DIE ÖSTERREICHISCHE BIBLIOTHEKENVERBUND
UND SERVICE GMBH

obv sg

# USING XSLT FOR NORMALIZATION IN ALMA
## THE OBVSG-WORKFLOW

IGELU, 18.19.2025

STEFAN SCHUH

# AGENDA



- Why XSLT?
- Our Workflow
- Samples

# Why XSLT?

# WHY NOT DROOLS?

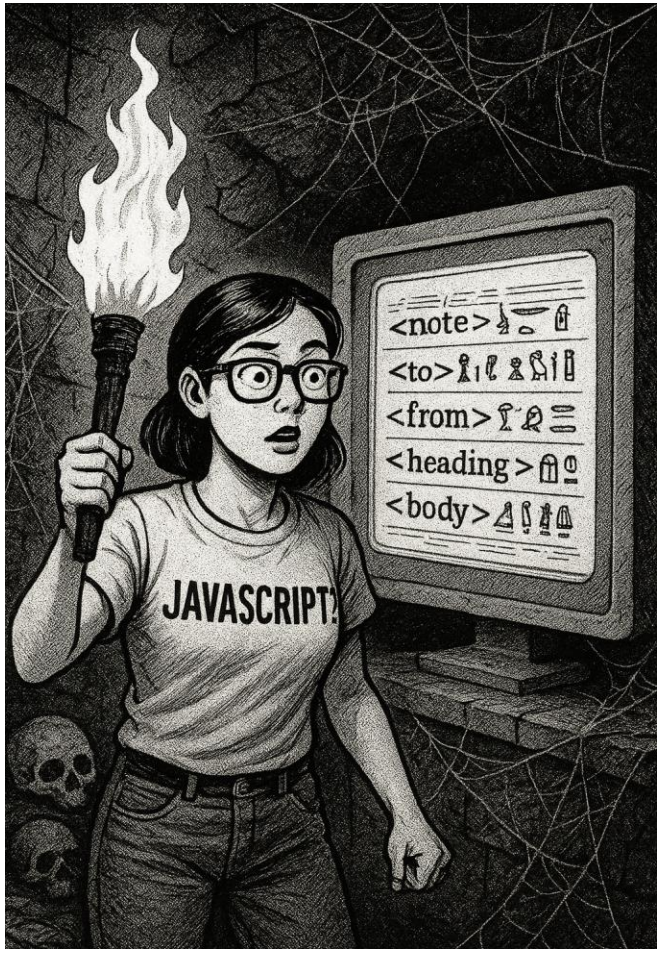
## THE SYNTAX WOULD BE WAY SIMPLER ...

- Mutation is bad.
- Order of operations is significant but opaque
- Funky undocumented behaviour
- Doesn't scale well at all
- Mutation is bad.
- No automated testing possible
- Unless you are a Ninja Turtle
- Leads to confusion and unexpected results
- Looking at you, OR operator … (#06569291)
- Because all of the above
- Still.
- Did something mutate on the way? We'll never know

# WHY XSLT?



The styntax, obviously.

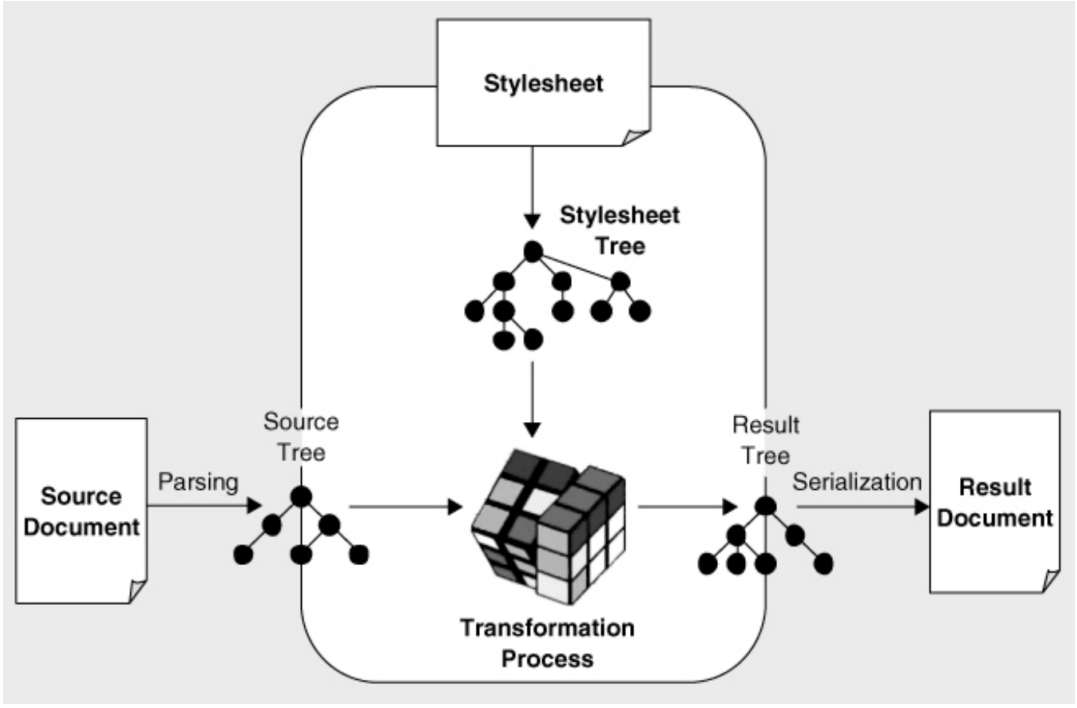


# WHY XSLT?



Aaaaand: The processing model!

- Great specification
- Fully functional, declarative
- Immutable data structures
- Input and output tree are separate
- Order of declarations doesn't matter
- Match and apply processing
- Editing support
- Testability outside of Alma



## WHY XSLT?
## MATCH AND APPLY

Describe desired output

Look for matching input

```
<xsl:template match="datafield[@tag='500']
                     [contains(subfield[@code='a'], 'Univ., Diss.')]">
  <datafield tag="502" ind1=" " ind2=" ">
    <subfield code="a">{subfield[@code="a"]}</subfield>
  </datafield>
</xsl:template>
```

# Our Workflow

# OUR WORKFLOW



- Write XSLT-stylesheets
- Write and run tests
- Bundle included stylesheets into one single stylesheet
- Copy/Paste into Alma
- Generate documentation
- Deploy documentation

- Emacs / VS Code
- XSpec (in the GitLab-CI)
- `xslt-bundler` (in the GitLab-CI)
- Ctrl-C/Ctrl-V
- `xslt-documentation-generator` (GitLab-CI)
- `rsync` (in the GitLab-CI)

# WRITING XSLT CODE



- Using version 3.0 because
  - Text-value-templates
  - Arrow operator (=>)
  - Simple map operator (!)
  - Maps and arrays
- Document each template or function in a comment, directly in the source
- Split large stylesheets into several files
- Automated testing using XSpec

- More expressive, maybe more readable code than 2.0
- Generate HTML-documentation for users to read
- Stay organized
- Boring but vital

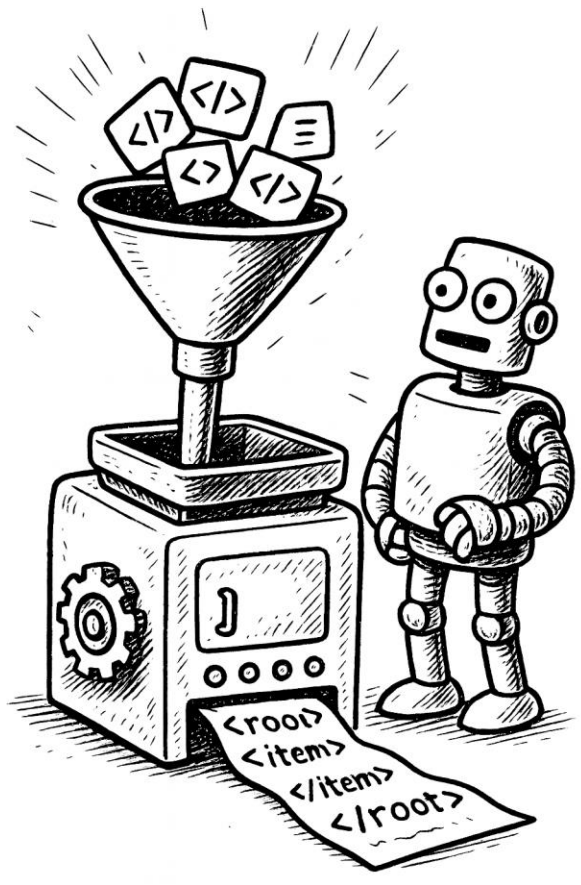
# BUNDLE STYLESHEETS



- Bundle included complex stylesheets into one single file

- `xslt-bundler` is an XSLT-Stylesheet that
  - Gets all code included by `<xsl:include>` declarations
  - Gets all data bound to variables with `doc()`
  - Writes all that into one physical file
  - Prepares the XSpec tests to be run against the bundled stylesheet

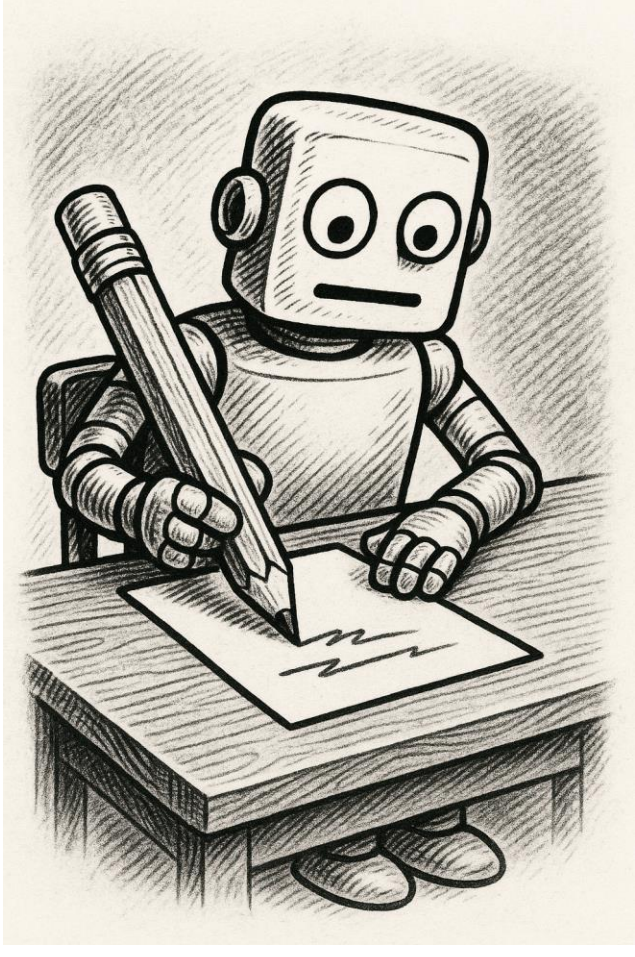
- Runs in our GitLab-CI



# GENERATE DOCUMENTATION



- Stylesheets (all it's includes) are read and analyzed
  - Structured comments preceding elements
  - Parameters and return types
  - Calls to functions and templates
  - Backreferences (functions and templates calling the element in question)
  - …
- Generate intermediary XML from this analysis
- Render HTML using the intermediary XML
  - Using Pico CSS as a minimal framework.
    - Classless – only semantic HTML
  - Markdown rendering using md-block
  - Syntax highlighting using Prism
- Deploy it
  - https://share.obvsg.at/xsldocs/



# Examples

## EXAMPLES

### REMOVE ISBD PUNCTUATION WITH DROOLS

```
rule "EXTR-077-rm505DESCR: delete description marks in MARC 505"
    when
        exists "505"
    then

        suffix "505.t" with "isbd" if (exists "505.t.*;")
        replaceContents "505.t.;isbd" with ""
        suffix "505.t" with "isbd" if (exists "505.t.*\\\\.")
        replaceContents "505.t.\\\\.isbd" with ""
        suffix "505.t" with "isbd" if (exists "505.t.*/")
        replaceContents "505.t./isbd" with ""
        suffix "505.t" with "isbd" if (exists "505.t.*--")
        replaceContents "505.t.--isbd" with ""
        replaceContents "505.t.isbd" with ""
```

## EXAMPLES

# REMOVE ISBD PUNCTUATION WITH XSLT

```
<xsl:function name="mrclib:remove-isbd" as="xs:string">
  <xsl:param name="s" as="xs:string" />

  <xsl:variable name="isbdChars" select="('.', ',', ':', ';', '/')" />
  <xsl:choose>
    <!-- Account for initials, don't remove abbreviation dot. -->
    <xsl:when test="matches($s, '\W[A-Z]\.$')">
      <xsl:value-of select="$s" />
    </xsl:when>
    <xsl:when test="substring($s, string-length($s)) = $isbdChars">
      <xsl:value-of select="substring($s, 1, string-length($s) - 1) => normalize-space()" />
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$s" />
    </xsl:otherwise>
  </xsl:choose>
</xsl:function>
```

## EXAMPLES

# COPY LANGUAGE CODE FROM 041 TO 008/35-37 WITH DROOLS

Instead of this:

```
rule "KATA-002-SPR008-02: copy first language code from 041 SF-a to 008 Pos. 35-37 - ger"
     when
        ((existsControl "008.{35,3}.   "
          or existsControl "008.{35,3}.###"
          or existsControl "008.{35,3}.|||")
         and exists "041.X.ger")
    then
        replaceControlContents "008.{35,3}.   " with "ger"
        replaceControlContents "008.{35,3}.   " with "ger"
end
```

For every language code ...

## EXAMPLES

# COPY LANGUAGE CODE FROM 041 TO 008/35-37 WITH XSLT

That:

```
<xsl:template match="controlfield[@tag='008']">
  <xsl:variable name="firstLang041a"
                select="../datafield[@tag='041'][1]/subfield[@code='a'][1]" />
  <controlfield tag='008'>{
    if (if $firstLang041a and matches(substring(., 36, 3), '   |###|\|\|\|'))
    then
      mrclib:replace-control-substring(., 35, 37, $firstLang041a)
    else .
  }</controlfield>
</xsl:template>
```

Once.

[Documentation for mrclib:replace-control-substring()]

# Conclusion

# CONCLUSION

## WHAT WE HAVE UNTIL NOW

- What we have working
  - CI-Workflow for bundling XSLT source and generating documentation
  - Ad hoc data corrections
  - Some normalizations (derive article, generate human readable coordinates in MARC 255 etc.)

- What remains to be done
  - Migrate all drools to XSLT
    - Will be done incrementally
    - XSLT before drools
    - Swap rules out one by one (if possible)

# CONCLUSION
## WHAT WE WOULD WISH FOR

- Deployment via API
  - Copying/Pasting is not a very good workflow
  - Browser automation maybe?

- Access to Metadata outside the MARC record
  - Agency information, i. e. what `addModifyingAgency` does in drools
  - Could be passed as stylesheet parameter, we hope Ex Libris will look into this

# CONCLUSION



Are we happy?

Yes. Very.

# THANKS!

Thank you for hearing me!

Questions?

stefan.schuh@obvsg.at

# IMAGE CREDITS



- Slide 6: Kay, Michael. *XSLT 2.0 and XPath 2.0 : Programmer's Reference*. 4. ed., 2008
- All the others: ChatGPT image generator