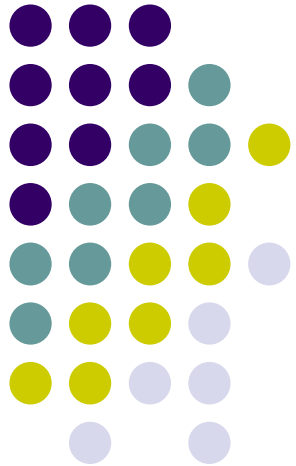
RDFe

Expression based translation
XML to RDF

Hans-Jürgen Rennau, parsQube GmbH, 2019-02-09

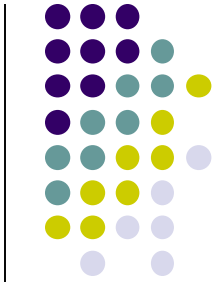


painters.xml

```
<painters>
  <painter>
    <name>
      <familyName>Magritte</familyName>
      <givenName>Rene</givenName>
    </name>
    <paintings>
      <painting>
        <title>Clairvoyance</title>
        <date>1936</date>
      </painting>
    </paintings>
  </painter>
</painters>
```

paintings.xml

```
<paintings>
  <painting>
    <title>Clairvoyance</title>
    <date>1936</date>
    <painter>
        <name>Magritte, Rene</name>
    </painter>
  </painting>
</paintings>
```



painter-zh.xml

```
<画家们>
  <画家>
    <姓名>
      <姓氏>Magritte</姓氏>
      <名字>Rene</名字>
    </姓名>
    <画作集>
      <画作>
        <标题>Clairvoyance</标题>
        <日期>1936</日期>
      </画作>
    </画作集>
  </画家>
</画家们>
```

culture.ttl

```
@prefix artist:   <http://example.com/resource/artist/> .
@prefix painting: <http://example.com/resource/opus/> .
@prefix cult:     <http://example.com/ontologies/culture/> .
@prefix rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
```

```
artist:1    rdf:type          cult:artist ;
            cult:lastName     "Magritte" ;
            cult:firstName    "Rene" ;
            cult:created      painting:1 .

painting:1  rdf:type          cult:painting ;
            cult:inception    1936 ;
            cult:title        "Clairvoyance" ;
            cult:createdBy    artist:1 .
```

Semantic content & tree shapes

painters.xml ✓ paintings.xml ✗ painters-zh.xml

Semantic content

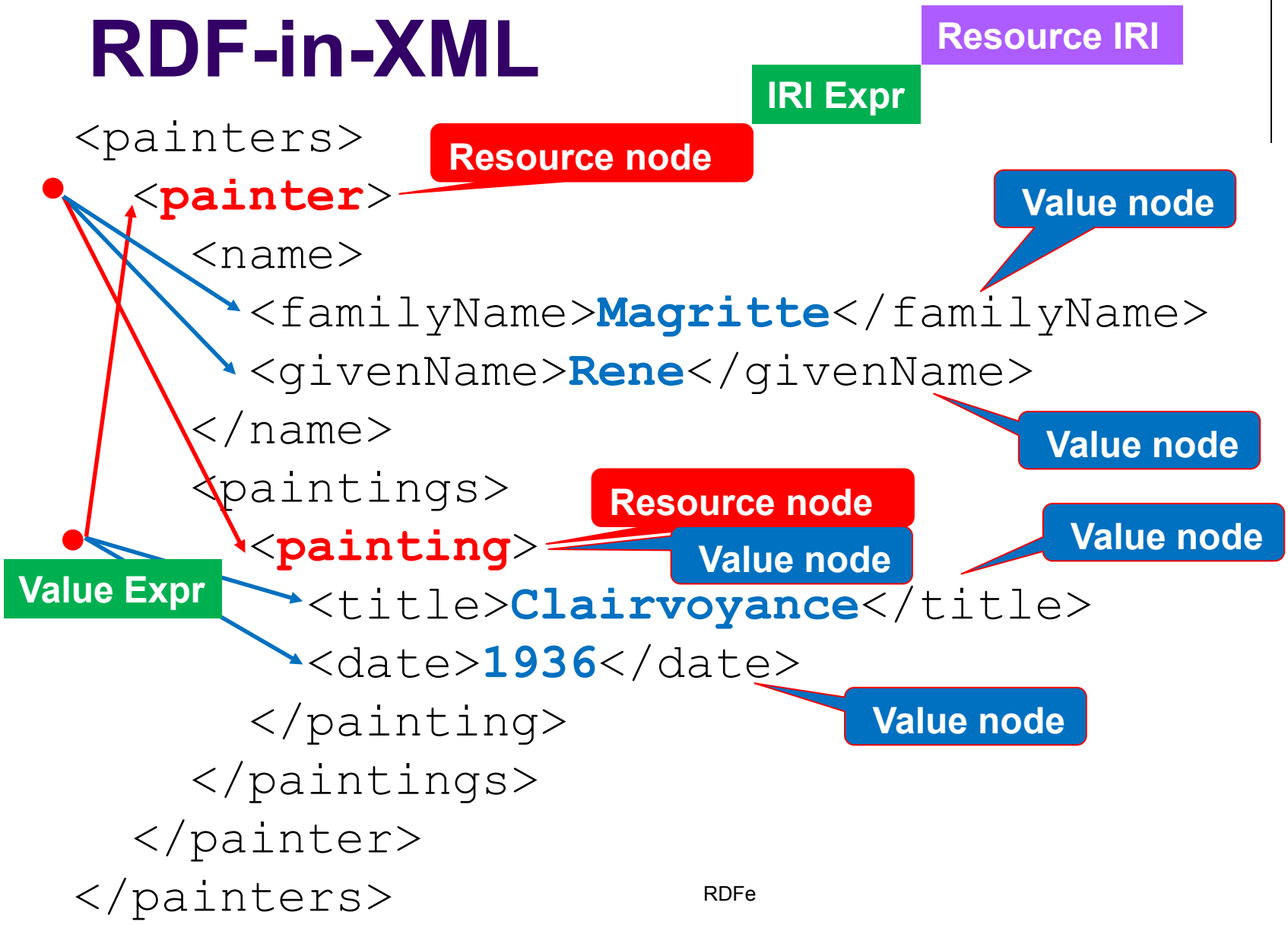
Tree shape

RDFe
RDFa
?

culture.ttl

GraphQL
?
?

RDF-in-XML


Resource IRI
IRI Expr
Resource node
Value node
Value node
Resource node
Value node
Value node
Value Expr
Value node
<painters>
<painter>
<name>
<familyName>Magritte</familyName>
<givenName>Rene</givenName>
</name>
<paintings>
<painting>
<title>Clairvoyance</title>
<date>1936</date>
</painting>
</paintings>
</painter>
</painters>

Key concepts

- Resource nodes
- Value nodes
- Value expression
- IRI expression

Mapping

Property IRI

semantic relationship: subject – object

=>

structural relationship: resourceNode – valueNode

Value Expr

Let RDF pull …

```
<semanticMap>
  <resource type="cult:artist"
      targetNodeName="painter"
      iri="?" >
   <property iri="cult:lastName"  value="?" />
   <property iri="cult:firstName" value="?" />
   <property iri="cult:created"   value="?" />
  </resource>a
  <resource type="cult:painting"
      targetNodeName="painting"
      iri="?" >
   <property iri="cult:inception" value="?" />
   <property iri="cult:title"     value="?" />
   <property iri="cult:createdBy" value="?" />
  </resource>
</semanticMap >
```

… pull *the strings of XPath!*

```
<semanticMap>
  <resource type="cult:artist"
      targetNodeName="painter"
      iri="'artist:' || 1 + count(preceding-sibling::painter)">
   <property iri="cult:lastName"  value="name/familyName" />
   <property iri="cult:firstName" value="name/givenName" />
   <property iri="cult:created"   value="painters/painting" />
  </resource>
  <resource type="cult:painting"
      targetNodeName="painting"
      iri="'painting:' || 1 + count(preceding::painting)" />
   <property iri="cult:inception" value="date" />
   <property iri="cult:title"     value="title" />
   <property iri="cult:createdBy" value="ancestor::painter" />
  </resource>
</semanticMap >
```

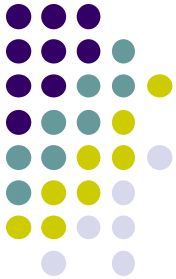
Semantic map - outline

```
<re:semanticMap  iri="http://example.com/semap/painters/"
                 targetName="painters"
                 targetNamespace=""
                 xmlns:re="http://www.rdfe.org/ns/model" …>

  <re:namespace iri="http://example.com/resource/artist/"       prefix="artist"/>
  <re:namespace iri="http://example.com/resource/opus/"         prefix="painting"/>
  <re:namespace iri="http://example.com/ontologies/culture/"    prefix="cult"/>

  <re:resource targetNodeName="painter" …>…</resource>
  <re:resource targetNodeName="painting" …>…</resource>

</re:semanticMap>
```

Target constraint

Resource models

Resource model (for <painter>)

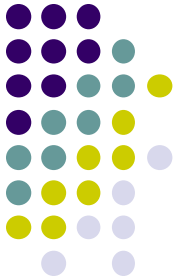


```
…
<re:resource
    targetNodeName              ="painter"
    assertedTargetNodes         ="//painter"
    iri                         ="'artist:'||1 + count(preceding-sibling::painter)"
    type                        ="cult:artist"
    modelID                     ="artist">

    <re:property iri="cult:lastName"
                 value="name/familyName"/>
    <re:property iri="cult:firstName"
                 value="name/givenName"/>
    <re:property iri="cult:created"
                 value="paintings/painting"
                 type="#resource"/>

</re:resource>
…
```

Target node constraint

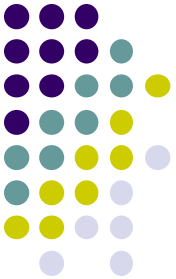
Value expressions

Resource model (for <painting>)



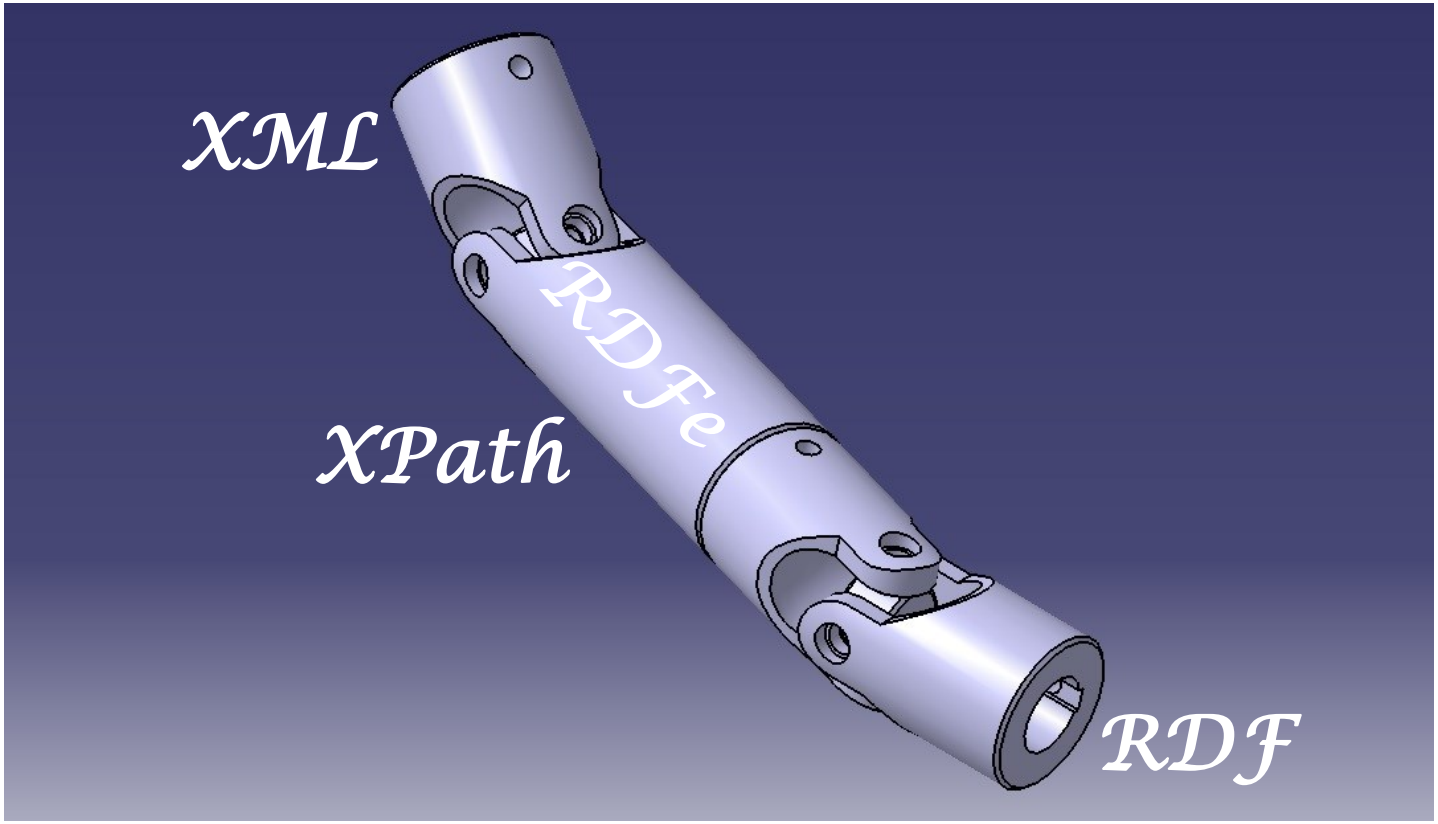
```
…
<re:resource targetNodeName="painter" …>…</re:resource>
<re:resource
    targetNodeName          ="painting"
    assertedTargetNodes     ="//painting"
    iri                     ="'painting:' || 1 + count(preceding::painting)"
    type                    ="cult:painting"
    modelID                 ="painting">

    <re:property iri="cult:inception"
    <re:property iri="cult:title"
    <re:property iri="cult:createdBy"

                 value="date" type="xs:integer"/>
                 value="title"/>
                 value="ancestor::painter"
                                type="#resource"/>

</re:resource>
…
```

XPath – the moving part *(path)* betwiXt XML and RDF

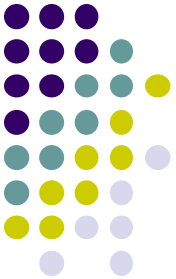


Resource type: cult:painter			
Resources Property IRI	XPath		
	painters.xml //painter	paintings.xml //painter	painters-zh.xml // 画家
cult:lastName	name/familyName	name/replace(., ',.*', '')	姓名 / 姓氏
cult:firstName	name/givenName	name/replace(., '.*?,\s*', '')	姓名 / 名字
cult:created	paintings/painting	ancestor::painting	画作集 / 画作
Resource type: cult:painting			
Resources Property IRI	XPath		
	painters.xml //painting	paintings.xml //painting	painters-zh.xml // 画作
cult:inception	date	date	日期
cult:title	title	title	标题
cult:createdBy	ancestor::painter	painter	ancestor::画家

XPath – the moving part *(path)* betwiXt XML and RDF



RDFe processing model



- Phase 1:

  Create ***asserted*** **resource descriptions**

```
<re:resource assertedTargetNodes="//painter"/>
```

- Phase 2:

  Create ***required*** **resource descriptions**

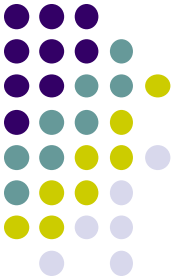
```
<re:property iri="…" value="…" type="#resource"/>
```

```
for $rnode in eval(@value)  [@type = "#resource"]
                            [resourceNotYetDescribed()]:
    $rmodel = rmodel-for-rnode ($rnode) return
    rnode-to-rdescription ( $rnode, $rmodel )
```

Find r. model

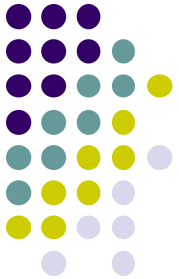
Apply it!

RDFe – advanced features



- Imports
- Evaluation context
- Navigation across document boundaries
- Document discovery
- Conditional properties
- @list, @reversed, @inverseIri
- @card

Voilà, a catalog of paintings …



```
<paintings>
    …
    <painting ID="Q3506878">
        <createrID>Q7836</createrID>
        <title>Clairvoyance</title>
        <date>1936</date>
        <movement>Surrealism</movement>
        <genre>self-portrait</genre>
        <materialsUsed>oil paint, canvas</materialsUsed>
        <collection>private collection</collection>
        <depicts>egg, table, easel, bird, palette,
                 paint brush, tablecloth, chair, man</depicts>
    </painting>
    …
</paintings>
```

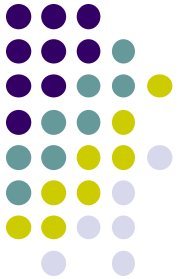
… with a semantic map

```
<re:semanticMap …>
   <re:namespace iri="https://www.wikidata.org/wiki/" prefix="wiki"/>
   <re:resource
      targetNodeName="painting"
      assertedTargetNodes=""
      iri="'wiki:' || @ID"
      type="wiki:painting"
      modelID="painting">
   <re:property iri="wiki:createdBy"          value="'wiki:' || createrID" type="#iri"/>
   <re:property iri="wiki:inception"          value="date" type="xs:integer"/>
   <re:property iri="wiki:title"              value="title"/>
   <re:property iri="wiki:genre"              value="genre"/>
   <re:property iri="wiki:movement"           value="movement"/>
   <re:property iri="wiki:materialUsed"       value="tokenize(materialsUsed, ',\s*')"/>
   <re:property iri="wiki:depicts"            value="tokenize(depicts, ',\s*')"/>
  </re:resource>
</re:semanticMap>
```

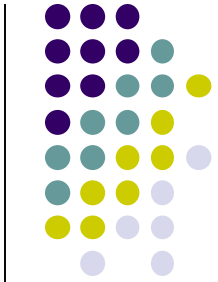
Let‘s pull it in!

```
<re:semanticMap targetName="painters" …>
  <re:import href="paintings.rdfe.xml"/>
  …
 <re:resource targetNodeName="painter" …>
   …
   <re:property    iri          ="cult:created"
                   inverseIri   ="cult:createdBy„
                   type         ="#resource"
                   value=        " paintings/painting
                                  for $p in paintings/painting return
                                    doc('paintings-catalog.xml')
                                    //painting[title = $p/title] [date = $p/date] "
   />
 </re:resource>
   …
</re:semanticMap>
```

```
@prefix cult: <http://example.com/ontologies/culture/> .
…
artist:1 rdf:type cult:artist ;
        cult:lastName "Magritte" ;
        cult:firstName "Rene" ;
        cult:created painting:1 .
painting:1 cult:createdBy artist:1 ;
          rdf:type cult:painting ;
          cult:inception 1936 ;
          cult:title "Clairvoyance" .
```



```
@prefix wikidata: <https://www.wikidata.org/wiki/> .
…
artist:1 rdf:type cult:artist ;
        cult:lastName "Magritte" ;
        cult:firstName "Rene" ;
        cult:created wiki:Q3506878 .
wiki:Q3506878 cult:createdBy artist:1 ;
            rdf:type cult:opus ;
            wiki:createdBy wiki:Q7836 ;
            wiki:inception 1936 ;
            wiki:title "Clairvoyance" ;
            wiki:genre "self-portrait" ;
            wiki:movement "Surrealism" ;
            wiki:materialUsed "oil paint" ;
            wiki:materialUsed "canvas" ;
            wiki:depicts "egg" ;
            wiki:depicts "table" ;
            wiki:depicts "easel" ;
            wiki:depicts "bird" ;
            wiki:depicts "palette" ;
            wiki:depicts "paint brush" ;
            wiki:depicts "tablecloth" ;
            wiki:depicts "chair" ;
            wiki:depicts "man" .
```



Better use a context

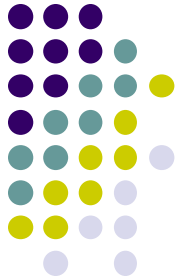
```
<re:semanticMap targetName="painters" …>
   …
   <re:context>
      <re:var name="uriPaintings" value="'paintings-catalog.xml'"/>
      <re:var name="docPaintings"
               value="doc(resolve-uri($uriPaintings, base-uri()))"/>
      <re:fun name="getPainting" params="title, date" as="element(painting)"
               code=" $docPaintings//painting [title = $title] [date = $date]"/>
   </re:context>
   …
  <re:resource targetNodeName="painter" …>
   …
   <re:property    iri          ="cult:created"
                   inverseIri   ="cult:createdBy"
                   type         ="#resource"
                   value        ="paintings/painting/$getPainting(title, date)"
   />
  </re:resource>
</re:semanticMap>
```

Conditional settings

```
<re:resource targetNodeName="painter" …>
  …
  <re:property   iri=          "cult:created"
                 inverseIri=   "cult:createdBy">
                 type=         "#resource"
                 value=        "paintings/painting"

      <re:valueItemCase
                 test=         "$rdfe:valueItem/@wikidataID"
                 value=        "$docPaintings//painting
                                  [@ID = $rdfe:valueItem/@wikidataID]"/>

</re:property>
```

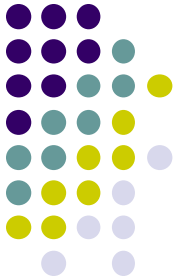
If the item returned by **@value** passes **@test**:
resume navigation & dive into the catalog document.
Else
use paintings/painting

RDFa - RDFe



- *RDF**a***
  - Translation control: embedded **a**ttributes $\mathcal{A}$
  - Preferred target XML:
    - Text oriented, human authored
    - Markup layout oriented (e.g. section, para, title, …)
- *RDF**e***
  - Translation control: external **e**xpressions $\mathcal{E}$
  - Preferred target:
    - Date oriented, machine generated
    - Markup semantic (e.g. painter, protein, price, …)

RDFe – main goals



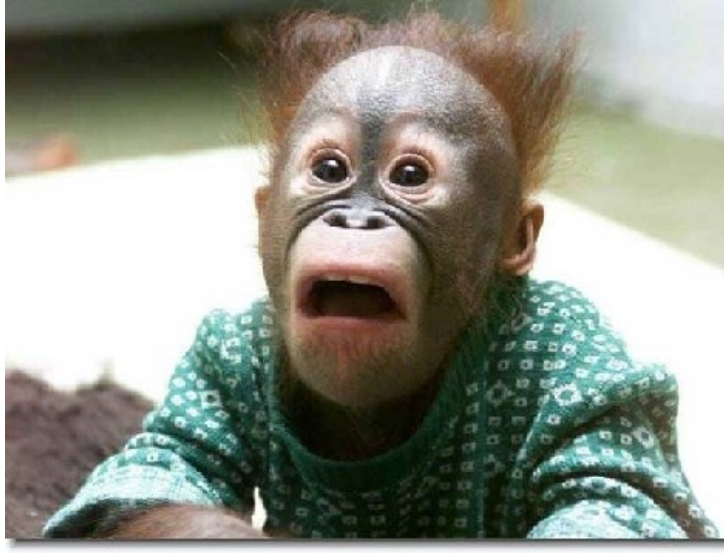
- Cope with **any XML** data

  Map XML of any complexity and any „distance“
  from the RDF model

- Cope with **Linked Data** scenarios

  Map & connect the contents of multiple documents
  with multiple document types,
  discovered iteratively.

RDFe – issues



Performance.

The reference implementation is naive.

https://github.com/hrennau/shax

Ideal solution:

RDFe support built into XQuery processor

Merci !