Value Sets via Ontology Views

Landon T Detwiler1, James F Brinkley1,2,3, MD, PhD

Departments of Biological Structure1, Computer Science and Engineering2, Medical Education and Biomedical Informatics3,
University of Washington, Seattle, WA

Introduction:
We present methods for defining value sets from ontologies using reusable template queries and parameterized URLs.

Motivation:
One of the challenges of the Human Studies Database (HSDB) project [1] is to provide methods for describing sets of allowed values (value sets) permissible in data entry form fields (i.e. case report forms or CRFs). Pre-defined enumerations of permissible values could be encoded directly into data entry applications. But such an approach lacks a reusable description of how and from where its values were derived. As one method for providing such a description, we implemented an ontology query-based solution in our Query Manager (QM) application [2].

Example:
The QM supports multiple query languages and data models. The Query Execution Service (QES) extension provides a REST interface which allows URLs to refer to [the results of] query evaluations. We present an example which demonstrates the utility of combining the QM and QES components for value set definition. For our example use-case, we will extract a “Telecommunication scheme” (TS) value set, as an XML schema definition (XSD), from the Ontology of Clinical Research (OCRe).

The source:

![XML schema definition](image)

Instances of the class “Telecommunication scheme”, from OCRe, viewed in the Protégé ontology editor.

Basic SPARQL query:

```sparql
CONSTRUCT { 
  ?valset:TelecomScheme ?valset:hasMember _<00. 
  _<00 rdf:type ?valset:Member. 
  _<00 ?valset:URI _<01. 
  _<00 ?valset:label ?valset:label } 
FROM http://purl.org/sig/OCRe>
WHERE 
  (?root rdfs:label "Telecommunication scheme". 
  ?valset rdfs:label ?label.) 
```

Reusable template SPARQL query (qid=135):

```sparql
CONSTRUCT { 
  <$0> ?valset:hasMember _<00. 
  _<00 rdf:type ?valset:Member. 
  _<00 ?valset:URI _<01. 
  _<00 ?valset:label ?valset:label } 
FROM <$0>
WHERE 
  (?root rdfs:label <$2>. 
  ?valset:label ?valset:label. ) 
```

Parameterized URL (qid=136):

```
http://purl.org/sig/TemplateQuery?qid=135&args= 
valset:TelecomScheme, <http://purl.org/sig/OCRe>, 
"Telecommunication scheme"
```

The above was saved in the QM as a “URL query”.

```
<#0> = valset:TelecomScheme 
<#1> = http://purl.org/sig/OCRe> 
<#2> = "Telecommunication scheme"
```

Results (partial) of executing the above URL:

```
```

Why template queries?

The “reusable template query” (e.g. qid=135) extracts value sets from an ontology where members are represented as class instances. This is a common representational pattern for value sets [3]. We’d like to reuse our query to extract other value sets similarly represented. Template queries support query reuse by providing replaceable placeholders in key locations (i.e. CRFs). Placeholder substitutions are passed to the template query via parameterized URLs. The example query has template placeholders for the URI of the generated value set, the URL of the source ontology, and the (human-readable) name of the value set. With the substitutions shown in our “parameterized URL” (qid=136) our example query extracts, as an RDF value set, the names of all instances of TS in OCRe.

Format conversion template query (qid=132):

```
let $members = doc:<#0> print/Description 
where 
<#0> simpleType name="<#1>" OC:QFEntityURI="<#0>" 
<#0> restriction base="xsd:string" 
for $member in $members return 
cast/enumeration 
value="$member/valset:label/text()" 
OC:QFEntityURI="{"$member/valset:URI@rdf:resource"}" 
)`
```

Conclusion:

Ontologies provide a rich source for deriving allowed values for data entry forms, such as CRFs. Common approaches for representing value set membership include the class instance pattern illustrated here. Template queries and parameterized URLs allow us to encode the pattern in a reusable way. Similar value sets may be generated by changing parameters in the URL. Additionally, queries may be chained to create a kind of query processing pipeline.

Acknowledgements:

Supported by NIH HL087706, RR026040, RR25014

References: