UNIVERSITÄT
BIELEFELD

2nd OTTR user forum:

DiProMag & OTTR

Basil Ell, Moritz Blum, Florian Schröder
{bell, mblum}@techfak.uni-bielefeld.de

2021-06-18

DiProMag

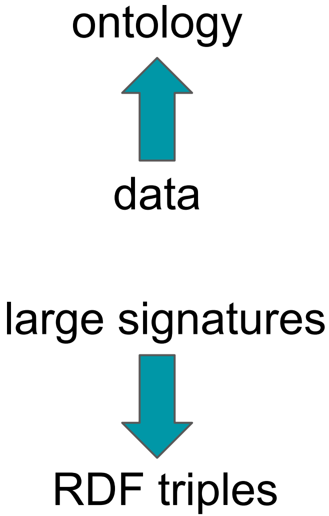
Digitization of a process chain for the production, characterization and prototypical application of magnetocaloric alloys => DiProMag Ontology

- semantic description of experiments
  - general scientific experiments (ambitions, reasons for parameter choices, hypothesis, ...)
  - physical process parameters (objects, elements, temperatures, ...)
- AI systems which use the semantic data

Ontology Development with OTTR

Combination of two strategies:

- Bottom-up: Domain experts design OTTR templates for their use cases in cooperation with ontology experts - data is the starting point.
- Top-down: High-level templates are developed, which can later be specified by low-level templates to arrive at an RDF representation step by step.



ontology

data

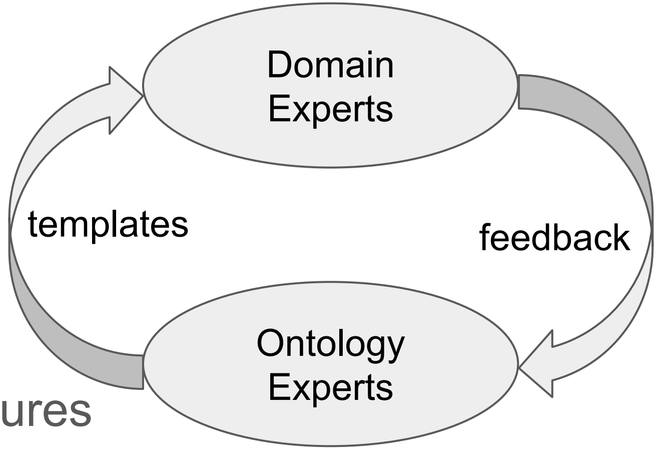
large signatures

RDF triples

Ontology Development with OTTR

Main benefits of / reasons for using OTTR in the communication between ontology and domain experts:

- separation of design and content
- better abstraction
- modular, encapsulated patterns



We started by simply designing OTTR Template signatures with example parameters with the domain experts. In the next step, we make the templates more precise by developing template bodies which use more low level templates.

Ontology Development with OTTR: Target Pattern

```
dipromag:Target[    owl:Class ?target,
                    ottr:IRI ?compound,
                    xsd:float ?density,
                    ottr:IRI ?density_unit,
                    xsd:float ?thickness,
                    ottr:IRI ?thickness_unit     ] ::{

    ax:SubClassOf( ?target, dipromag:target ),
    ottr:Triple( ?target, dipromag:basedOnCompound, ?compound ),
    dipromag:PhysicalProperty( ?target, dipromag:density, ?density, ?density_unit ),
    dipromag:PhysicalProperty( ?target, dipromag:thickness, ?thickness, ?thickness_unit )

}
```

Ontology Development with OTTR: Target Instantiation

dipromag:Target(?target=p:target_4, ?compound=p:Ni_Mn_Sn, ?density="9"^^xsd:float, ?density_unit=p:g_per_qcm, ?thickness="1.7"^^xsd:float, ?thickness_unit=p:mm)

instantiates the template body:

- **ax:SubClassOf**(p:target_4, dipromag:target),
- **ottr:Triple**(p:target_4, dipromag:basedOnCompound, p:Ni_Mn_Sn),
- **dipromag:PhysicalProperty**(
  p:target_4, dipromag:density, "9"^^xsd:float, p:g_per_qcm),
- **dipromag:PhysicalProperty**(
  p:target_4, dipromag:thickness, "1.7"^^xsd:float, p:mm)

Ontology Development with OTTR: Target Triples

```
p:target_4 rdf:subClassOf dipromag:target .

p:target_4 dipromag:basedOnCompound p:Ni_Mn_Sn .

p:target_4 dipromag:hasPhysicalProperty _:b1 .

_:b1 dipromag:hasPropertyType dipromag:density .

_:b1 dipromag:hasPropertyValue "9"^^xsd:float .

_:b1 dipromag:hasPropertyUnit  p:g_per_qcm .

p:target_4 dipromag:hasPhysicalProperty :_b2 .

_:b2 dipromag:hasPropertyType dipromag:thickness .

_:b2 dipromag:hasPropertyValue "1.7"^^xsd:float .

_:b2 dipromag:hasPropertyUnit p:mm .
```

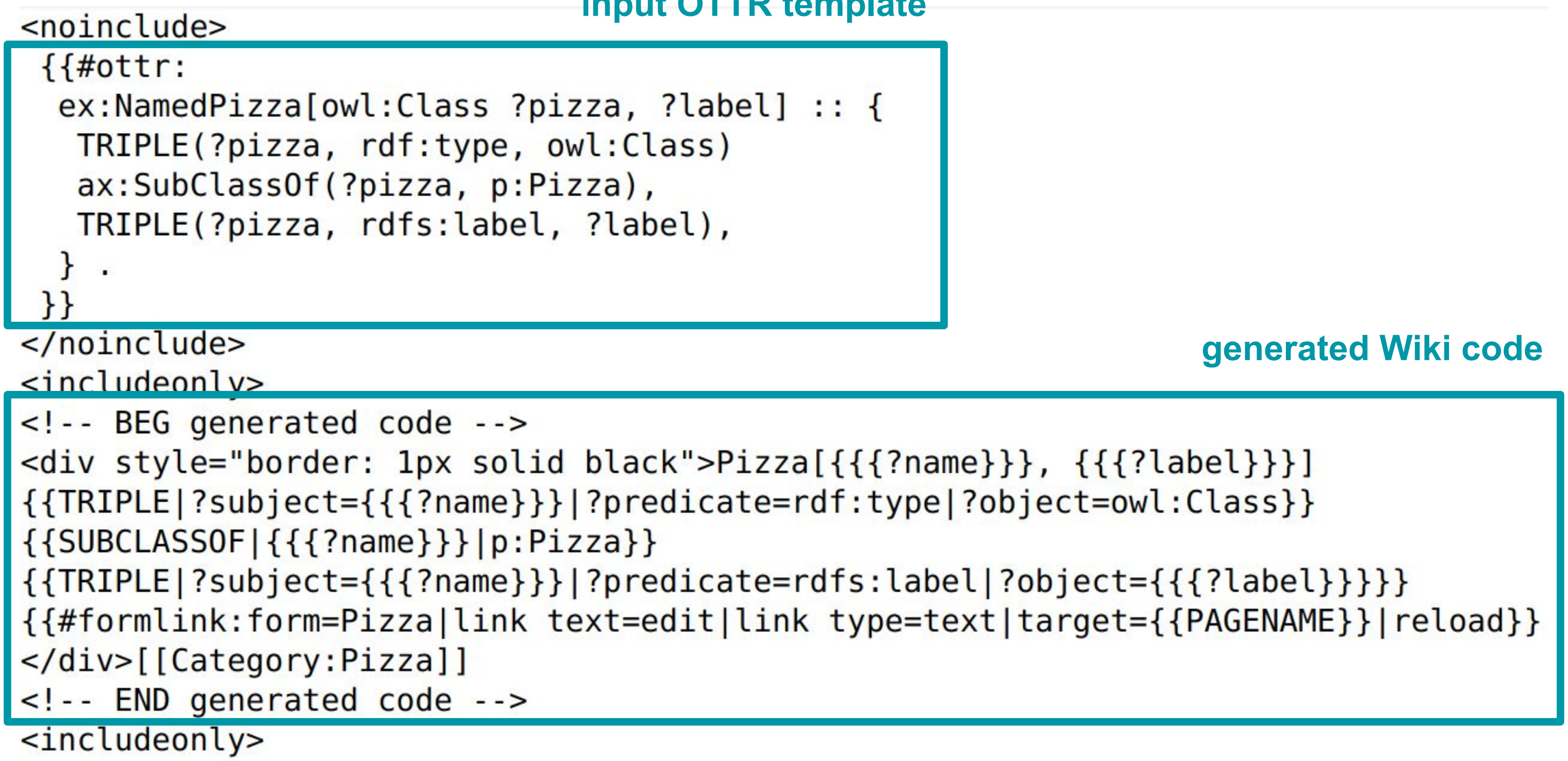
Upcoming: OTTR SMW Extension

The goal is to integrate OTTR into Semantic Media Wiki.

- function that compiles stOTTR syntax into wiki code, s.t. OTTR can be used in and interpreted by the SMW
- each OTTR template becomes a Wiki template
- relational data is stored directly in the Wiki, merges with the normal user content, and is accessible to the users
- simplified user interaction through forms
- in addition to supporting nested template definitions, we plan to integrate:
  - typed parameters
  - optional arguments
  - non blank parameters
  - default values
  - list expanders
  - blank nodes

Pizza Example: Template Creation

Page: Template Pizza

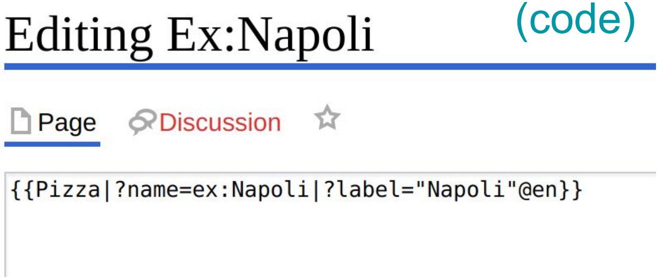


input OTTR template

generated Wiki code

```
<noinclude>
 {{#ottr:
  ex:NamedPizza[owl:Class ?pizza, ?label] :: {
   TRIPLE(?pizza, rdf:type, owl:Class)
   ax:SubClassOf(?pizza, p:Pizza),
   TRIPLE(?pizza, rdfs:label, ?label),
  } .
 }}
</noinclude>
<includeonly>
<!-- BEG generated code -->
<div style="border: 1px solid black">Pizza[{{{?name}}}, {{{?label}}}]
{{TRIPLE|?subject={{{?name}}}|?predicate=rdf:type|?object=owl:Class}}
{{SUBCLASSOF|{{{?name}}}|p:Pizza}}
{{TRIPLE|?subject={{{?name}}}|?predicate=rdfs:label|?object={{{?label}}}}}
{{#formlink:form=Pizza|link text=edit|link type=text|target={{PAGENAME}}|reload}}
</div>[[Category:Pizza]]
<!-- END generated code -->
<includeonly>
```

Pizza Example: Template Instantiation

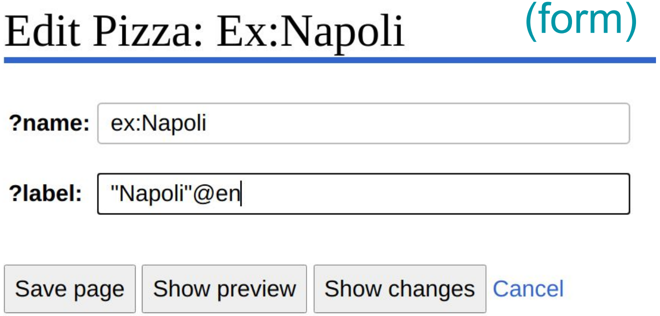


(code)

Editing Ex:Napoli

```
{{Pizza|?name=ex:Napoli|?label="Napoli"@en}}
```

OR

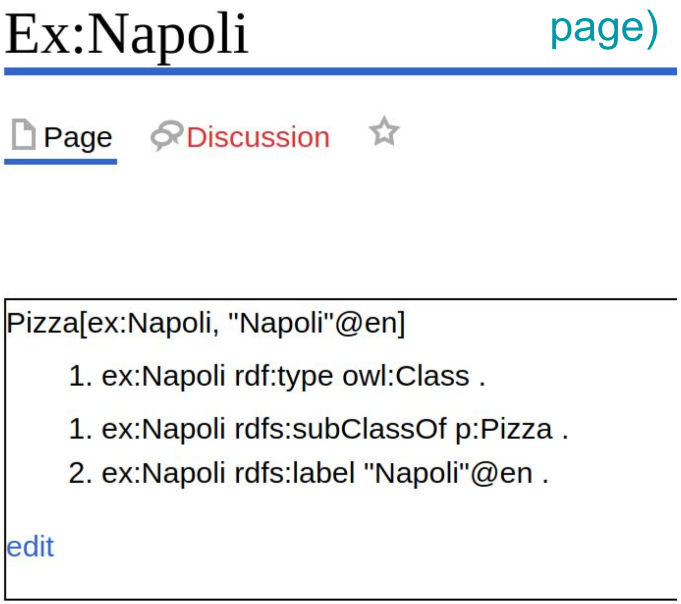


(form)

Edit Pizza: Ex:Napoli

?name: ex:Napoli

?label: "Napoli"@en



(generated wiki page)

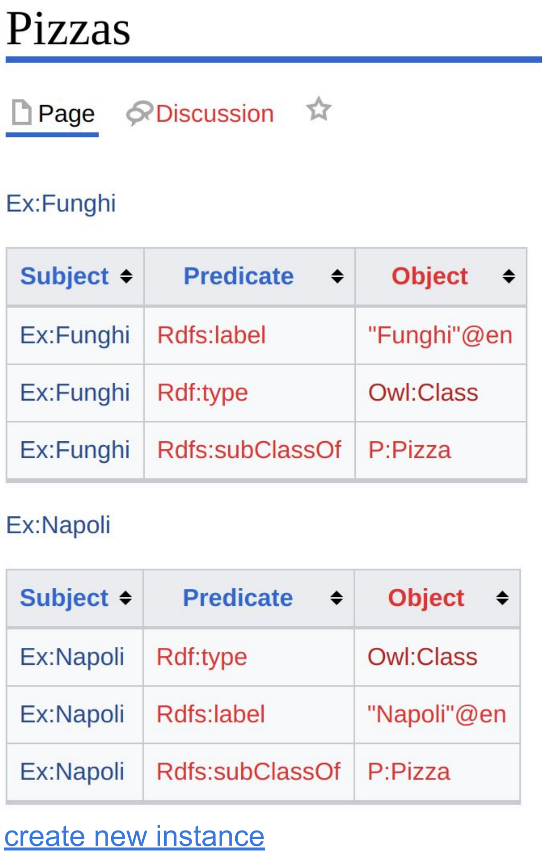
Ex:Napoli

Pizza[ex:Napoli, "Napoli"@en]

1. ex:Napoli rdf:type owl:Class .

1. ex:Napoli rdfs:subClassOf p:Pizza .
2. ex:Napoli rdfs:label "Napoli"@en .

Pizza Example: Data Overview



Ex:Funghi

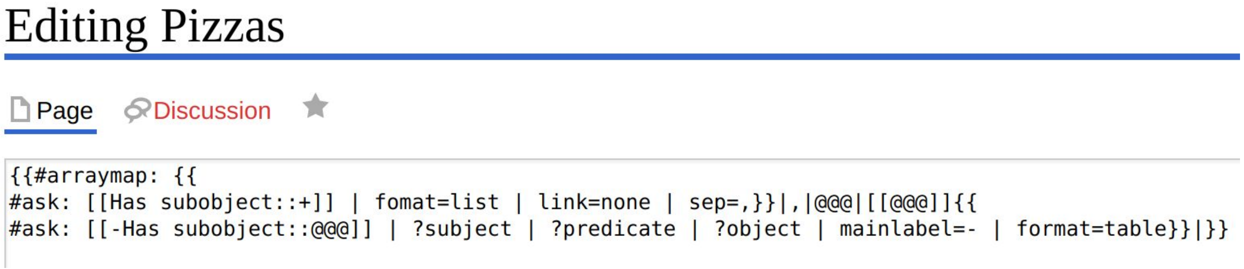
Subject	Predicate	Object
Ex:Funghi	Rdfs:label	"Funghi"@en
Ex:Funghi	Rdf:type	Owl:Class
Ex:Funghi	Rdfs:subClassOf	P:Pizza

Ex:Napoli

Subject	Predicate	Object
Ex:Napoli	Rdf:type	Owl:Class
Ex:Napoli	Rdfs:label	"Napoli"@en
Ex:Napoli	Rdfs:subClassOf	P:Pizza

create new instance

SMW directly allows queries over the data, e.g. list all created RDF triples for each OTTR instance.



```
{{#arraymap: {{
#ask: [[Has subobject::+]] | fomat=list | link=none | sep=,}}|,|@@@|[[@@@]]{{
#ask: [[-Has subobject::@@@]] | ?subject | ?predicate | ?object | mainlabel=- | format=table}}|}}
```

Thanks for your attention!

Question Time