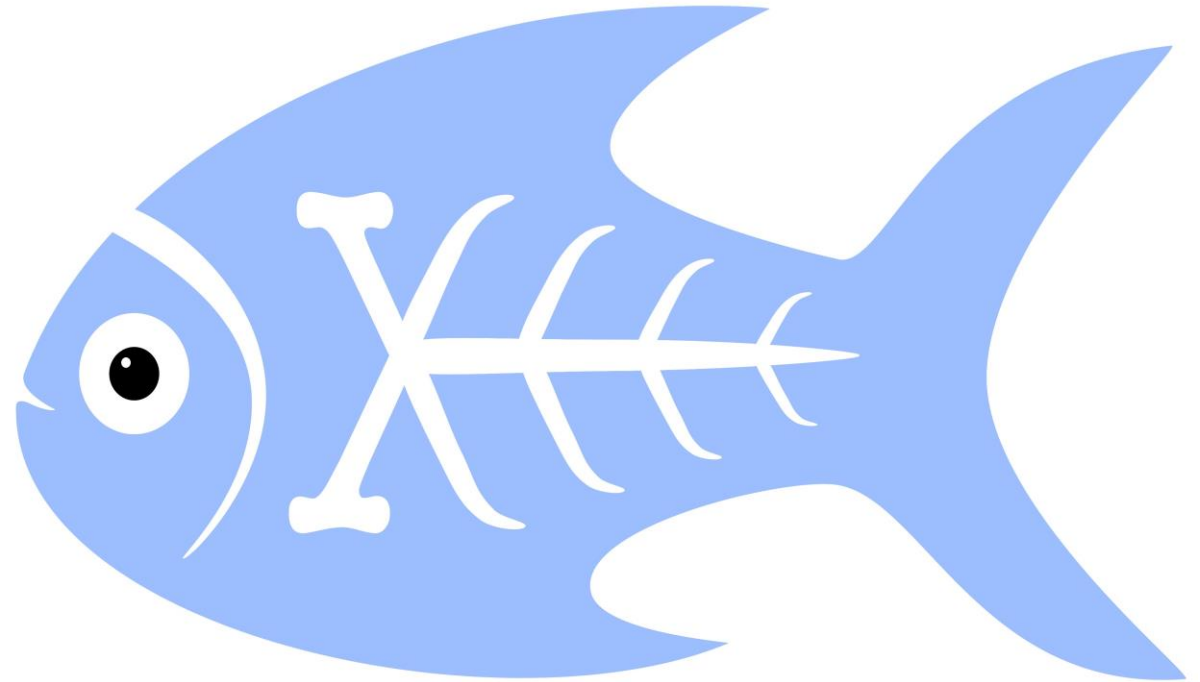


# Introduction to XProc 3.0 – Part 2

Markup UK 2020  
Webinar



GIT for this webinar: <https://github.com/xatapult/markupuk-2020>



# What have we learned in part 1?

- XProc is a *pipeline* language for documents, it chains *steps*
- Documents flow in and out of steps through *ports*
  - Documents can be XML, HTML, text, JSON or binary
- One input and one output port can be *primary*: These ports *implicitly* connect (unless *explicitly* connected)
  - Primary ports are called `source` and `result` by convention
- You can connect a port to:
  - Another port (either *implicit* for primary ports or *explicit*: `<p:pipe>` or `@pipe`)
  - To a document stated inline (`<p:inline>`)
  - To a document on disk (`<p:document>` or `@href`)
- *Options* are additional switches for the steps and/or your pipelines

And **most** important: I'm Kanava, the  
XProc logo!



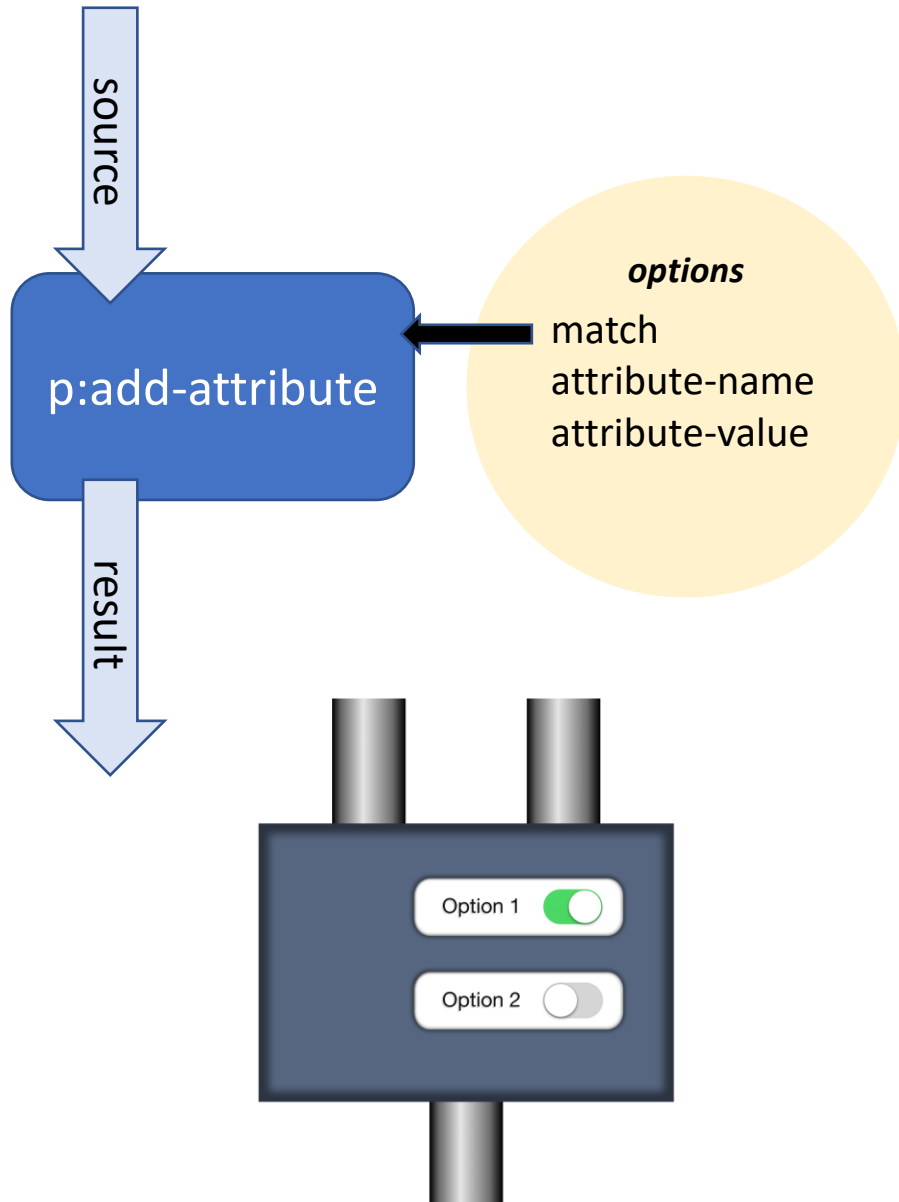
# What are we going to do today?

- Revisit options
  - Setting option with attributes or **<p:with-option>**
- Variables
  - How to create, how to use
- Core (or Compound) steps
  - Show example of multiple document handling using **<p:for-each>** and **<p:viewport>**

Sounds kinda interesting, tell me more!



# Options revisited



Set an option's value:

```
<p:add-attribute  
  match="/"*"  
  attribute-name="timestamp"  
  attribute-value="{current-dateTime()}" />
```

Declare an option for your own step:

```
<p:option name="author" select="'Erik'" />
```

Piece of cake, you've seen this  
last time...



# Setting option's values 1: Use attributes

```
<p:add-attribute match="/*" attribute-name="timestamp"  
  attribute-value="{current-dateTime()}" />
```

- Use AVTs (Attribute-Value Templates) {...} to insert XPath expressions
- Works only for:
  - Simple atomic values (strings, integers, booleans, etc.)
  - Single values (so no sequences!)
  - Map and array typed values, for instance:

```
<p:xslt parameters="map{ 'myparam' : 'myvalue' }" ...
```



# Setting option values 2: Use <p:with-option>

See: `markupuk-2020/101-B/example-1/example-1a.xpl`

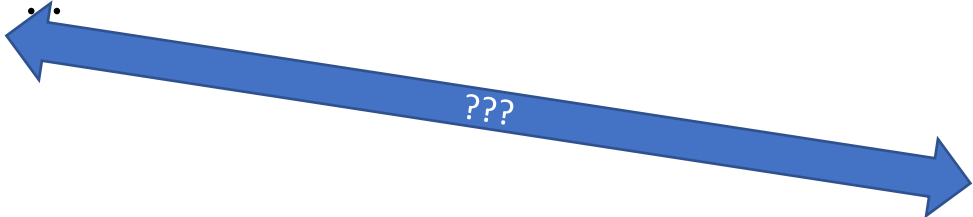
```
<p:add-attribute>  
  <p:with-option name="match" select="'/*'"/>  
  <p:with-option name="attribute-name" select="'timestamp'"/>  
  <p:with-option name="attribute-value" select="current-dateTime()"/>  
</p:add-attribute>
```

Much more verbose, but  
sometimes you can't do  
without...



# To quote or not to quote...

`<p:add-attribute match="/*" ..`



`<p:with-option name="match" select="'/*'"/>`

- The `match` option needs an XPath expression
  - Passed to `p:add-attribute` as a string
  - Processed *by the step itself*, not by the surrounding pipeline
- An attribute's value is just an atomic value (with the AVTs expanded)
- Any `select` attribute in a pipeline contains an XPath expression
  - This will be processed *by the pipeline*
  - The resulting *value* will be passed to the step



# Don't try this at home

See: `markupuk-2020/101-B/example-1/example-1b.xpl`



```
<p:add-attribute>
  <p:with-option name="match" select="/*"/>
  <p:with-option name="attribute-name" select="'timestamp'"/>
  <p:with-option name="attribute-value" select="current-dateTime()"/>
</p:add-attribute>
```

*No more quotes...*

- `/*` will be executed against the implicit connection
- Result passed as option's value: This is a piece of text to demonstrate ...
- The `match` option expects a valid XPath expression, which this is not...
- Result: Error...

\$!\$#@!...





# Some use-cases for `<p:with-option>`

- If you need to pass something more complex than a single atomic value:

See: `markup-2020/101-B/example-1/example-1c.xpl`

```
<p:directory-list path=".">  
  <p:with-option name="include-filter" select="('\.txt$', '\.xml$')"/>  
</p:directory-list>
```

- If you want to process your XPath expression against something floating out of another port:

```
<p:with-option name="attribute-value" select="/*/@status"  
  pipe="some-port@some-step"/>
```

It's getting complicated...  
Don't worry, most of the times setting options will  
be really simple!



# Variables

See: `markupuk-2020/101-B/example-2/example-2a.xpl`

- Declare a variable:

```
<p:variable name="id" select="..." />
```

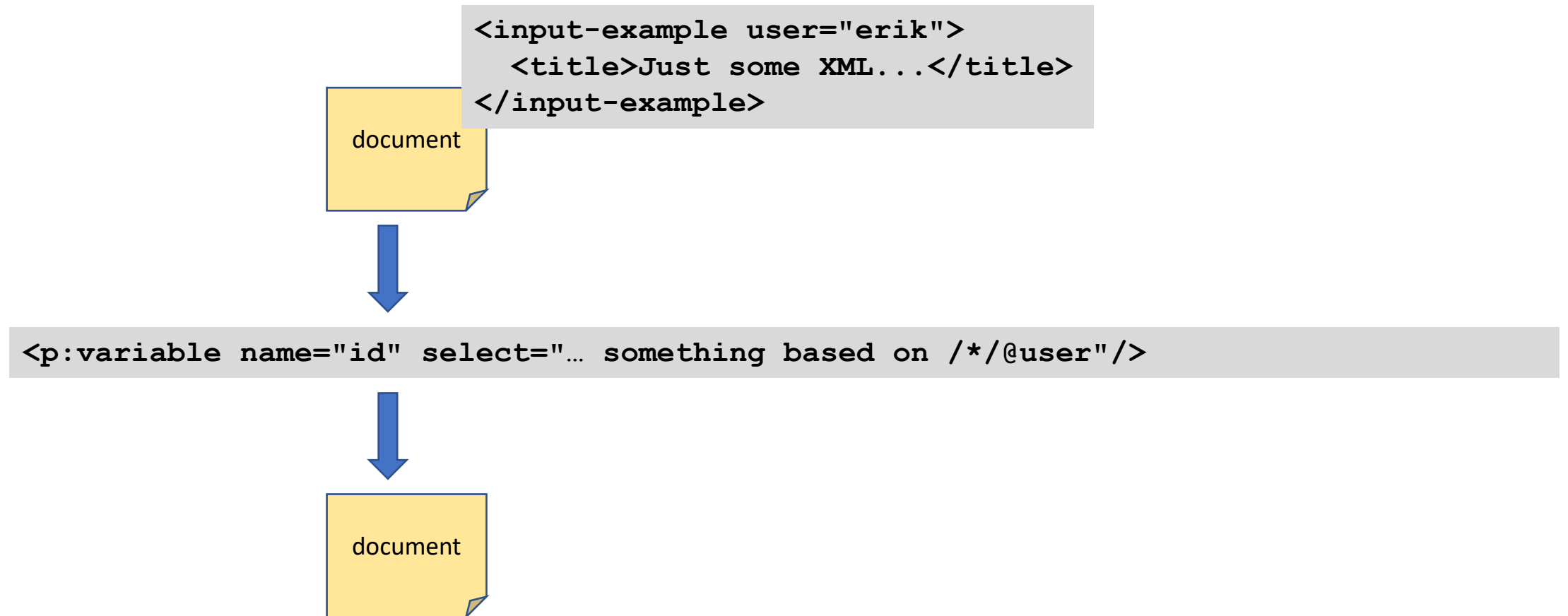
- Use a variable:

```
<p:add-attribute match="/*"  
  attribute-name="id" attribute-value="{ $id }" />
```



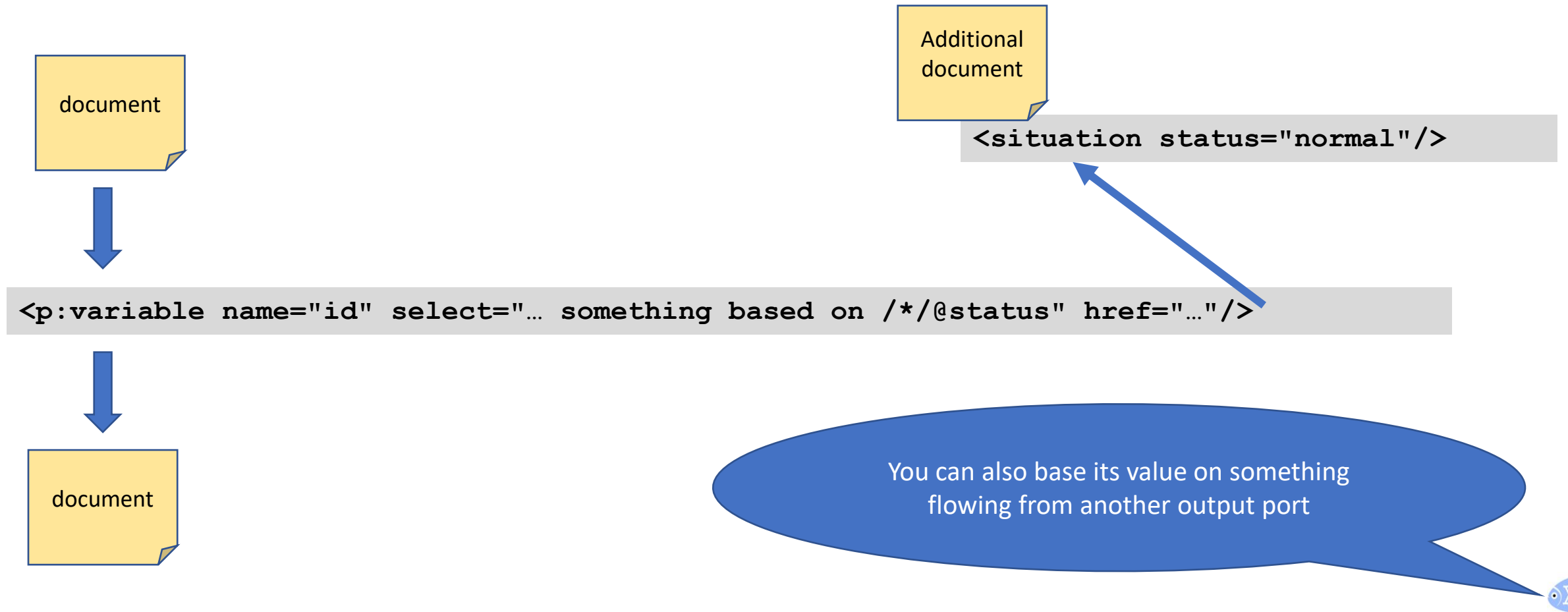
# Variables with values from document

See: `markupuk-2020/101-B/example-2/example-2b.xpl`



# Variables with values from *additional* document

See: `markupuk-2020/101-B/example-2/example-2c.xpl`



# The core (or compound) steps

- **p:for-each**: loop over multiple documents or parts of a document
- **p:choose / p:when / p:otherwise**: Make choices
- **p:if**: Make a single choice (there is no else)
- **p:viewport**: Work on only a part of a document
- **p:try / p:catch**: Error catching and handling
- **p:group**: Grouping of instructions

Regrettably, there is no time  
to look at them all...



# Use p:for-each to split a document

See: [markupuk-2020/101-B/example-3/example-3a.xpl](#), [example-3b.xpl](#) and [example-3c.xpl](#)

```
<documents>
```

```
  <doc filename="output1.xml">
```

```
    <contents>This is document number 1</contents>
```

```
  </doc>
```

```
  <doc filename="output2.xml">
```

```
    <contents>This is document number 2</contents>
```

```
    <more>It has some more...</more>
```

```
  </doc>
```

```
</documents>
```

Input  
document

Pipeline

output1.xml

output2.xml

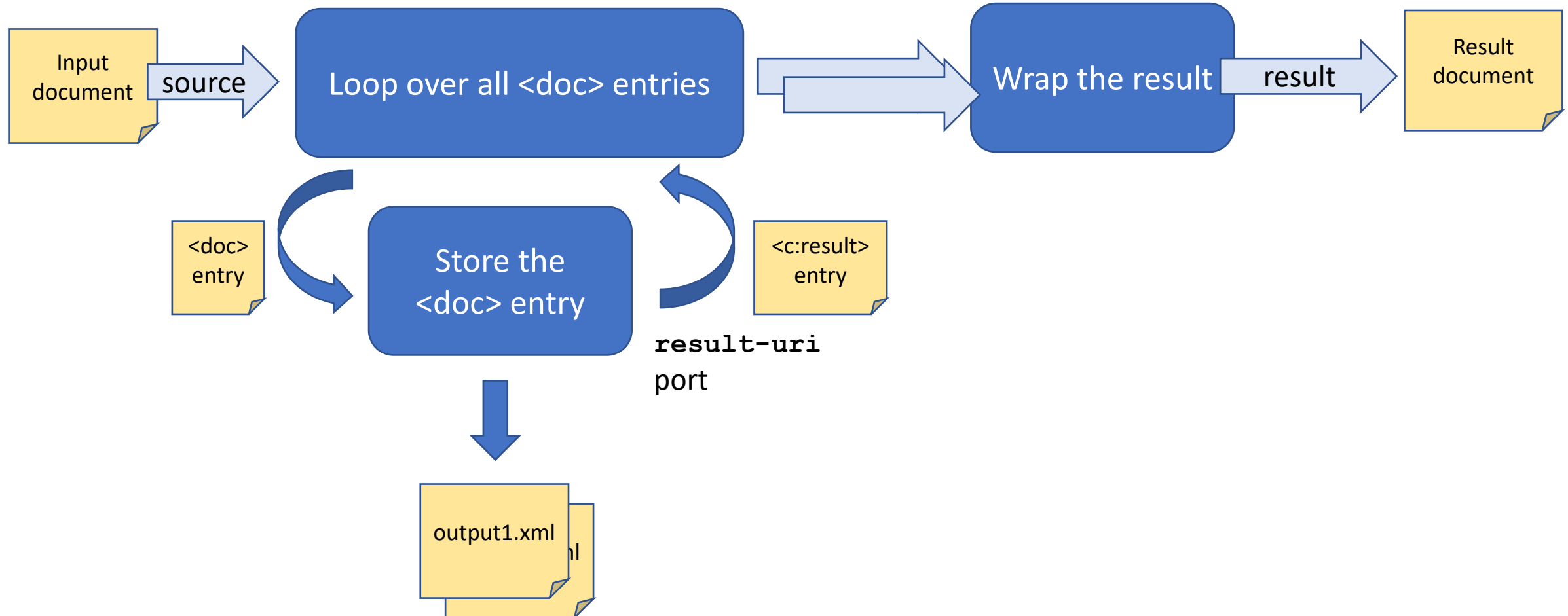
?

Boring contents Erik



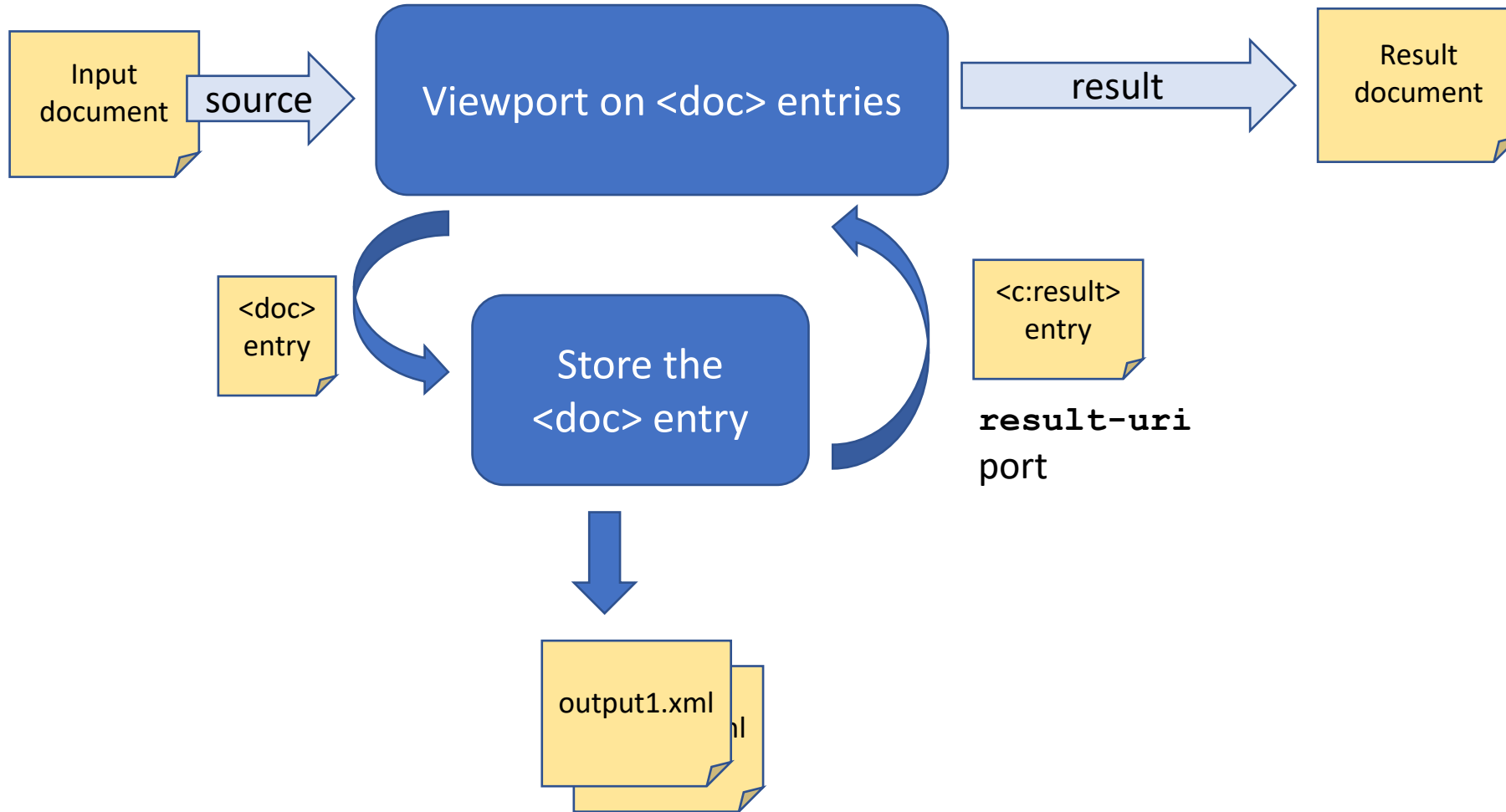
# Output the resulting filenames

See: `markupuk-2020/101-B/example-3/example-3d.xpl`



# Use p:viewport to split the document

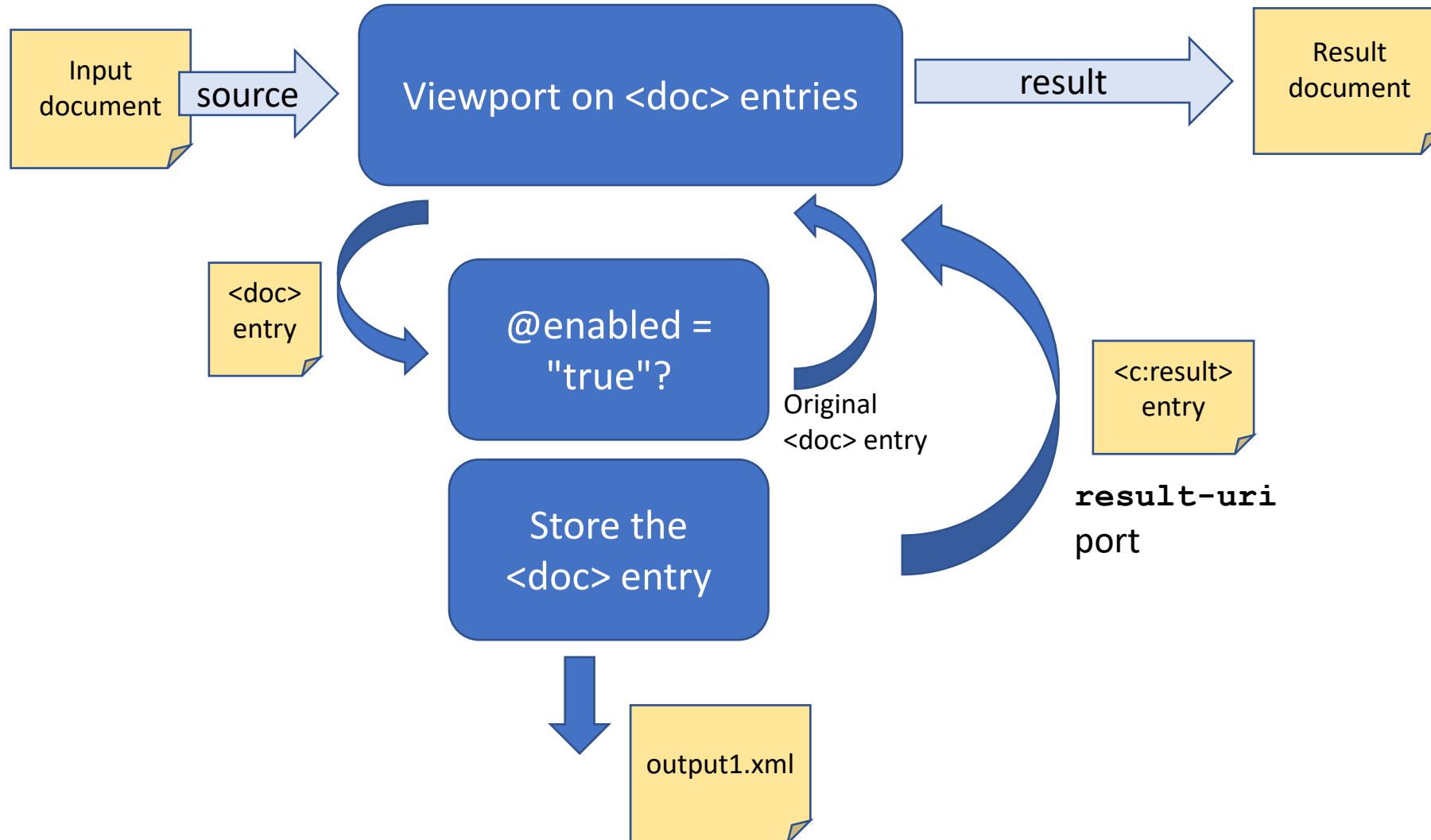
See: `markupuk-2020/101-B/example-3/example-3e.xpl`





# Only store files marked enabled

See: `markupuk-2020/101-B/example-3/example-3f.xpl`



# Wrap up:

- You can set options by attribute or using `<p:with-option>`
  - Watch out: Who is going to interpret the XPath expressions?
- You can define and use variables
  - XPath expression do not have to be based on the document flowing through
- There are core steps for looping, decision making, etc.

We looked at:

- `p:for-each`
- `p:viewport`
- `p:if`



# Goodbye and thank the fish, again!

Your guide today: Erik Siegel – [erik@xatapult.nl](mailto:erik@xatapult.nl)

Specification: <https://spec.xproc.org/>

Processors:

- Morgana: <https://www.xml-project.com/>
- Calabash: <https://xmlcalabash.com/>

Articles on XProc: <https://www.xml.com>

Book: <https://xmlpress.net/publications/xproc-3-0/>

See you!  
And remember,  
Kanava says:  
***XProc rocks...***



# Who Am I?

- Erik Siegel
- Content Engineer, XML Specialist, Technical Writer
- Company: Xatapult
  - Groningen, The Netherlands
  - Customers mostly in publishing and standardization
- Part of the XProc 3.0 editing committee
- Writer of the XProc 3.0 Programmer Reference
- Contact:

[erik@xatapult.nl](mailto:erik@xatapult.nl)

[www.xatapult.com](http://www.xatapult.com)

[www.linkedin.com/in/esiegel/](https://www.linkedin.com/in/esiegel/)

+31 6 53260792

