Co-funded by the Horizon 2020
Framework Programme of the European Union
Grant Agreement Number 644771

XLIFF 2.0 AND ENRICHMENT WORKFLOWS IN THE BROWSER

FEISGILTT 2016, DUBLIN

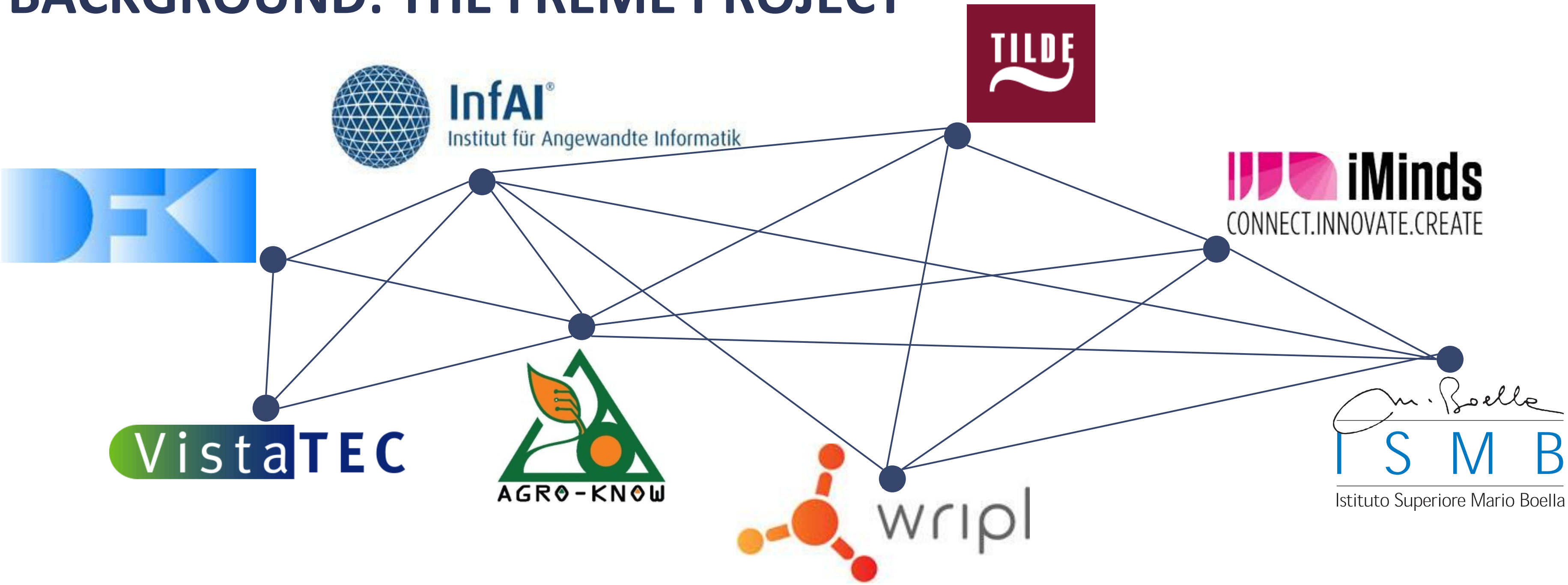
FREME

OPEN FRAMEWORK OF E-SERVICES FOR MULTILINGUAL AND SEMANTIC ENRICHMENT OF DIGITAL CONTENT

www.freme-project.eu

Felix Sasaki, DFKI / W3C Fellow
on behalf of the FREME Consortium

BACKGROUND: THE FREME PROJECT


InfAI
Institut für Angewandte Informatik
TILDE
iMinds
CONNECT.INNOVATE.CREATE
VistaTEC
AGRO-KNOW
wripl
ISMB
Istituto Superiore Mario Boella

- Two year H2020 Innovation action; start February 2020
- Industry partners leading four business cases around digital content and (linked) data
- FREME = A framework for multilingual and semantic enrichment of digital content
- Is there a real need for this? Oh yes! See the following business cases

FREME FROM A TECHNICAL PERSPECTIVE

A framework for multilingual and semantic enrichment of digital content that provides access via a set of APIs and GUIs to six E-services.

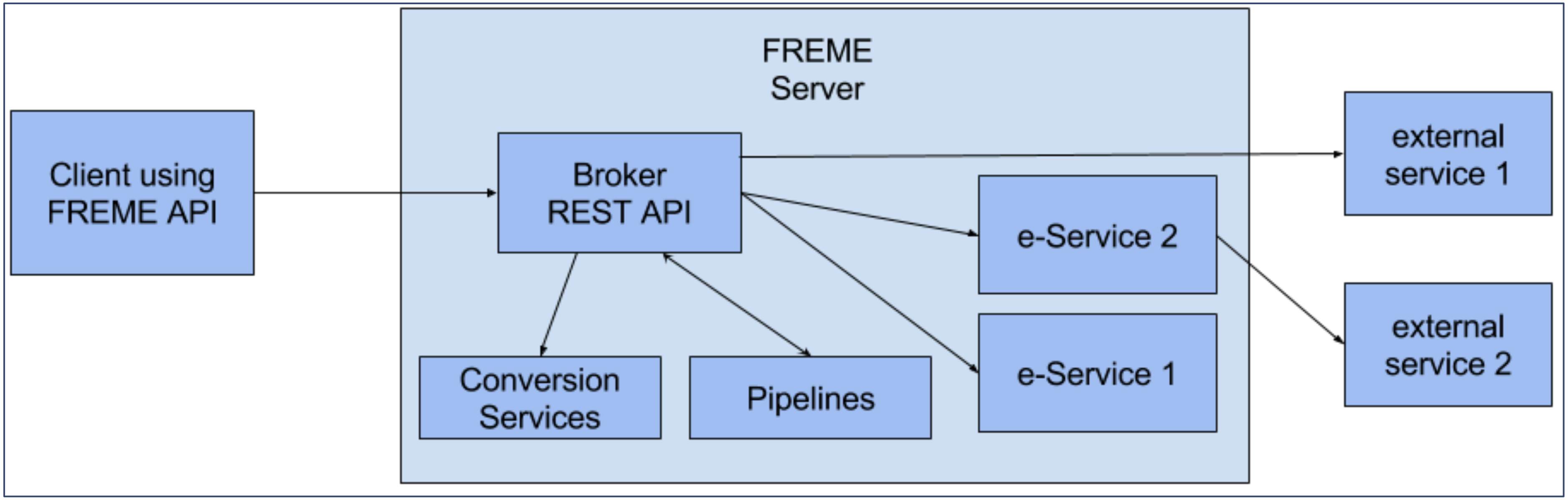
- **e-Entity** for enriching content with information on named entities;
- **e-Link** for enrichment with linked data sources;
- **e-Terminology** for detecting terms and enriching them with term related information;
- **e-Translation** for providing custom machine translation systems;
- **e-Internationalisation** for processing a variety of digital content formats; and
- **e-Publishing** for exporting the outcome of enrichment processes in the ePub format.

FREME FROM A TECHNICAL PERSPECTIVE

How to access FREME – several options:

- A life version including documentation at http://api.freme-project.eu/doc/current/
- A development version at http://api-dev.freme-project.eu/doc/
- FREME packages that you can run on your own! See the documentation for details
- Source code in a GitHub project https://github.com/freme-project/
- The framework is available under Apache 2.0 license to ease commercial use
- Underlying services have various licensing conditions

DESIGN OF THE FRAMEWORK



- Client makes a Web service request.
- The broker evokes the actual e-Service.
- The e-Services are part of the server (e.g. e-Entity), or provided externally (e.g. e-Translation).
- Supportive modules provide conversion of digital content formats or pipelining of services (e.g. e-Terminology followed by e-Translation)

FREME = **a framework, not a platform**: modular approach & ease of extensibility

ALL YOU NEED IS STANDARDS

- HTTP to make web service requests
  - No dependency on a given programming language
- Standards to represent enrichment information
  - See next slide
- Write a wrapper for your existing tools to enable them to produce & consume the enrichment information
  - Enable distributed data and language technology services

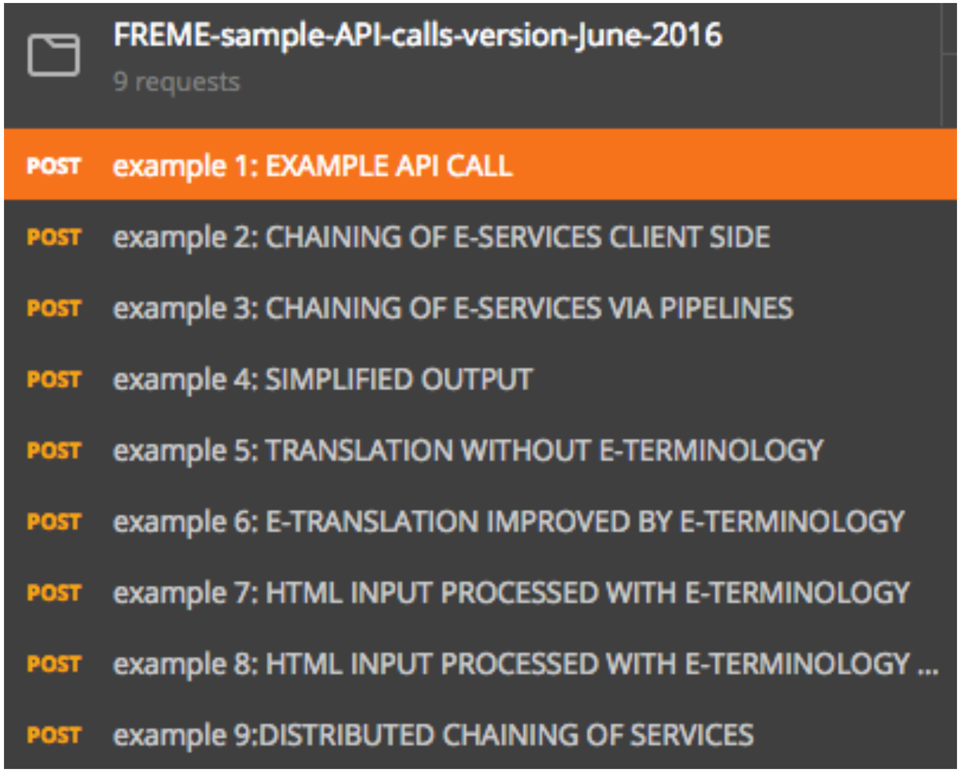
LINGUISTIC LINKED DATA AND OTHER STANDARDS PUT IN ACTION VIA FREME

- **NIF** (Natural Language Processing Interchange Format) for representing digital content and enrichment information in a format agnostic manner, based on the linked data stack;
- **OntoLex lemon** for representing lexical information, to be used e.g. for improving machine translation output;
- **Internationalization Tag Set 2.0** for representing various types of enrichment information in a standardized manner, related e.g. to terminology named entities; and
- The general **linked data technology stack** (RDF, SPARQL etc.)

FREME is built on outcomes of standard driving projects in FP7 in the area of linguistic linked data: LIDER and FALCON

Cf. http://lider-project.eu/ and http://falcon-project.eu/

EXAMPLES



FREME-sample-API-calls-version-June-2016
9 requests

POST example 1: EXAMPLE API CALL
POST example 2: CHAINING OF E-SERVICES CLIENT SIDE
POST example 3: CHAINING OF E-SERVICES VIA PIPELINES
POST example 4: SIMPLIFIED OUTPUT
POST example 5: TRANSLATION WITHOUT E-TERMINOLOGY
POST example 6: E-TRANSLATION IMPROVED BY E-TERMINOLOGY
POST example 7: HTML INPUT PROCESSED WITH E-TERMINOLOGY
POST example 8: HTML INPUT PROCESSED WITH E-TERMINOLOGY ...
POST example 9:DISTRIBUTED CHAINING OF SERVICES

- HTTP requests created with Google Chrome plugin postman
- All postman requests and requests via CURL and jQuery can be downloaded from

https://github.com/freme-project/Documentation

FREME

EXAMPLE API CALL

EXAMPLE API CALL

http://api.freme-project.eu/current/e-entity/freme-ner/documents/?
informat=text&outformat=turtle&language=en&dataset=dbpedia&
input=Welcome to the city of Prague

- The request is made to the API for the e-Entity service, a service that enriches content with named entities.
- The input format of content is plain text; the output format is turtle.
- The content to enrich is “Welcome to the city of Prague”.
- The language or the content is English.
- The dataset used for the enrichment is DBpedia.

See example requests: example 1

EXAMPLE OUTPUT: USING NIF TO STORE CONTENT …

```
(1) <http://freme-project.eu/#char=0,29>
(2)       a nif:String , nif:Context , nif:RFC5147String ;
(3)       nif:beginIndex  "0"^^xsd:int ;
(4)       nif:endIndex    "29"^^xsd:int ;
(5)       nif:isString    "Welcome to the city of Prague"^^xsd:string .
```

1) Identifying the content via a URI
2) Adding certain types from NIF*
3) Identifying the start offset of the content
4) Identifying the end offset of the content
5) Providing the string content itself.

* For More on NIF: see a dedicated tutorial http://de.slideshare.net/m1ci/nif-tutorial

… AND ENRICHMENT INFORMATION

```
(1) <http://freme-project.eu/#char=23,29>  …
(2) nif:anchorOf "Prague"^^xsd:string ;
(3) nif:beginIndex "23"^^xsd:int ;
(4) nif:endIndex "29"^^xsd:int ;
(5) nif:referenceContext  <http://freme-project.eu/#char=0,29> ;
(6) itsrdf:taClassRef <http://dbpedia.org/ontology/City>.
```

1) Identifying the annotation via a URI
2) Providing the string content of the annotation
3) Identifying the start offset of the content
4) Identifying the end offset of the content
5) Relating the content to annotations
6) Enrichment with ITS 2.0 class information (“Prague” = a city)

FREME

CHAINING OF SERVICES, SIMPLIFICATION OF OUTPUT AND FORMAT COVERAGE

CHAINING OF E-SERVICES

Example: combining e-Entity and e-Link

1) e-Entity input: “Welcome to the city of Prague.”
2) Input to e-Link – goal: find geo position from DBpedia data source

Chaining can be done:

1) On the client side
2) By calling the FREME pipelining service

For more infos on pipelines see

http://api.freme-project.eu/doc/current/tutorials/pipeline-entity-link.html

1) The outcome is again NIF, see next slide

See example requests: example 2 and example 3

NIF OUTPUT EXAMPLE (PARTIAL)* FOR CHAINING SERVICES

```
<http://freme-project.eu/#char=0,30>
nif:anchorOf "Welcome to the city of Prague.".
<http://freme-project.eu/#char=23,29>
nif:anchorOf "Prague".
<http://dbpedia.org/resource/Prague>
    <http://www.w3.org/2003/01/geo/wgs84_pos#lat>
        "50.0880428938909”;
    <http://www.w3.org/2003/01/geo/wgs84_pos#long>
        "14.4207572937012" .
```

*The original output has 110 lines … **linked data can be complex!**

SIMPLIFIED OUTPUT HELPS API DEVELOPERS TO CONSUME LINKED DATA

- FREME provides user specified filter mechanism to simply the output
- Supports CSV, XML or JSON
- Example for query output as CSV

uri,ident,class,string,charbegin,charend

"http://freme-project.eu/#char=23,29",http://dbpedia.org/resource/Prague,http://dbpedia.org/ontology/Location,Prague,23,29

For more infos on filtering, see

http://api.freme-project.eu/doc/current/knowledge-base/filtering.html

See example requests: example 4

CHAINING OF E-SERVICES TO IMPROVE RESULTS

- e-Services can be improved by taking up the results of previous services
- Example with simplified output: e-Translation service “understands” the output of e-terminology (providing translation suggestions)

e-Translation only

```
source,target
The EU in brief. The EU is a unique economic and political partnership between 28 European countries that
together cover much of the continent.,In de EU. De EU is een uniek economische en politiek partnerschap
van 28 Europese landen samen op dat deel van het continent.
```

e-Translation followed by e-Terminology

```
source,target
The EU in brief. The EU is a unique economic and political partnership between 28 European countries that
together cover much of the continent.,"De voorschriften in DE EU. De EU is een uniek partnerschap tussen
politiek en economie in de Europese landen, die gezamenlijk 28 verpakking van het continent.
```

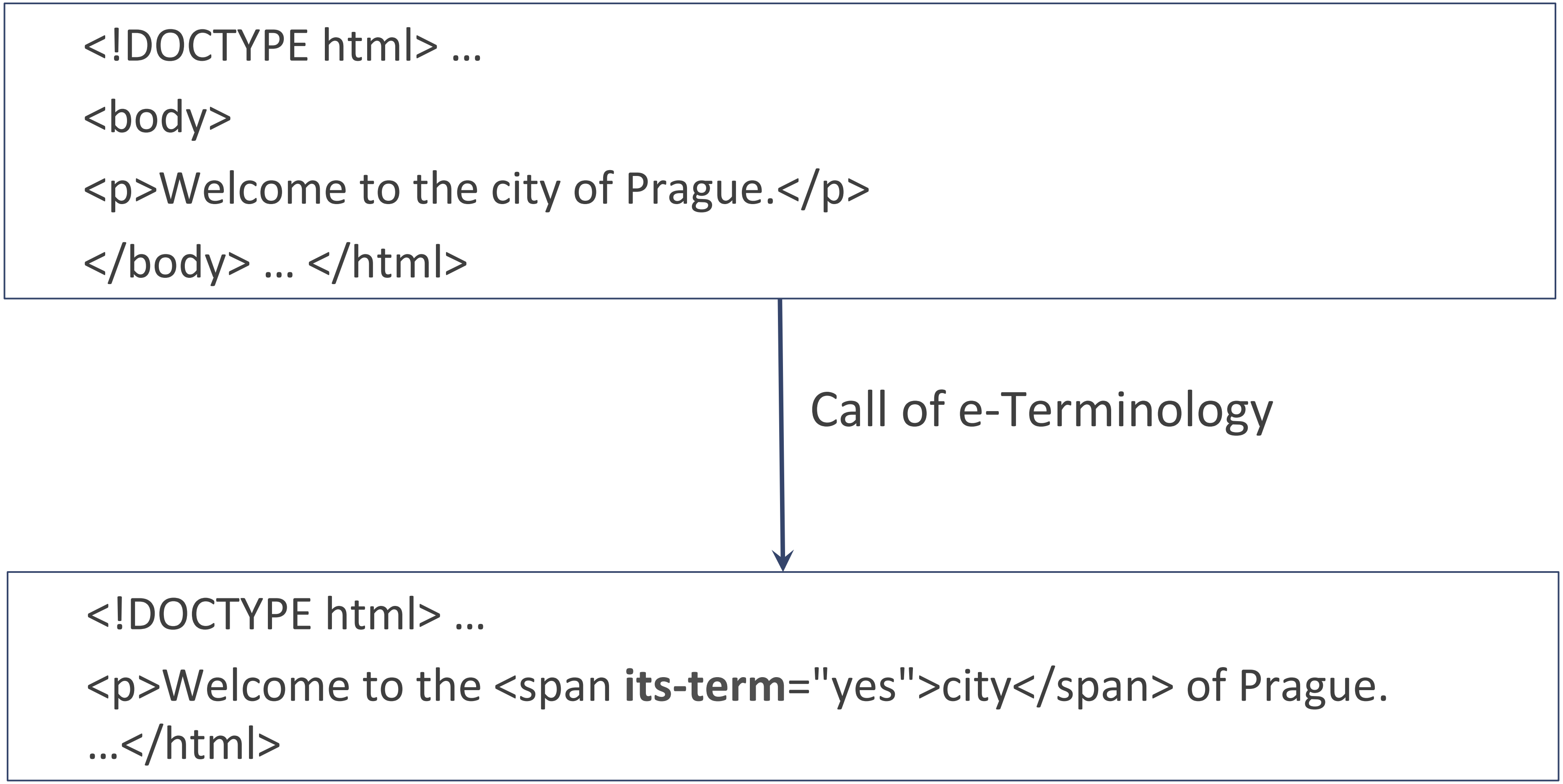
See example requests: example 5 (e-Translation only) and example 6 (chain of e-Translation and e-Terminology)

FORMAT COVERAGE

- Processing of various content formats
  - NIF, RDF, Text, HTML, OpenOffice, XLIFF 1.2, …
- Many formats are processed via e-Internationalization services
- Format specified in API call as input and (partially supported) output
- More information: http://api.freme-project.eu/doc/current/knowledge-base/eInternationalization.html

See example requests: example 7

ROUNDTRIPPING EXAMPLE WITH HTML aND E-TERMINOLOGY



```
<!DOCTYPE html> …
<body>
<p>Welcome to the city of Prague.</p>
</body> … </html>
```

Call of e-Terminology

```
<!DOCTYPE html> …
<p>Welcome to the <span its-term="yes">city</span> of Prague.
…</html>
```

See example requests: example 8

CHAINING OF SERVICES BEYOND FREME - POSSIBLE VIA STANDARDISED ENRICHMENT INFORMATION

- Services provided by the project “Digitale Kuratierungstechnologien” http://digitale-kuratierung.de/

  The services also deploy NIF and ITS 2.0 for storing digital content and enrichment information

The following pipeline will then be possible for you calling the services

1) Call http://digitale-kuratierung.de/ based service to do named entity recognition - output is NIF + ITS 2.0
2) Feed the output to e-Link to gather further information

> Distributed & decentralized language and data technologies, made possible via standardized workflows!

See example requests: example 9

XLIFF and Linked Data in the browser – Why?

- Web developers <> XML technology stack
- Web developers <> Linked data technology stack
- Enrichment of XLIFF content depends on
  - Special purpose tools
  - Programming languages
- Making XLIFF interesting for new clients who use a browser centred ecosystem
  - Without the need to change the XML toolchain!

XML Processing in the browser

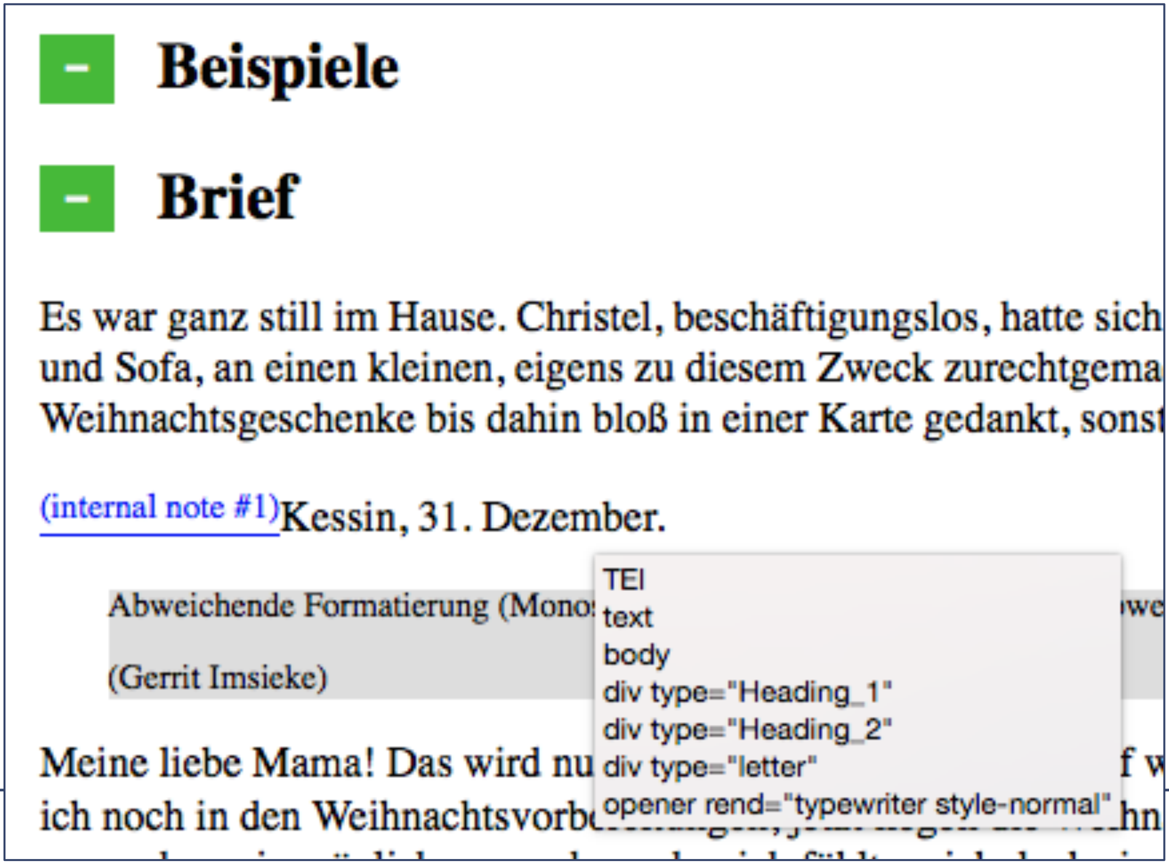
- Native DOMParser allows to process XML content

```
var parser = new DOMParser();
var xmlDoc = '<myDoc>Hello world</myDoc>';
var doc = parser.parseFromString(xmlDoc,'application/xml');
```

- In the past also native XSLT processing (XSLT 1.0) was in browsers
- Today possible with Saxon-CE library

Editing XML in the browser relying on Saxon-CE - WYSIYWG

- Proof-of concept example from Gerrit Imsieke, see http://publishinggeekly.com/wp-content/uploads/2011/06/sxedit/sxedit.html



Beispiele

Brief

Es war ganz still im Hause. Christel, beschäftigungslos, hatte sich
und Sofa, an einen kleinen, eigens zu diesem Zweck zurechtgema
Weihnachtsgeschenke bis dahin bloß in einer Karte gedankt, sonst

(internal note #1)Kessin, 31. Dezember.

Abweichende Formatierung (Mono: we

(Gerrit Imsieke)

Meine liebe Mama! Das wird nu f w
ich noch in den Weihnachtsvorb hn

TEI
text
body
div type="Heading_1"
div type="Heading_2"
div type="letter"
opener rend="typewriter style-normal"

How to process arbitrary XML content in the browser

1) Have XSLT stylesheet that converts XML to HTML
2) Do the processing you want
3) Convert back to XML

- Step 1) and 3) rely on special purpose stylesheets

Saxon-CE Setup

- Add the following link to your HTML document

```
<script type="text/javascript" src="lib/Saxon-CE_1.1/Saxonce/
Saxonce.nocache.js"></script>
```

Saxon-CE Example

```
(1) var proc = Saxon.newXSLT20Processor();
(2) var xslSource = Saxon.requestXML(xslSourcePath);
(3) var xmlSource = Saxon.parseXML(xmlString);
(4) proc.importStylesheet(xslSource);
(5) var fragment = proc.transformToFragment(xmlSource, document);
(6) return Saxon.serializeXML(fragment);
```

Description:

1) Create XSLT20 processor
2) Read XSLT stylesheet
3) Read XML source
4) Important stylesheet to processor
5) Do Transformation
6) Give back XML result fragment

Demo: XLIFF in the Browser

- See http://fsasaki.github.io/stuff/feisgiltt2016/
- Using HTML as pivot format
- Sending HTML to FREME
- Converting the content back to XLIFF

Conclusion: XLIFF and Enrichment workflows work in the browser

- Bridging three technology stacks
  - XML, Linked Data, Browser (Json)
- Encapsulating complexity of XML and Linked Data and Language Technology from Web developers
  - See callFremeNer() method
- “Too much information is no information”
  - Yves Savourel, FEISGILTT 2014, Dublin ☺
- Business value of linked data comes only with adaptability
  - Example: choice of data sets

CONTACTS

E-mail: info@freme-project.eu

CONSORTIUM