Member Nodes - Story #8833

Problems utilizing pyshacl within SlenderNodes

2019-08-05 19:37 - John Evans

Status:	New	**Start date:**	2019-08-05
Priority:	Normal	**Due date:**	
Assignee:	John Evans	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
Story Points:			

Description

Opening this ticket to document issues encountered when trying to utilize pyshacl within SlenderNodes.

History

#1 - 2019-08-05 20:36 - John Evans

We don't seem to be able to validate all datatypes, although I'm not sure how big of an issue this really is. For example, suppose we are looking at the "dateModified" keyword.

Suppose the following SHACL graph:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix schema: <http://schema.org/> .
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

schema:DatasetShape
    a sh:NodeShape ;
    sh:targetClass schema:Dataset ;
    sh:property [
        sh:path schema:encoding ;
        sh:node schema:MediaObjectShape ;
        sh:minCount 1 ;
        sh:message "JSON-LD is missing a top-level encoding keyword." ;
    ] .

schema:MediaObjectShape
    sh:property [
        sh:path schema:dateModified ;
        sh:datatype xsd:date      ;
        sh:message "JSON-LD dateModified does not validate" ;
    ] .
```

Suppose the following JSON-LD:

```
        {
            "@context": { "@vocab": "http://schema.org/" },
            "@type": "Dataset",
            "encoding": {
                "dateModified": "2004-02-02"
            }

        }
```

That should be a valid date, right?   The SHACL playground again reports an error with the encoding keyword without revealing the deeper error. Using the debug option on the command line shows the following:

```
$ pyshacl -s tmp/shacl2.ttl -i rdfs -f human -df json-ld tmp/jsonld2.json -d
Constraint Violation in DatatypeConstraintComponent (http://www.w3.org/ns/shacl#DatatypeConstraintComponent):
    Severity: sh:Violation
    Source Shape: [ sh:datatype xsd:date ; sh:message Literal("JSON-LD dateModified does not validate") ; sh:p
ath schema:dateModified ]
    Focus Node: [ ]
    Value Node: Literal("2004-02-02")
    Result Path: schema:dateModified
    Message: JSON-LD dateModified does not validate

Constraint Violation in NodeConstraintComponent (http://www.w3.org/ns/shacl#NodeConstraintComponent):
    Severity: sh:Violation
    Source Shape: [ sh:message Literal("JSON-LD is missing a top-level encoding keyword.") ; sh:minCount Liter
al("1", datatype=xsd:integer) ; sh:node schema:MediaObjectShape ; sh:path schema:encoding ]
    Focus Node: [ ]
    Value Node: [ ]
    Result Path: schema:encoding
    Message: JSON-LD is missing a top-level encoding keyword.
```

I honestly do not understand the issue here. An alternative approach might be to use regular expression validation, but that doesn't seem like a great approach.

#2 - 2019-08-09 17:05 - John Evans

The point of failure here seems to be in the pyshacl.constraints.core.value_constraints module, DatatypeConstraintsComponent class, evaluate method.

The "datatype" and "lang" attributes for the value node, an rdflib.term.Literal('2004-02-02'), are both None, and the "datatype rule" being invoked is for "rdflib.term.Literal('2004-02-02')", but it is checked against "rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#string')", so no check is actually invoked, and the default result of False is returned.

#3 - 2019-08-09 18:20 - John Evans

We can attempt to get around this by invoking a sh:pattern (i.e. regular expression) check instead. The shacl graph here is as follows:

```
@prefix ex: <http://www.w3.org/example#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix schema: <http://schema.org/> .
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix d1: <http://ns.dataone.org/schema/2019/08/SO/Dataset#> .

d1:DatasetShape
    # Specifies required properties of a Dataset instance which
    # is used to describe a dataset and how it can be accessed.
    a sh:NodeShape ;
    sh:targetClass schema:Dataset ;

    sh:property [
        # A Dataset must have an encoding property of type MediaObject
        # The encoding property provides the location and format of
        # an alternate encoding of metadata describing the dataset.

        sh:path schema:encoding ;
        sh:class schema:MediaObject ;
        sh:minCount 1 ;
        sh:message "A MediaObject is required in order to identify the location of the XML encoding of the met
adata."@en ;
        sh:severity sh:Violation ;

        sh:property [
            sh:path schema:dateModified ;
            sh:minCount 1 ;
            sh:message "A dateModified property indicating when the encoding was last updated is recommended."
@en ;
        ] ;
        sh:property [
            sh:path schema:dateModified ;
        sh:pattern "^-?[0-9]{4}-(0[1-9]|1[1-2])-([0-2][0-9]|3[0-1])($|T([0-1][0-9]|2[0-3]):[0-5][0-9]:[0-5][0-
9]([.][0-9]+)?(Z$|([+-](((0[0-9]|1[0-3]):[0-5][0-9])$|14:00$))|$))" ;
            sh:message "A dateModified property must be xsd:date."@en ;
        ] ;
    ] .
```

This works against the previous data graph and holds up in unit tests. However, it will fail under one circumstance.

If the "@context" is set to be inline, i.e. "http://schema.org", then behavior is different than if the "@context" is expanded, i.e. {'@vocab': 'http://schema.org/'}. In the former, there is a network fetch of the context from schema.org. This causes the dateModified node datatype to be

```
rdflib.term.Literal('2019-08-08T00:00:01', datatype=rdflib.term.URIRef('http://schema.org/Date'))
```

It seems that rdflib doesn't return the string value correctly here, resulting in 'None' instead of the expected '2019-08-08T00:00:01'. The 'None' string then fails the regex check, of course.

The network fetch does not occur in the latter case, the datatype ends up being

```
rdflib.term.Literal('2019-08-08')
```

and rdflib does extract the intended string properly to check against the regex, which passes as expected.

So this is a bit problematic. It seems that any of the "date" keys from the schema.org context document have @type 'Date' or 'DateTime', so these keys will all have the same problem.

One way around this issue, at least for the scanner, would be to examine the '@context' before taking any further action, and if it's inline, swap it out with an expanded context to prevent the network fetch, this would prevent the date literals from augmenting themselves with the the schema.org type.