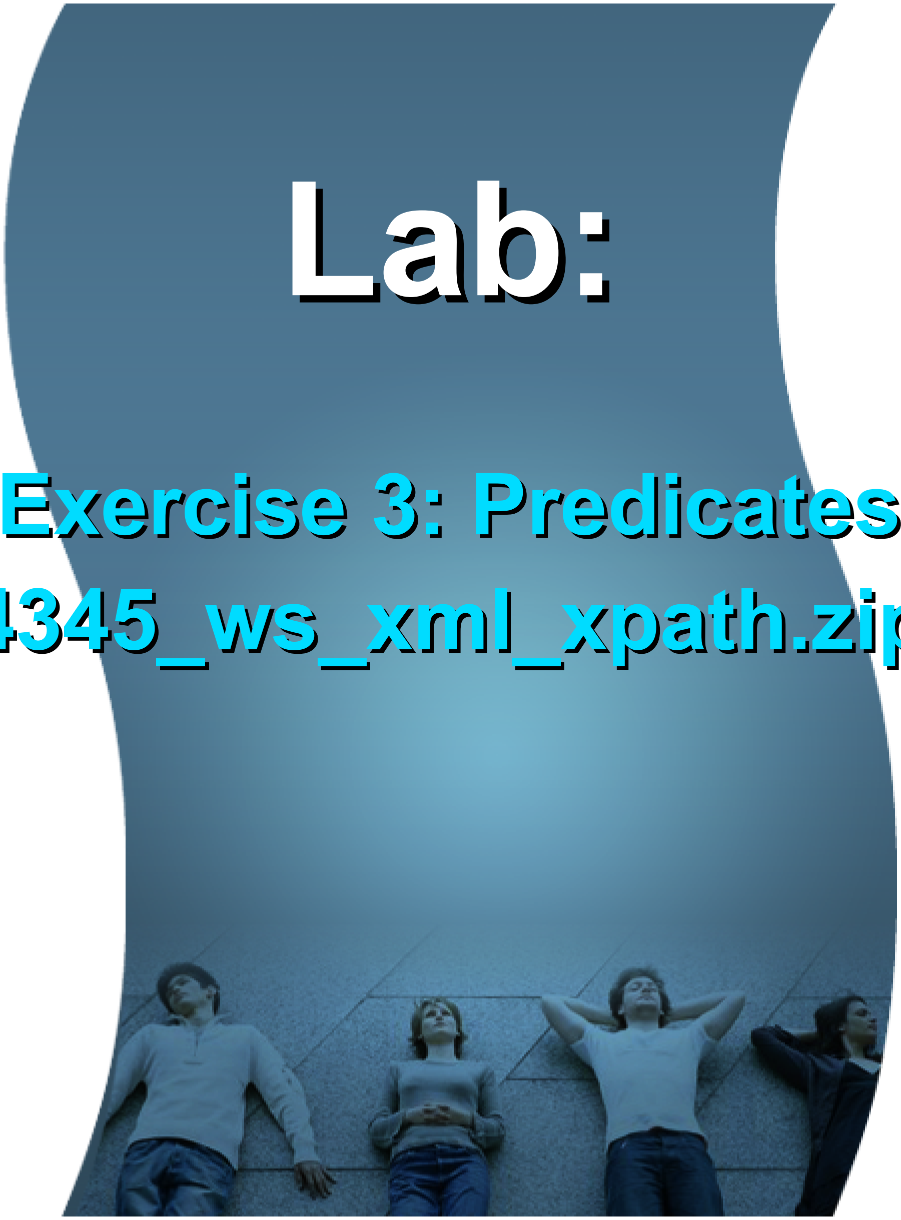


XPath: /people/person[@born<1950]/name[first_name='Alan']

Results from file:///C:/handson_ws/ws_xml_xpath/samples/people.xml

One match found

1:
```
<name>
        <first_name>Alan</first_name>
        <last_name>Turing</last_name>
    </name>
```



Lab:

Exercise 3: Predicates

4345_ws_xml_xpath.zip

Non-Node set Expressions

Non-Node Set Expressions

- Numbers
  - > 3.141529
  - > 2+2
- Strings
  - > “JavaPassion”
- Booleans
  - > true()
  - > 32.5 <76.2E-21
  - > position() = last()
- They cannot be used in *match* pattern of *xsl:template*

Numbers

- Basic arithmetic operators
  - +, -, *, div, mod
- Example
  - <xsl:value-of select=“6*7”/>

Strings

- Ordered sequence of Unicode characters
- Work with = and != comparison operators

Functions

Functions

- Might return a value of one of the four types
  - node set
  - boolean
  - number
  - string

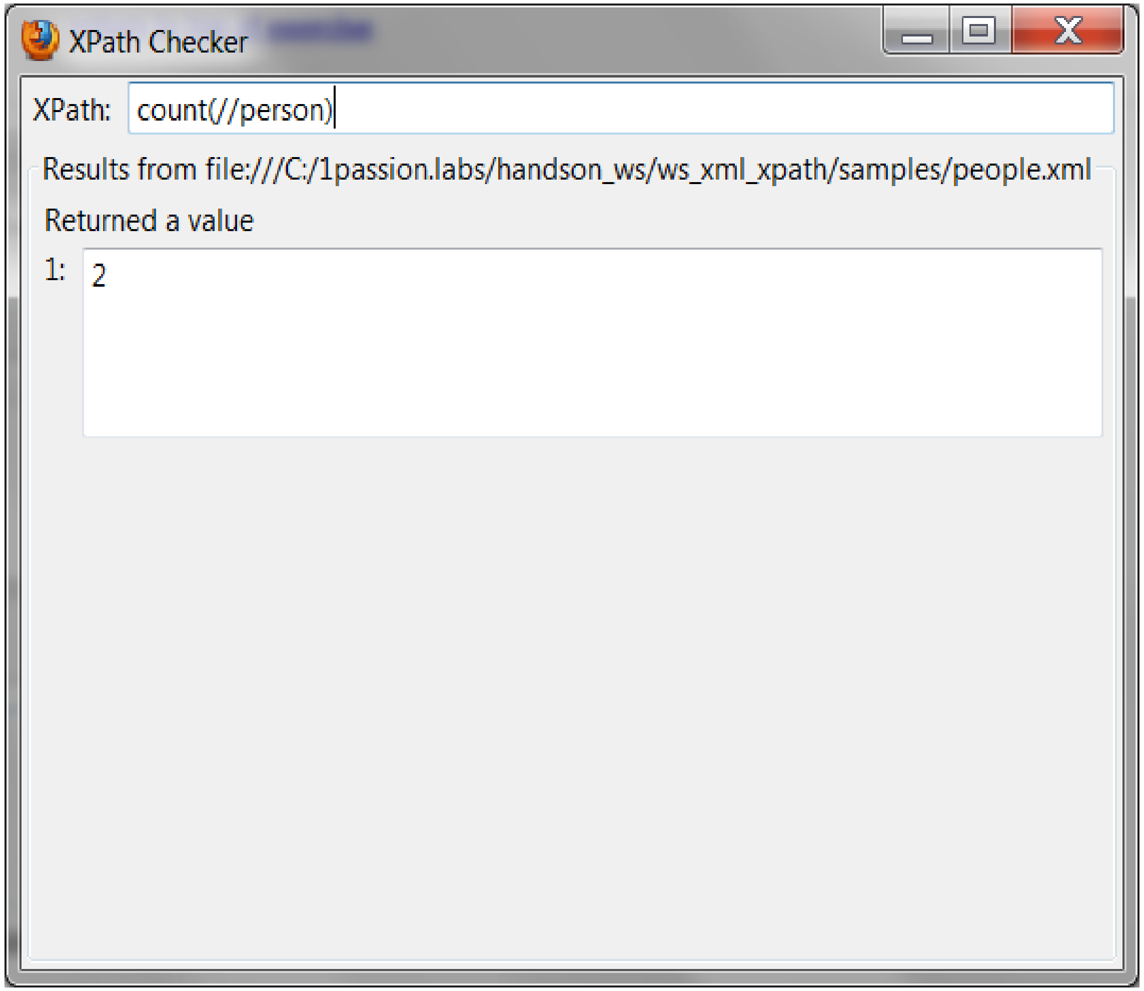
Note Set Functions

- position()
  - Current node’s position in the node set

```
<xsl:template match=“person”>
      Person <xsl:value-of select=“position()”/>
      <xsl:value-of select=“name”/>
</xsl:template>
```

- count(<location path>)
  - Number of nodes in the node set argument

count(//person)


XPath Checker

XPath: count(//person)

Results from file:///C:/1passion.labs/handson_ws/ws_xml_xpath/samples/people.xml

Returned a value

```
1: 2
```

String Functions

- string()
  - Converts any type of argument to a string
    - Booleans: “true” or “false”
    - Node sets: string value of first node in the set
- starts-with(arg1, arg2)
  - Returns true if the first argument starts with second argument
    - starts-with(‘Richard’, ‘Ric’) returns true
    - starts-with(‘Richard’, ‘Rick’) returns false

String Functions

- contains(arg1, arg2)
  - Returns true if first argument contains the second argument
    - contains(‘Richard’, ‘ar’) returns true
    - contains(‘Richard’,’art”) returns false
- substring(arg1, position, length)
  - Returns substring of arg1 whose length is length starting from postion
  - length argument is optional
    - substring(‘MM/DD/YYYY’, 1, 2) returns ‘MM’
    - substring(‘MM/DD/YYYY’, 2) returns ‘M/DD/YYYY’

String Functions

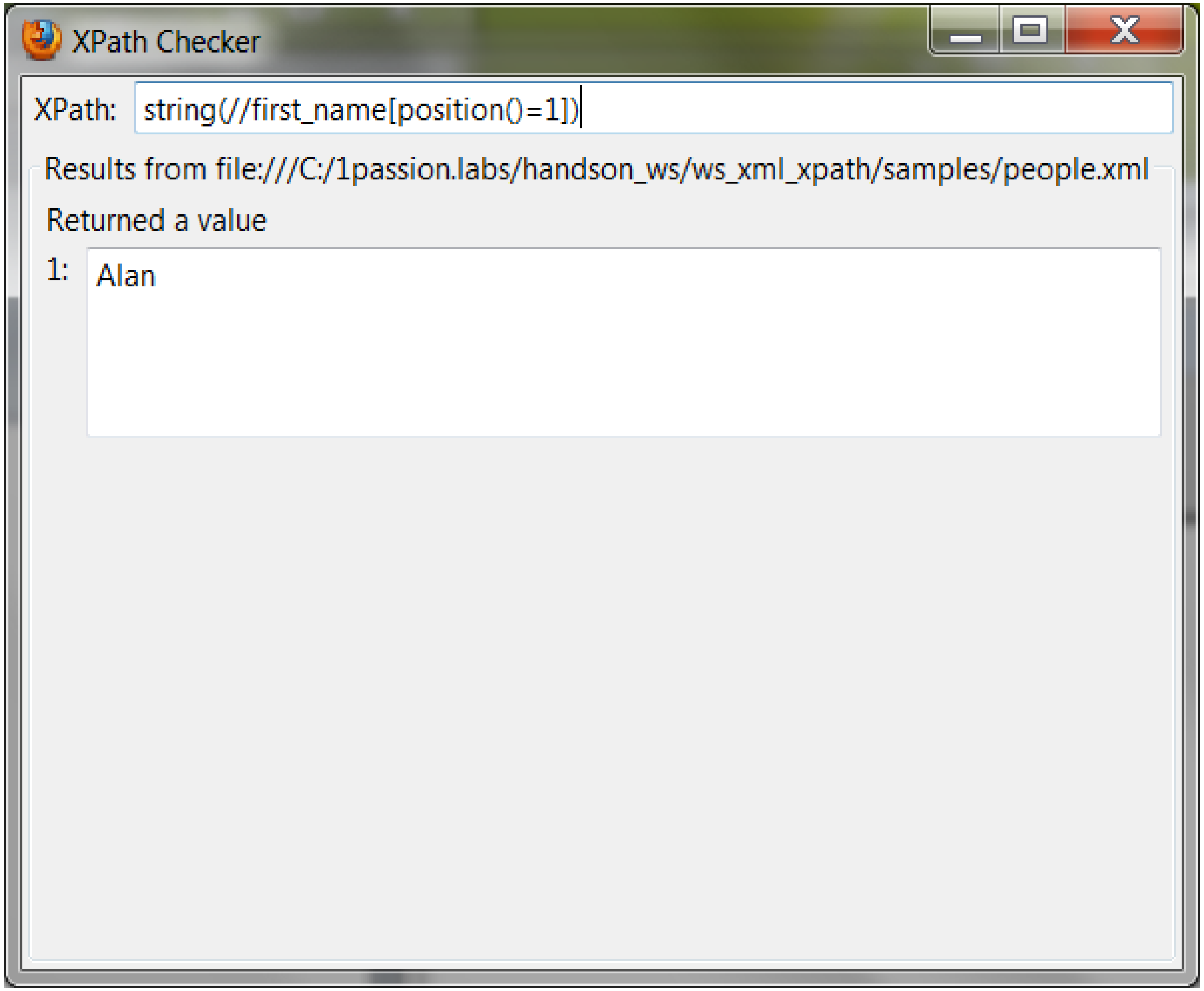
- substring-before(arg1, arg2)
  - Returns the substring of the first argument string that precedes the second argument’s initial appearance
    - substring-before(‘MM/DD/YYYY’, ‘/’) returns ‘MM’
- substring-after(arg1, arg2)
  - Returns the substring of the first argument string that follows the second argument’s initial appearance
    - substring-after(‘MM/DD/YYYY’, ‘/’) returns ‘DD/YYYY’

String Functions

- string-length(arg1)
  - Returns a length of the string value of the argument
  - Whitespace characters are included
  - Markup characters are not counted
  - arg1 is optional - returns length of context node
    - string(//first_name[position()=1])

      Alan
    - string-length(//first_name[position()=1]) returns 4

string(//first_name[position()=1])



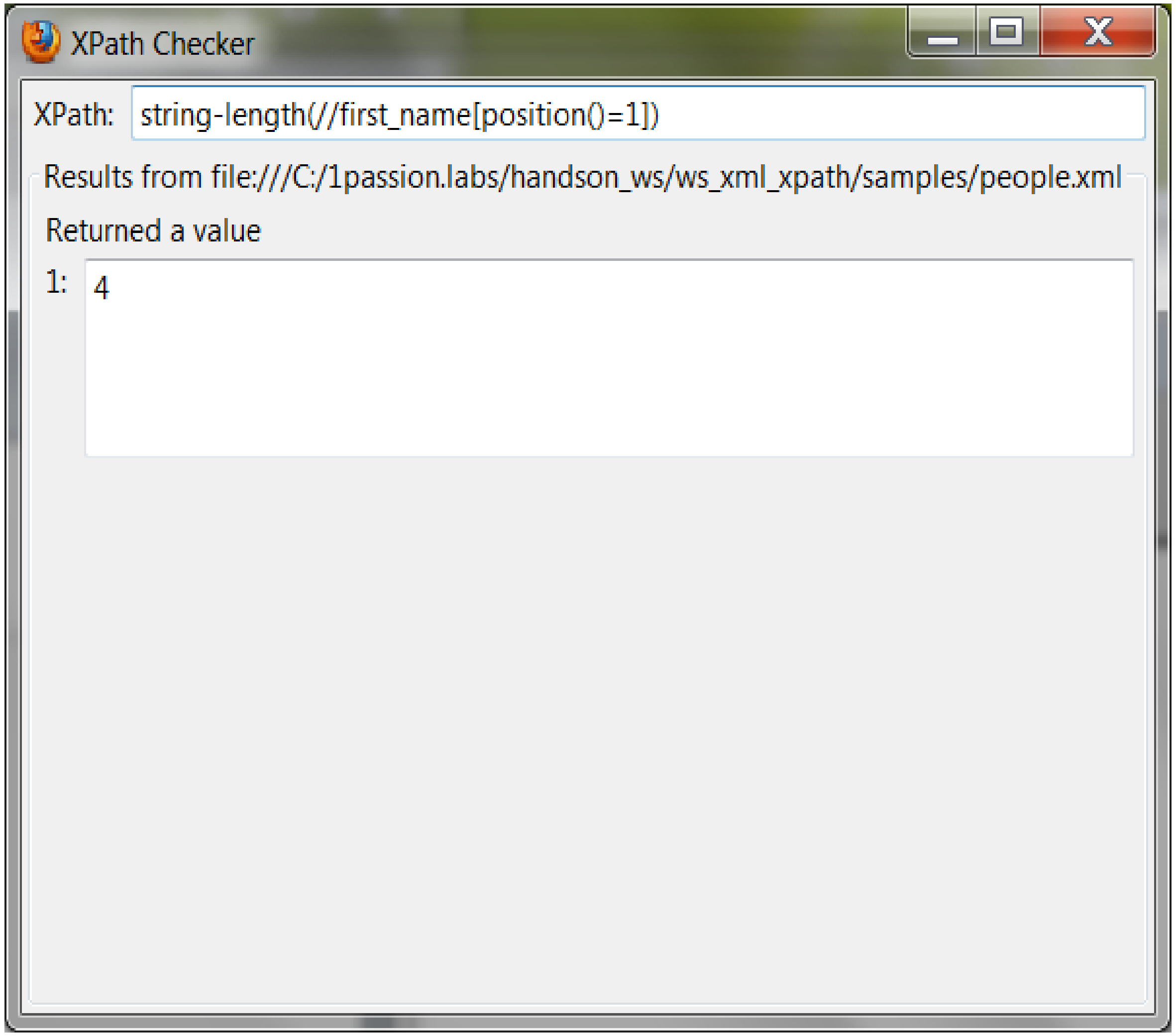
XPath: string(//first_name[position()=1])

Results from file:///C:/1passion.labs/handson_ws/ws_xml_xpath/samples/people.xml

Returned a value

1: Alan

string-length(//first_name[position()=1])



XPath: string-length(//first_name[position()=1])

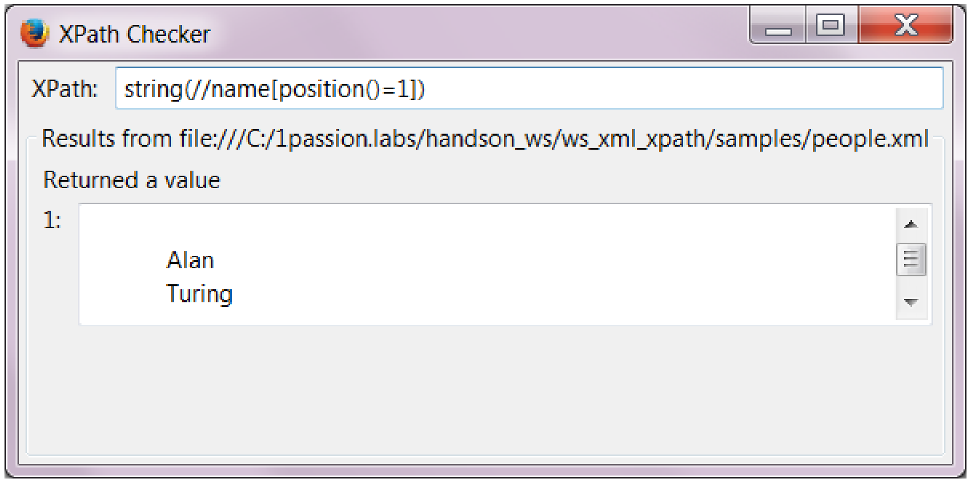
Results from file:///C:/1passion.labs/handson_ws/ws_xml_xpath/samples/people.xml

Returned a value

```
1: 4
```

Normalizing white spaces

- normalize-space(arg) - Normalize white spaces


XPath Checker

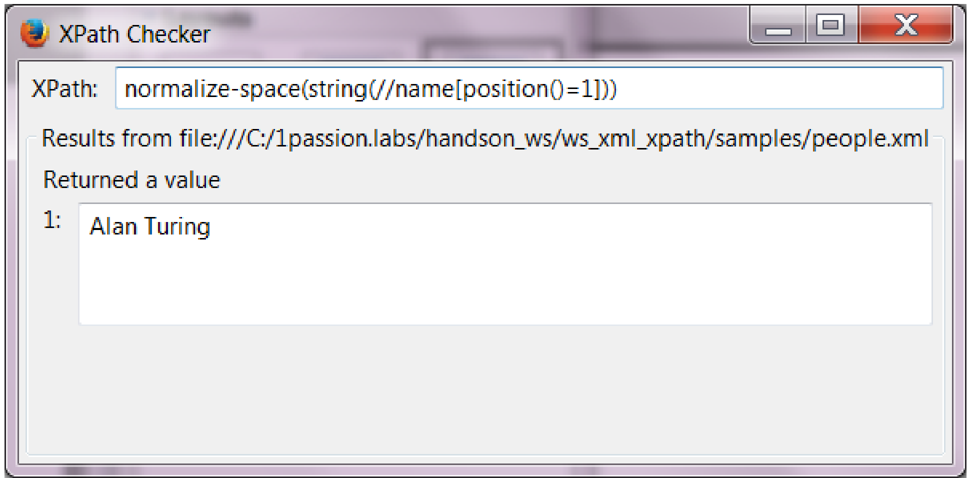
XPath: string(//name[position()=1])

Results from file:///C:/1passion.labs/handson_ws/ws_xml_xpath/samples/people.xml

Returned a value

```
1:
          Alan
          Turing
```

without normalizing


XPath Checker

XPath: normalize-space(string(//name[position()=1]))

Results from file:///C:/1passion.labs/handson_ws/ws_xml_xpath/samples/people.xml

Returned a value

```
1: Alan Turing
```

with normalizing

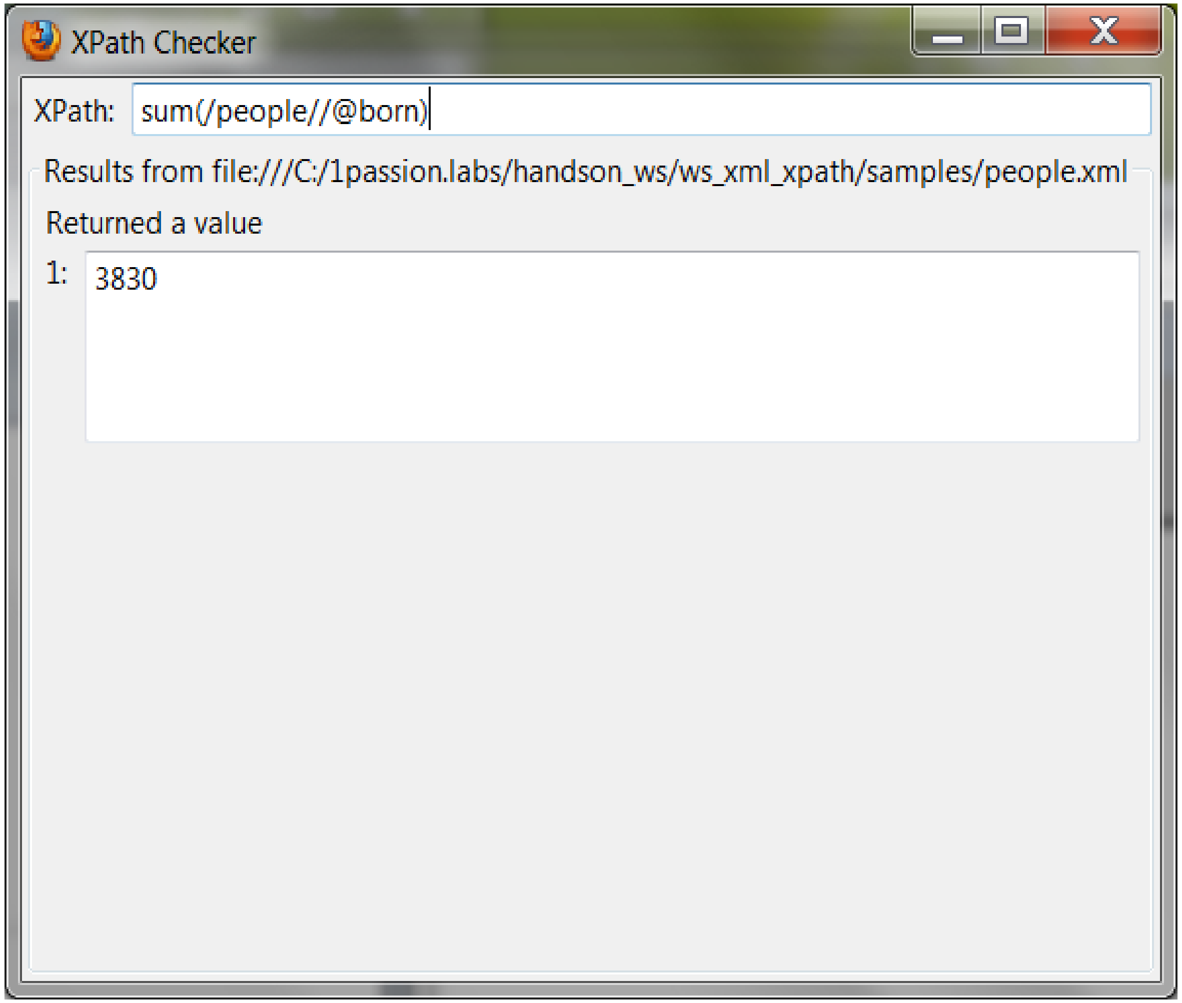
Boolean Functions

- true()
  - returns true
- false()
  - returns false
- boolean(arg1)
  - Converts arg1 to a boolean and returns result
  - If no argument, use context node
  - If arg1 is node set, true if it contains at least one node

Number Functions

- number(arg1)
  - Converts arg1 to a number
  - If no argument, use context node
- sum(arg1)
  - Take a node set as an argument, converts each node in the set to its string value, then converts each of those strings to a number. And finally, it adds the numbers and returns the result

sum(/people//@born)

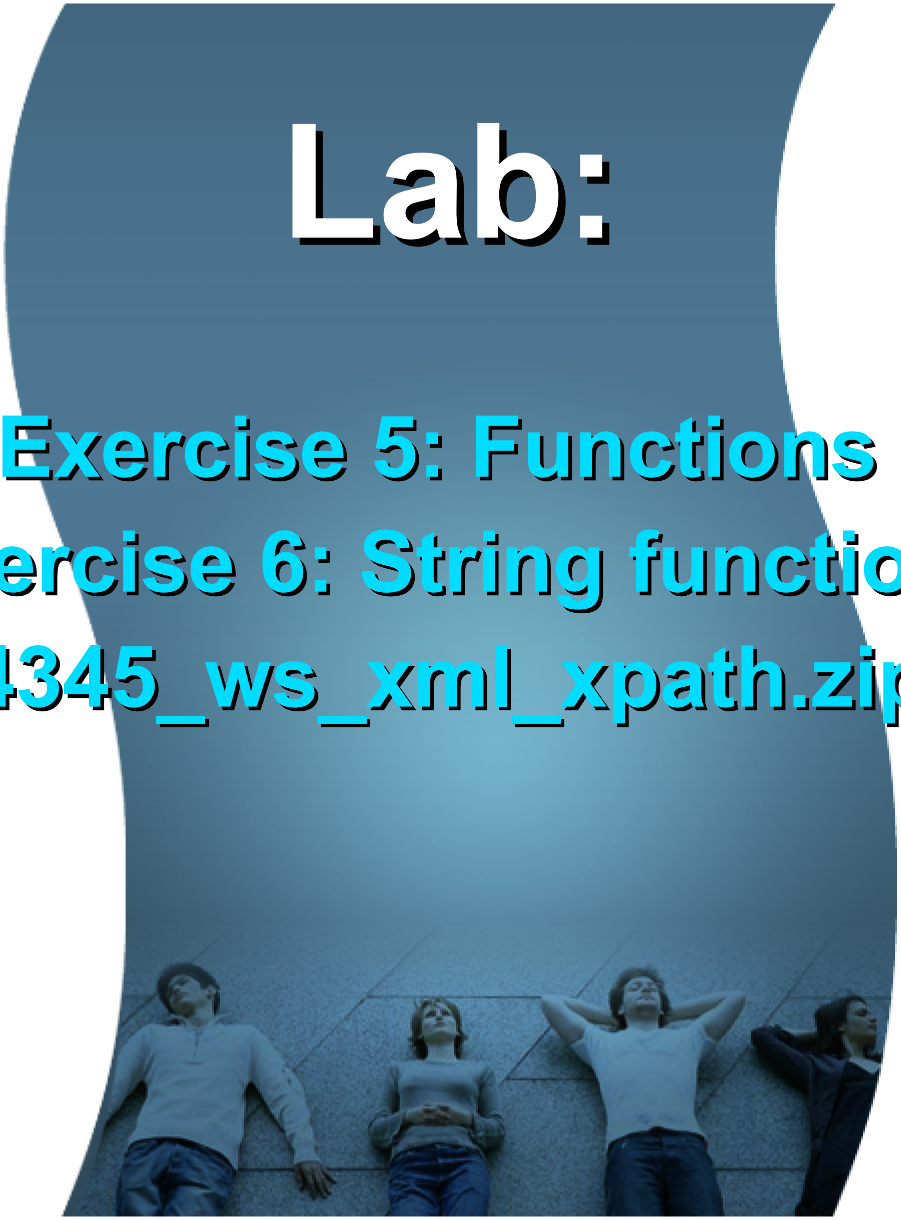


XPath: sum(/people//@born)

Results from file:///C:/1passion.labs/handson_ws/ws_xml_xpath/samples/people.xml

Returned a value

1: 3830



Lab:

Exercise 5: Functions

Exercise 6: String functions

4345_ws_xml_xpath.zip

Summary

Summary

- XPath expression data types
- Node types
- Node set
- Location path
- Wild cards
- Predicates
- Functions

References

- “XML in a Nutshell” written by Elliotte Rusty Harold & W. Scott Means, O’Reilly, Jan. 2001(1st Edition), Chapter 9 “XPath”

Learn with Passion!
JPassion.com