XTraining

Part 4: Regex

…and:

Putting It All Together!

Regine Heberlein 5/29/2018

Challenge

Across all our repos…

- how many finding aids have no dsc?
- how many finding aids have more than one dsc?
- how many finding aids contain extents with a unit attribute?
- what are the unique values used in the unit attribute?

Challenge Answers

NB: Results may vary depending on time of execution and SVN status.

NB: behavior of XPath over directories is "for each file" → instead of count(), use built-in count

1. //ead[not(descendant::dsc)]: 86
2. //ead[descendant::dsc[2]]: 1639
3. //ead[descendant::extent/@unit]//eadid: 2219
4. Better handled with script—see following

Challenge #4 Query (No-frills edition)

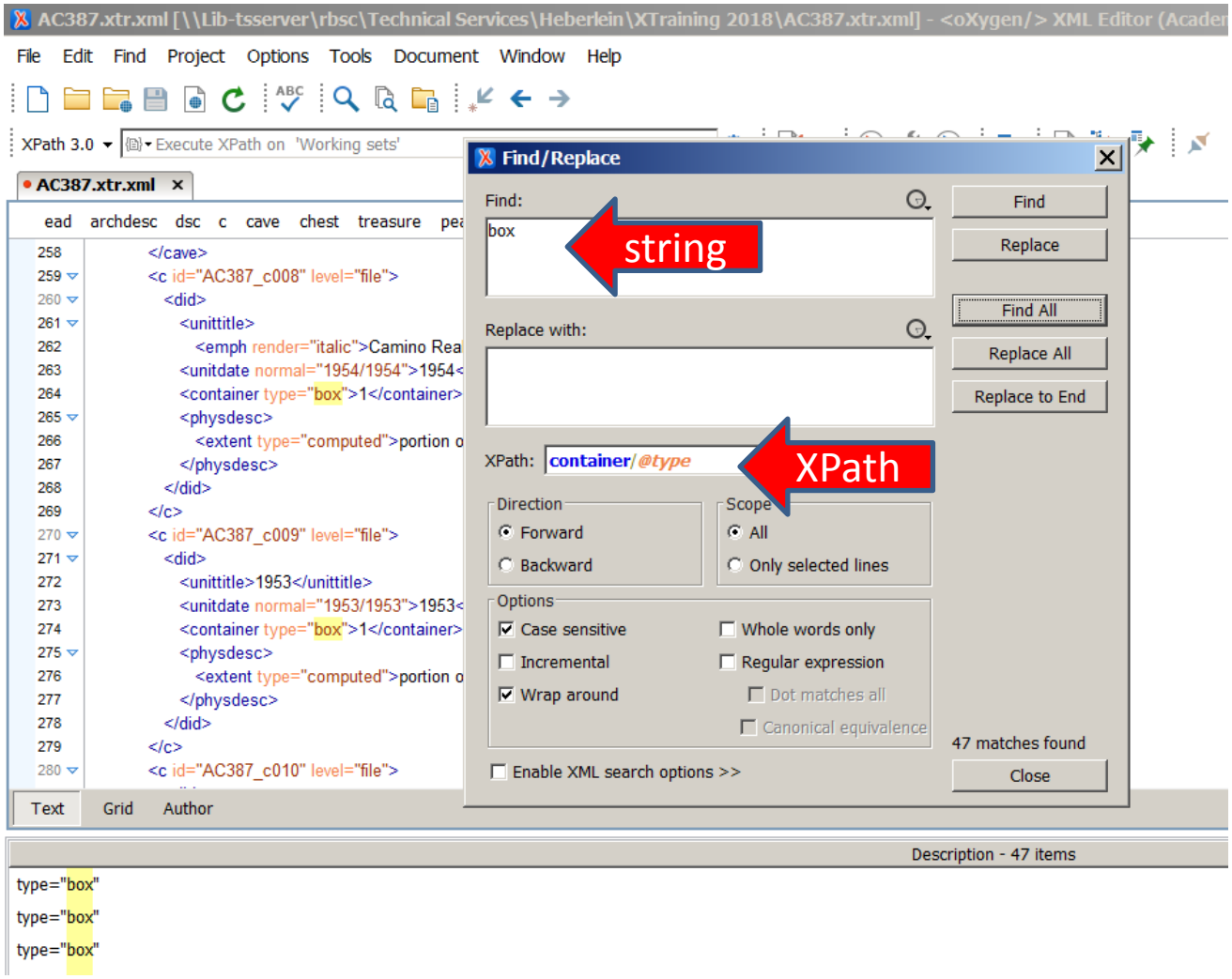
```
xquery version "1.0";
declare namespace ead = "urn:isbn:1-931666-22-9";
declare default element namespace "urn:isbn:1-931666-22-9";
declare copy-namespaces no-preserve, inherit;

declare variable $COLL as document-node()+ :=
collection("file:///C:/Users/heberlei/Documents/SVN%20Working%20Copies/trunk/eads?recurse=yes;select=*.xml");

distinct-values($COLL//@unit)
```

Challenge #4 Query Results

footage
boxes
volumes
folders
folios
items
pages
box
linear feet

packages
tubes
shelves
linearfeet
folder
leaves
item
boxess
portfolios
volume

Useful Tool: Query a Directory or Project from Find/Replace Dialog



Let's replace something!

Replace all box-type containers with basket-type containers.

1. Use find/replace
2. In "find", enter the search string
3. In XPath, return the node you want to change
4. **Test** by clicking on "find all"
5. In "replace", enter the replacement string

Regular Expressions (“regex”)

“Some people, when confronted with a problem, think ‘I know, I'll use regular expressions.’ Now they have two problems.”

--Jamie Zawinski, 1997

What Regex Will Process

- Literal characters
- Metacharacters
- Quantifiers
- Character classes

Useful Resources

http://www.regular-expressions.info/

http://regexpal.com/

http://regexbuddy.com/ ($$)

Regex Metacharacters

.	**any character (☝)**	**()**	**group**
****	**escape character**	**[]**	**range**
|	**union**	**[^]**	**negative range**
?	**zero or one**	**{}**	**count**
*****	**zero or more**		
+	**one or more**	**^**	**anchor: start of string**
		$	**anchor: end of string**

Try these

.

.+

Charles

Charles.

Charles\.

Charl.

Charl.+

Charlie?

Charl(y|ie?)

Charl[eo]

Charl[^eo]

Charl(ie|ene)

Charl.{2}

Charl.{2,4}

Very Cool: Escaping Sequences, Part I

\d	any digit
\D	any non-digit
\w	any word character
\W	any non-word char.
\s	any whitespace char.
\S	any non-whitespace char.
\t	tab
\n	newline
\r	carriage return

Aside for the Determined: Note the Difference

`.+`

(doesn't match \n)

v.

`[\D\S]+`

(matches anything)

Aside for the Determined: Note the Difference

[\D\S]+

"any character that is either not a digit or not whitespace [i.e., anything]"

[^\d\s]+

"any character that is neither a digit nor whitespace"

[^\D\S]+

"any character that is neither not a digit nor not whitespace [i.e., nothing]"

Even Cooler!
Escaping Sequences, Part II: Character Classes

Syntax:

\p{} match characters in category

\P{} match characters not in category

Some Classes:

L	Letter	S	Symbols
M	Mark	Z	Separators
P	Punctuation		

Try it out

Interpret the results from this:

Find: \p{L}+

XPath: unitdate[not(matches(., 'undated'))]

Enable XML search options: Element contents

→ Finds letter characters in unitdates other than those set to 'undated'

The Coolest! Escaping Sequences Part II: Unicode Blocks

Syntax:

\p{Is*block*} / \P{Is*block*} (XML regex)

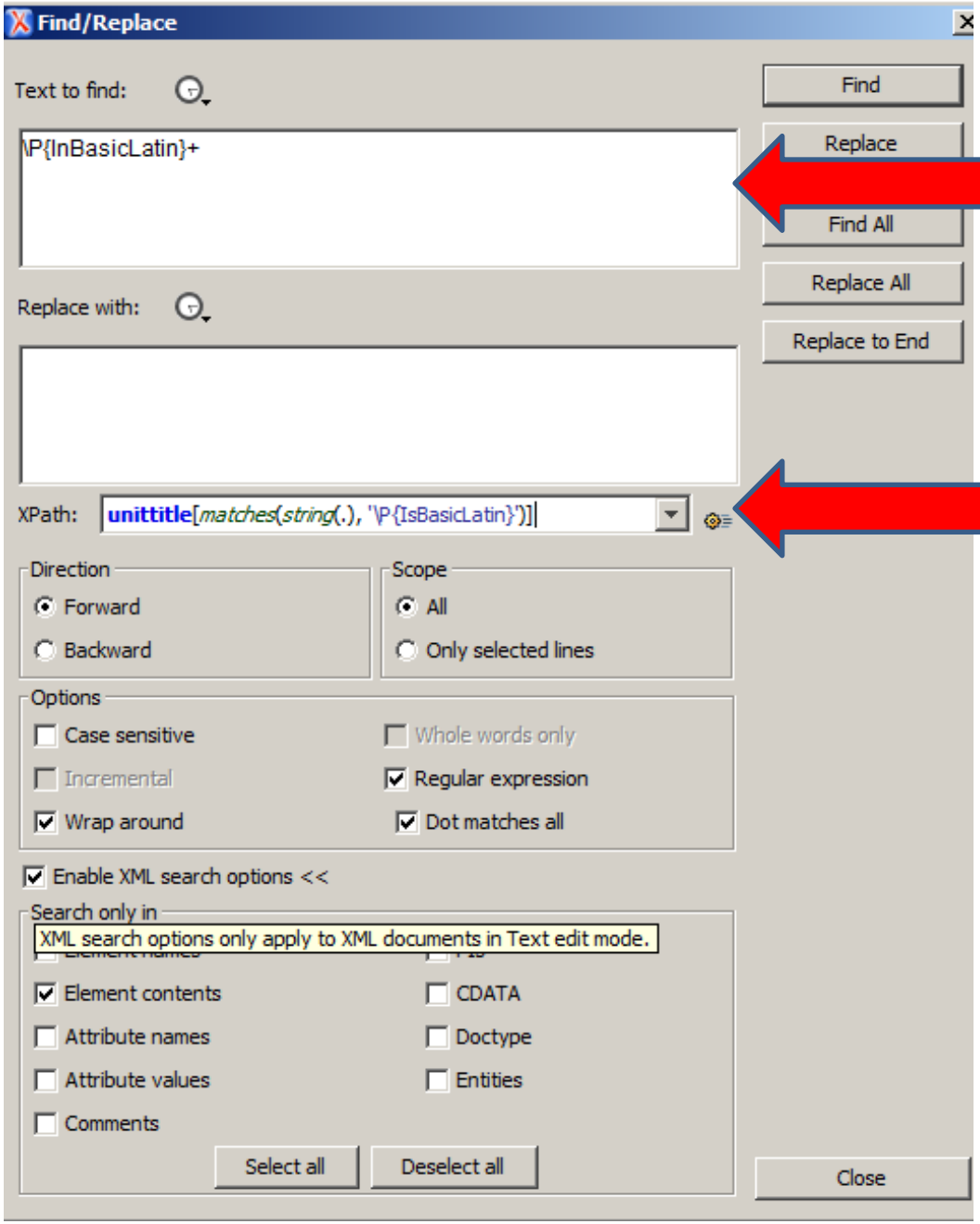
\p{In*block*} / \P{In*block*} (Java regex)

Some Blocks:

BasicLatin

Latin-1Supplement

GeneralPunctuation



NB: Search box takes Java regex

NB: Xpath dialog takes XML regex

Exercise

1. Find any unittitles containing characters not in the Basic Latin Block.
2. Find any characters in unittitle that are not in the Basic Latin Block.
3. Find any characters in element contents that are in Latin-1 Supplement
4. Puzzler! Find any characters in unittitle that are neither in Basic Latin nor in Latin-1 Supplement

Exercise Answer Key

#1. Search: [\D\S]+

XPath: unittitle[matches(string(.), '\P{IsBasicLatin}')]

→ NB: XPath filter uses XML regex engine

#2. Search: \P{InBasicLatin}+

→NB: Find/Replace uses Java regex engine

XPath: unittitle

#3. Search: \p{InLatin-1Supplement}+

Use XPath text() OR option "element content"

#4. Search: [^\p{InBasicLatin}\p{InLatin-1Supplement}]+

XPath: unittitle

Greedy Quantifiers
(And How To Make Them Lazy)

+

*

?

{2,}

→ Add '?'

Try These

Charles+ v. Charles+?

Charles* v. Charles*?

Charles? v. Charles??

Functions that Take Regex

matches()

tokenize()

replace()

//creation[contains(., 'MarcEdit')]

//creation[matches(., 'MarcEdit')]

→***What about*** *"*MARCEdit***"?***

//creation[contains(., 'MarcEdit') or contains(., 'MARCEdit')]

//creation[matches(., 'marcedit', **'i'**)]

Regular Expression Flags

i	case insensitive
s	dot matches all (including \n)
m	multiline mode
x	ignore whitespace

```
//persname[matches(., 'charl.+', 'is')]
```

tokenize()

//unitdate[matches(., '\d{4}-\d{4}')]

//unitdate[matches(., '\d{4}-\d{4}')]/tokenize(., '-')

Puzzler! In Series 2 of AC387.xtr, split the series unittitle on the colon and return the second token only.

(Hint: analyze your data first with //@level[.='series'])

→ //unittitle[ancestor::c[@level='series'][2]]/tokenize(., ':')[2]

matches()

//@normal[matches(., '/')]

//@normal[matches(., '^\d{4}$')]

//@normal[not(matches(., '/'))]

Puzzler! Find any unittitles containing characters from both Latin-1 Supplement and General Punctuation

→Search: [\D\S]+

→XPath: unittitle

[matches(string., '\p{IsLatin-1Supplement}') and matches(., '\p{IsGeneralPunctuation}')]

replace()

```
replace(
        //unitdate[matches(., 'no date')],
        'no date',
        'undated'
)
```

Puzzler! Replace the second part of any ranges in unitdate with 'no clue'

Hint: Use "for…in…" syntax

replace() Puzzler Explained

replace() and tokenize() both take an item as input, not a sequence.

If you tried a direct approach starting with the replace() function, you may have gotten an error back for that reason.

Instead, use the for…in… syntax, which will apply the function to each item in the input sequence:

```
for $i in //unitdate[matches(., '\d{4}-\d{4}')]
return replace(tokenize($i, '-')[2], '.+', 'no clue')
```

Replacing with named groups

()

Find: Charless?|Charlotte

Use proxy anchors: "<" and ">"

(disable XML search options)

group:

(>)(Charl)(ess?|otte)(<)

Replace:	$0	Ctrl-z
	$1$4	Ctrl-z
	$1$2ene$4	Ctrl-z
	$1$3$5 ☝	

Pesky AC123!

Get accession numbers out of scopecontent and into unitid using XPath and regex in the oXygen editor

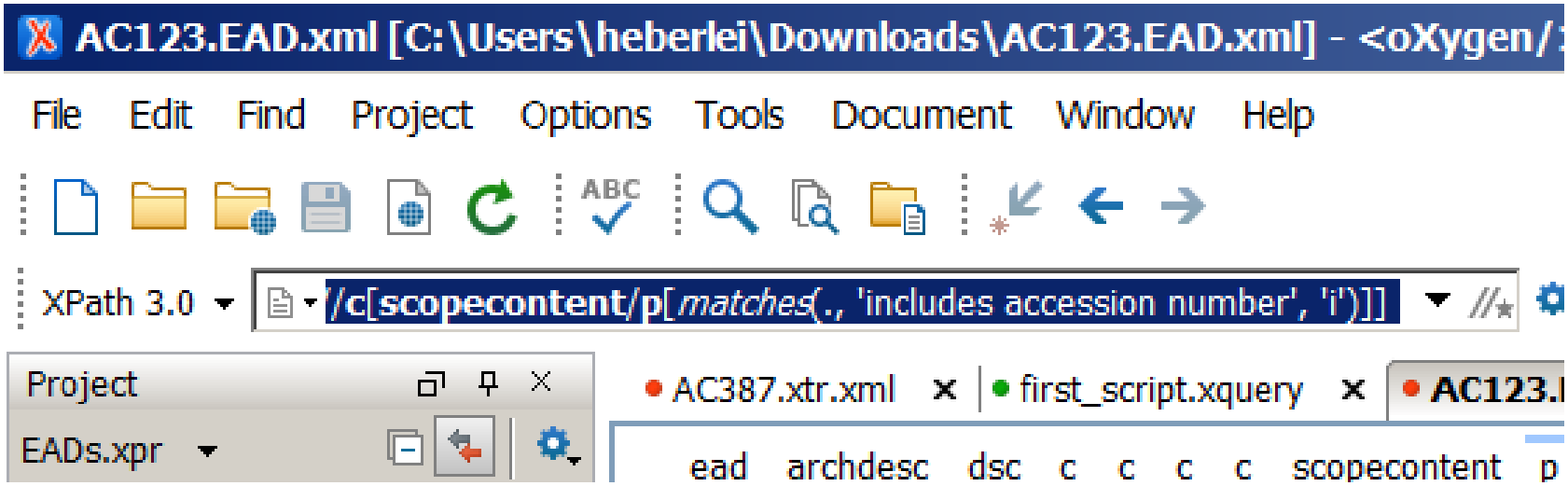
Current

```
<c level="file" id="AC123_c05939">
    <did>
            <container type="volume">
                    Accession Book 6
            </container>
            <unittitle>
                    undated
            </unittitle>
            <unitdate>
                    undated
            </unitdate>
            <physdesc>
                    <extent type="computed">
                        1 volume
                    </extent>
            </physdesc>
    </did>
    <scopecontent>
            <p>
                Includes accession numbers
                25001-30000.
            </p>
    </scopecontent>
</c>
```

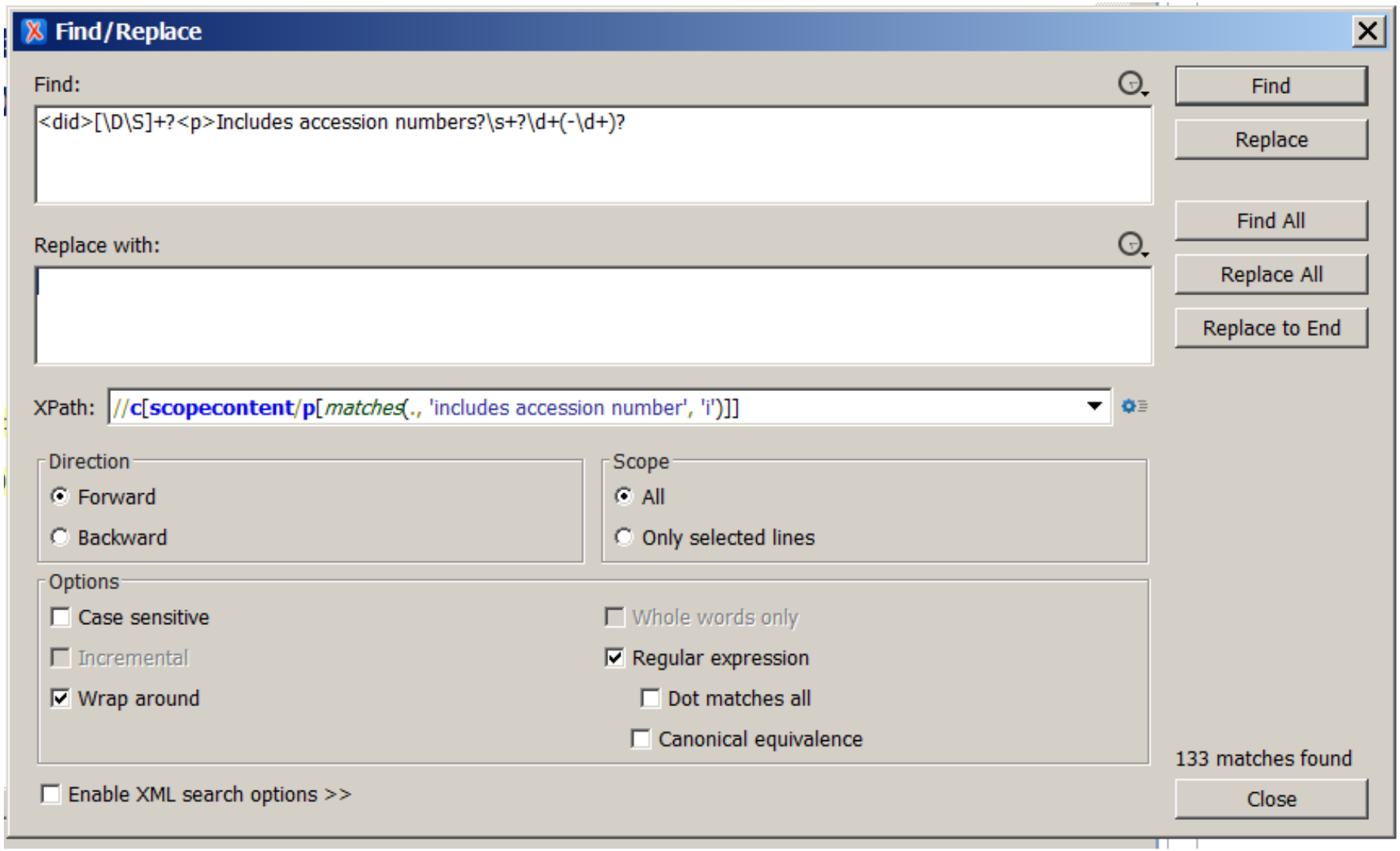
Desired

```
<c level="file" id="AC123_c05939">
    <did>
            <unitid type="accessionnumber">
                    25001-30000
            </unitid>
            <container type="volume">
                    Accession Book 6
            </container>
            <unittitle>
                    undated
            </unittitle>
            <unitdate>
                    undated
            </unitdate>
            <physdesc>
                    <extent type="computed">
                        1 volume
                    </extent>
            </physdesc>
    </did>
</c>
```

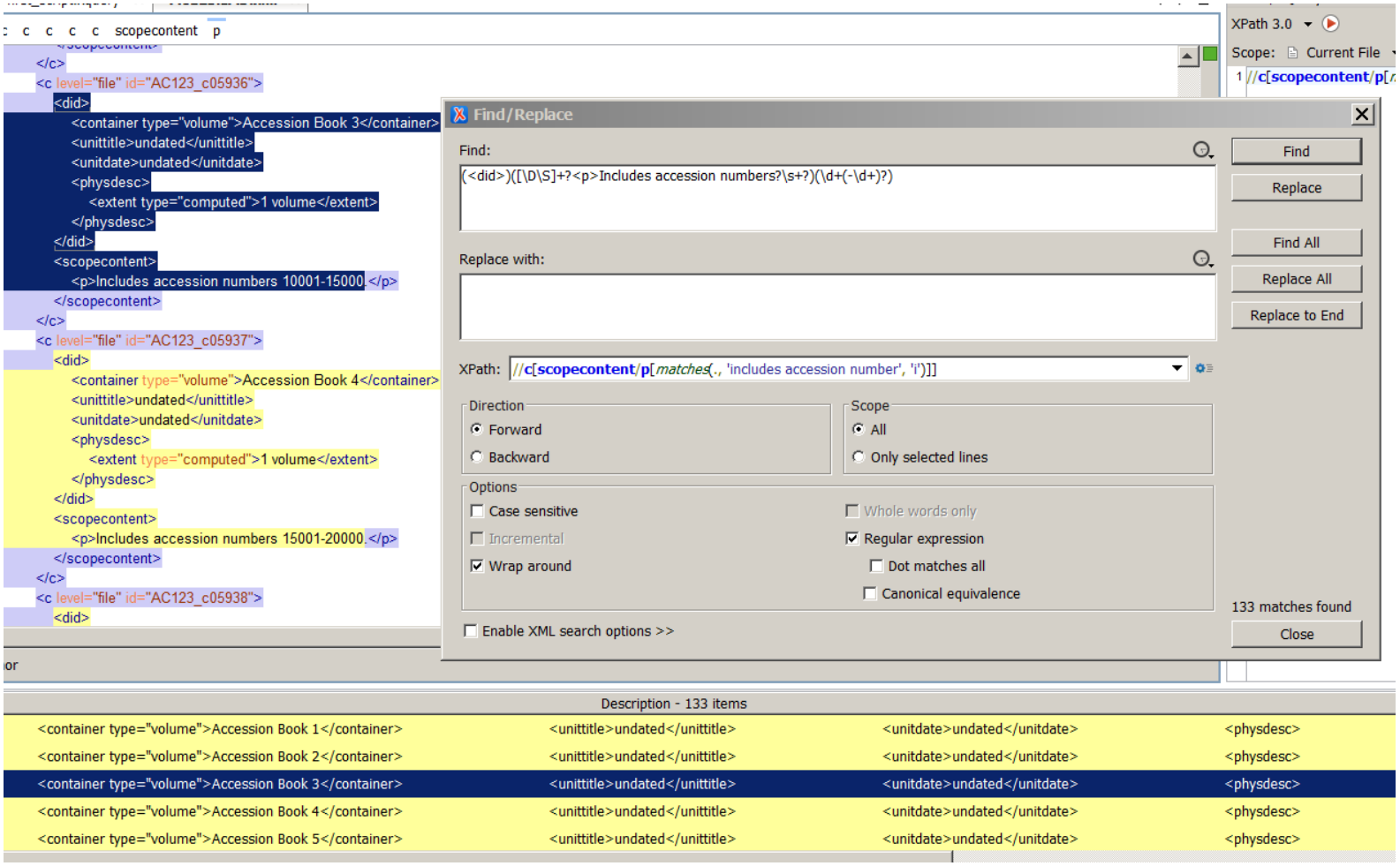
```
//c
[
scopecontent/p
[matches(., 'includes accession number', 'i')]
]
```



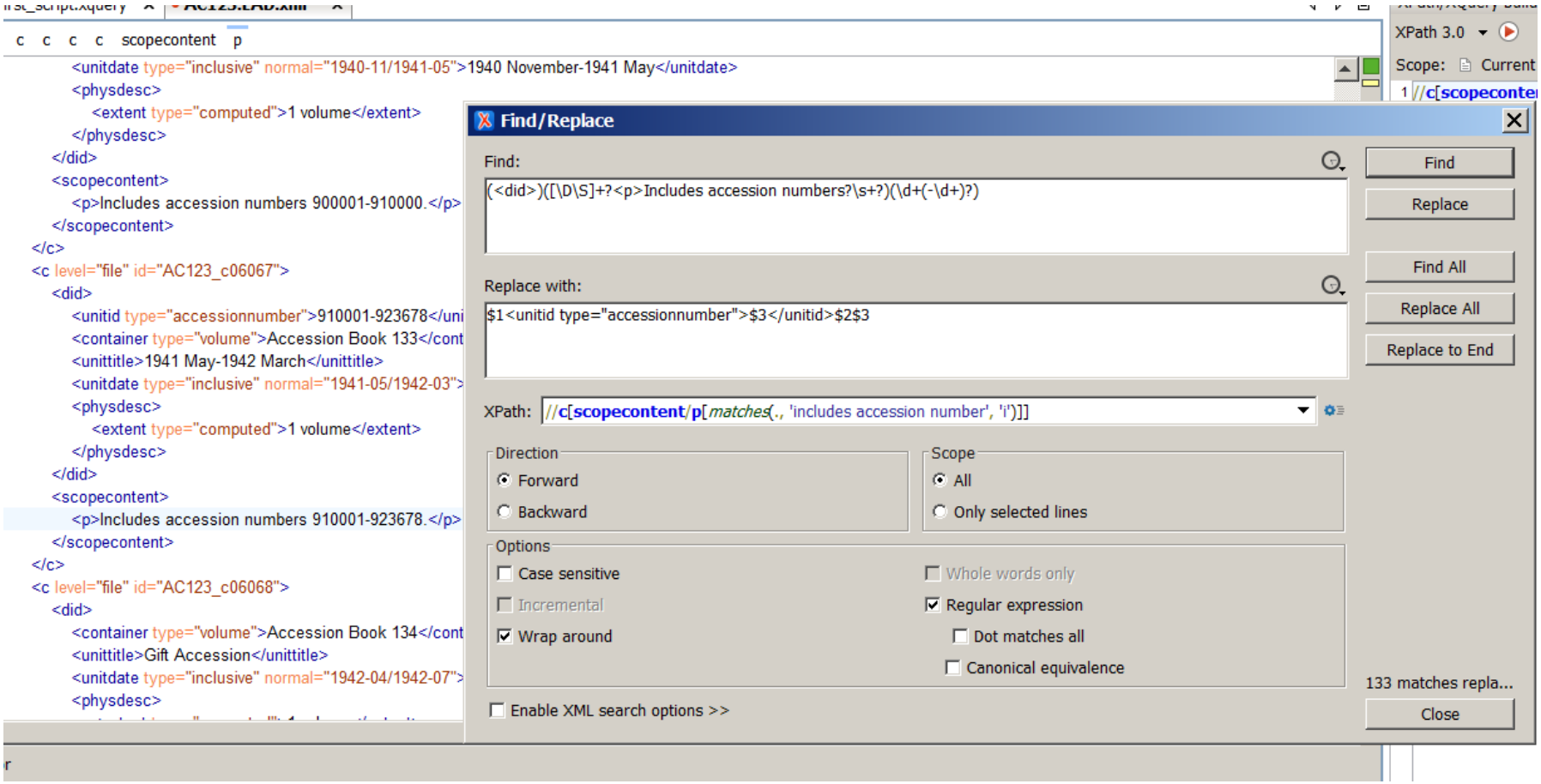
<did>[\D\S]+?<p>Includes accession numbers?\s+?\d+(-\d+)?



(<did>)
([\D\S]+?<p>Includes accession numbers?\s+?)
(\d+(-\d+)?)



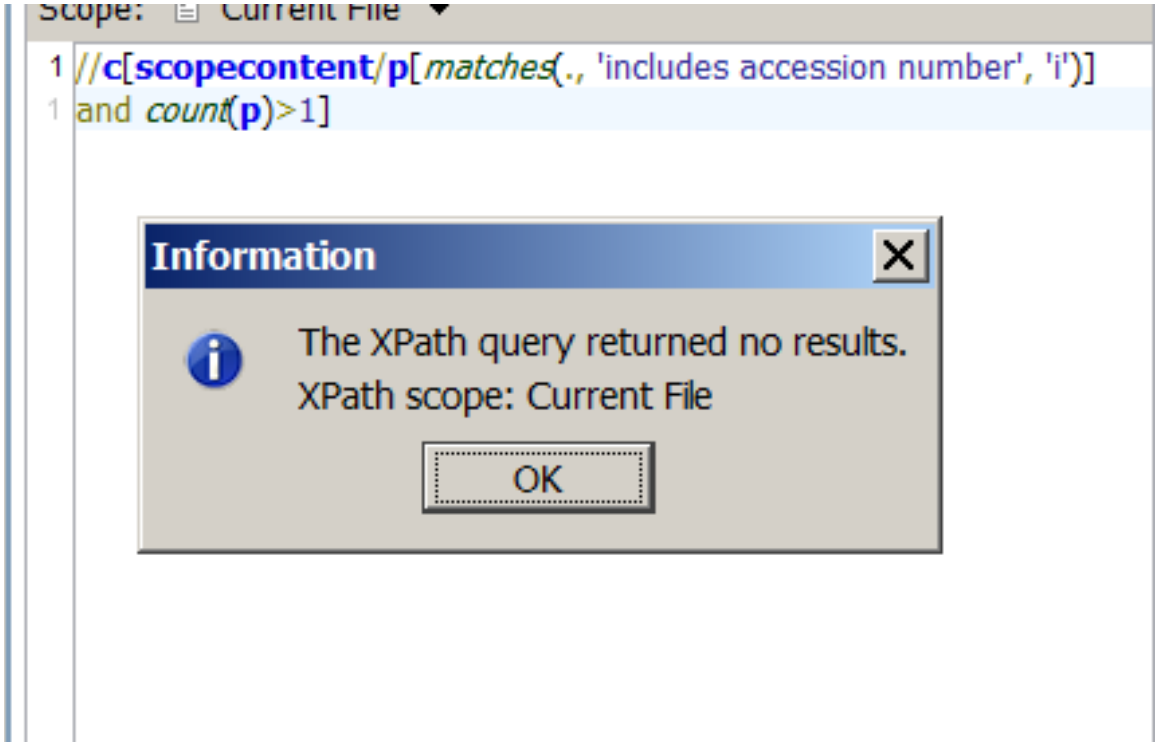
$1<unitid type="accessionnumber">$3</unitid>$2$3



Eh? ☺

```
<c level="file" id="AC123_c06066">
    <did>
        <unitid type="accessionnumber">900001-910000</unitid>
        <container type="volume">Accession Book 132</container>
        <unittitle>1940 November-1941 May</unittitle>
        <unitdate type="inclusive" normal="1940-11/1941-05">
                1940 November-1941 May
        </unitdate>
         <physdesc>
                <extent type="computed">1 volume</extent>
         </physdesc>
    </did>
    <scopecontent>
       <p>Includes accession numbers 900001-910000.</p>
    </scopecontent>
</c>
```

```
//c
[
        scopecontent/p
        [matches(., 'includes accession number', 'i')]
        and count(p)>1
]
```



c/scopecontent [p[matches(., 'includes accession number', 'i')]]

AC123.EAD.xml [C:\Users\heberlei\Downloads\AC123.EAD.xml] - <oXygen/> XML Editor (Academic use only)

File Edit Find Project Options Tools Document Window Help

XPath 3.0 //scopecontent/p[2]

AC387.xtr.xml | first_script.xquery | AC123.EAD.xml*

ead archdesc dsc c c c c scopecontent

```
68825          </physdesc>
68826        </did>
68827        <scopecontent>
68828          <p>Includes accession numbers 05001-10000.</p>
68829        </scopecontent>
68830      </c>
68831      <c level="file" id="AC123_c05936">
68832        <did>
68833          <unitid type="accessionnumber">10001-15000</unitid
68834          <container type="volume">Accession Book 3</contain
68835          <unittitle>undated</unittitle>
68836          <unitdate>undated</unitdate>
68837          <physdesc>
68838            <extent type="computed">1 volume</extent>
68839          </physdesc>
68840        </did>
68841        <scopecontent>
68842          <p>Includes accession numbers 10001-15000.</p>
68843        </scopecontent>
68844      </c>
68845      <c level="file" id="AC123_c05937">
68846        <did>
68847          <unitid type="accessionnumber">15001-20000</unitid
68848          <container type="volume">Accession Book 4</contain
68849          <unittitle>undated</unittitle>
68850          <unitdate>undated</unitdate>
68851          <physdesc>
68852            <extent type="computed">1 volume</extent>
68853          </physdesc>
```

Find/Replace

Find: [\D\S]+

Replace with:

XPath: //c/scopecontent[p[matches(., 'includes accession number', 'i')]]

Find | Replace | Find All | Replace All | Replace to End

Direction: Forward / Backward

Scope: All / Only selected lines

Options: Case sensitive; Incremental; Wrap around; Whole words only; Regular expression; Dot matches all; Canonical equivalence

Enable XML search options >>

133 matches found

Close

Text Grid Author

Description - 133 items

<scopecontent>	Includes accession numbers 00001-05000.	</scopecontent>
<scopecontent>	Includes accession numbers 05001-10000.	</scopecontent>
<scopecontent>	Includes accession numbers 10001-15000.	</scopecontent>
<scopecontent>	Includes accession numbers 15001-20000.	</scopecontent>
<scopecontent>	Includes accession numbers 20001-25000.	</scopecontent>

```
<c level="file" id="AC123_c06067">
    <did>
        <unitid type="accessionnumber">
            910001-923678
        </unitid>
        <container type="volume">
            Accession Book 133
        </container>
        <unittitle>1941 Mav-1942 March</unittitle>
        <unitdate type="inclusive"
                  normal="1941-05/1942-03">
                  1941 May-1942 March
        </unitdate>
        <physdesc>
            <extent type="computed">1 volume</extent>
        </physdesc>
    </did>
</c>
```

Challenge

Using the replace() function, which takes regex, retrieve the year of the photographs showing the production of The Rose Tattoo.

Hints:

1. How is your data patterned? Where is the title of the production, where is the format of the material?
2. Anchor your regex.
3. Once you have captured in your regex what you want to keep and what you want to discard, enclose both in parentheses; then replace your string with ‘$1’ (i.e., the first group)

Challenge Answer Key

- Notice that the title of the work is nested inside emph, whereas the format is in text()
- To get at the unitdate, which could be a sibling occuring before or after unittitle, step up to the parent, then down again to unitdate (to avoid using both the preceding-sibling and the following-sibling axis)

```
replace(//unittitle
        [matches(., 'photograph', 'i') and
        matches(emph, 'rose tattoo', 'i')]
        /../unitdate/@normal,
        '(^.{4})(.+$)',
        '$1')
```

→ Go to the unittitle that has an emph matching "rose tattoo" and a text node matching "photograph", then navigate to the @normal of its sibling unitdate. Replace the value of the @ with its first 4 characters.