Federated Semantics through Attaching Foreign Vocabulary

Attaching Foreign Semantics

To open up the closed world of a vocabulary,
foreign vocabulary can be added in:

- The main vocabulary
- The instance
- The media type of the instance
- The self-description of the Thing
- A **semantic style**

Foreign Semantics in a Vocabulary

Can be easily added in the "T-Box":

— An IPSO temperature **is a** QUDT Celsius Temperature

— An IPSO temperature **is a** UCUM Cel

This requires a place in the vocabulary where that T-Box can be added.
➔ Vocabularies need to be able to refer to other vocabularies.
➔ Can only be done at vocabulary design time

Foreign Semantics in an Instance

Add semantics in the structure of the Instance:

— E.g., HTML and Microformats

This requires a place in the instance where extensible information can be added.
➔ Instance developer needs to be aware about all applicable foreign semantics
➔ Instance may get large if the developer embraces many of these (optimizations?)

Foreign Semantics in the Media Type

Pretty much equivalent to adding foreign semantics to the vocabulary of that media type

Foreign Semantics in the Self-Description

Add semantics to self-description ("type information")

— E.g., in WoT TD

Requires type information in self-description to be able to reach into resource structure as well as the internal structure of resources

Only works for semantic information that changes at the time constants of changes to the self-description ("metadata")

Foreign Semantics in a Semantic Style

Attach information via **selectors** into the instance

— Similar to adding style semantics to HTML via CSS

Requires **selector language** to generically (or specifically!) identify internal structure of the resource

(This could also be used to add foreign semantics to a self-description or even a media type)

Conclusion 2017-12-18

We probably need all of these.

Work on:

- Seamlessly moving between on structure **built out of** resources and structure **within** resources
- Attaching semantic information to structured data; **selector** languages for the IoT

Somewhat related example provided by Klaus Hartke:

```
#using <http://example.org/wishi#>
#using ipso = <http://example.org/ipso/vocabulary/>

ipso:Object3303 </temp1> {
    is-a <http://iot.linkeddata.es/def/wot#Thing>
    is-a <http://iotschema.org/TemperatureCapability>

    ipso:Item5700 </temp1/value> {
        is-a <http://iot.linkeddata.es/def/wot#Property>
        is-a <http://iotschema.org/TemperatureProperty>

        media-type "application/senml+json" {
            field "$[*].v" {  // JSONPath
                is-a <http://iotschema.org/TemperatureData>
                type "number"
                minimum -50
                maximum 100
                unit <http://qudt.org/vocab/unit/DEG_C>
                unit <http://unitsofmeasure.org/ucum.html#Cel>
            }
        }
    }
}
```

Approaches

Transformation language (DSSSL/XSLT)

Augmentation language (CSS)

Rules

selector ➔ effect

```
.warning {color: red}
```

Selector

Selects zero or more structural elements into a "node set"

- HTML Elements in CSS
- XML Elements, Attributes or other parts of the ESIS in XSLT and XPath

Effect is then applied to each selection

Effect

Add attribute to each selection

— kept in a separate namespace by CSS (properties)

Generate additional structure (even in augmentation)

— CSS: `::before`, `::after`…

Can involve computation (e.g., counters)

Value spaces?

Selection components

Navigational

— E.g., XSLT/XPath axes (ancestor ancestor-or-self attribute child descendant descendant-or-self following following-sibling namespace parent preceding preceding-sibling self @ // .. .)

Type-based, Attribute-based

— XML GI, XML classes, ids, other attributes

Computational, inference-based

Computational, inference-based

- `:odd`, `:even`, … ➔ structural computation
- can the effect of other rules be used?
- Full "T-boxes" as in OWL