

Data Validation Workshop

Rail Data Forum 2025

Veronika Heimsbakk & Jose Emilio Labra Gayo

World assumptions

Open world assumption (OWA)

- › Admits incomplete knowledge.
- › Ontologies with Web Ontology Language (OWL).

The assumption that the truth value of a statement may be true irrespective of whether or not it is known to be true.



Example

Statement: In a **hole in the ground** there lived a **hobbit**.

Question: Do **Gandalf** live in a **hole in the ground**?

OWA: Unknown

Closed world assumption (CWA)

- › Shape constraints with Shape Constraint Language (SHACL).

Any statement that is true is known to be true. What is not currently known to be true is false.



Example

Statement: In a **hole in the ground** there lived a **hobbit**.

Question: Do **Gandalf** live in a **hole in the ground**?

CWA: No

Common

- › RDF & URIs
- › Rely on RDF Schema (RDFS)

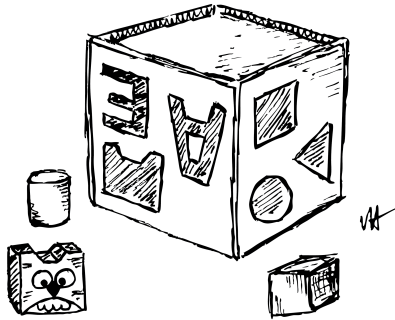
Difference



Inference

Validation

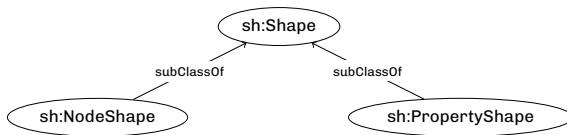
Shapes Constraint Language



SHACL Shape

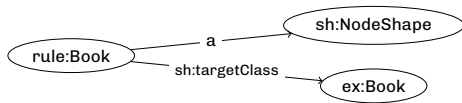
A collection of constraints for given RDF resource.

- › Shapes about focus nodes (**sh:NodeShape**).
- › Shapes about values of a property or path for the focus node (**sh:PropertyShape**).



sh:NodeShape

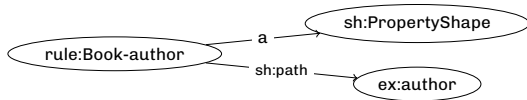
A *node shape* is a shape that describes focus nodes (subject)!



```
rule:Book
  a sh:NodeShape ;
  sh:targetClass ex:Book .
```

sh:PropertyShape

A *property shape* is a shape that describes properties and their values (and that is the subject of a triple that has *sh:path* as its predicate).



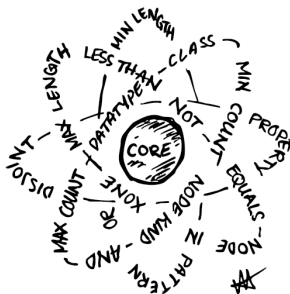
```
rule:Book-author
  a sh:PropertyShape ;
  sh:path ex:author .
```

Books with authors

```
rule:Book
  a sh:NodeShape ;
  sh:targetClass ex:Book ;
  sh:property rule:Book-author .

rule:Book-author
  a sh:PropertyShape ;
  sh:path ex:author .
```

SHACL Core Constraint Components



SHACL Core Constraint Components

Value type

sh:class	Each value node is an instance of a given type.
sh:datatype	Datatype of each value node.
sh:nodeKind	Node kind (IRI, blank node etc.) of each value node.

```
rule:Book
  a sh:NodeShape ;
  sh:targetClass ex:Book ;
  sh:property [
    sh:path ex:author ;
    sh:class foaf:Person ;
  ] ;
  sh:property [
    sh:path ex:published ;
    sh:datatype xsd:date ;
  ] .
```

SHACL Core Constraint Components — example

Example of conforming data

```
ex:TheHobbit a ex:Book ;  
  ex:author ex:JRR Tolkien ;  
  ex:published "1937-09-21"^^xsd:date .  
  
ex:JRR Tolkien a foaf:Person .
```

Example of non-conforming data

```
ex:TheHobbit a ex:Book ;  
  ex:author ex:JRR Tolkien ;  
  ex:published "1937"^^xsd:gYear .  
  
ex:JRR Tolkien a ex:Person .
```

SHACL Core Constraint Components

Cardinality

sh:minCount	Minimum cardinality as xsd:integer.
sh:maxCount	Maximum cardinality as xsd:integer.

Value range

sh:minExclusive	$x < \$value$
sh:minInclusive	$x \leq \$value$
sh:maxExclusive	$x > \$value$
sh:maxInclusive	$x \geq \$value$

```
rule:Book
  a sh:NodeShape ;
  sh:targetClass ex:Book ;
  sh:property [
    sh:path ex:mainTitle ;
    sh:datatype xsd:string ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
  ] ;
```

```
sh:property [
  sh:path ex:pages ;
  sh:datatype xsd:integer ;
  sh:minInclusive 10 ;
  sh:maxExclusive 10000 ;
] .
```


SHACL Core Constraint Components — example

Example of conforming data

```
ex:TheHobbit a ex:Book ;  
  ex:mainTitle "The Hobbit, or There and Back Again" ;  
  ex:pages "310"^^xsd:integer .
```

Example of non-conforming data

```
ex:TheHobbit a ex:Book ;  
  ex:mainTitle "The Hobbit, or There and Back Again" ;  
  ex:mainTitle "Hobbiten, eller ditut og attende" ;  
  ex:pages "5"^^xsd:integer .
```

SHACL Core Constraint Components

String-based

sh:minLength	Minimum length as xsd:integer.
sh:maxLength	Maximum length as xsd:integer.
sh:pattern	Regular expression.
sh:languageIn	A list of languages.
sh:uniqueLang	One unique tag per language.

```
rule:Book
  a sh:NodeShape ;
  sh:targetClass ex:Book ;
  sh:property [
    sh:path ex:mainTitle ;
    sh:datatype rdf:langString ;
    sh:minLength 2 ;
    sh:maxLength 100 ;
    sh:uniqueLang true ;
  ] ;
  sh:property [
    sh:path ex:isbn ;
    sh:pattern "^(?=(?:\D*\d){10}(?:\D*\d){3})?$(\d-)+$" ;
  ] .
```

Example of conforming data

```
ex:TheHobbit a ex:Book ;  
  ex:mainTitle "Hobbiten, eller ditut og attende"@nn ;  
  ex:mainTitle "Hobbiten, eller Fram og tilbake igjen"@nb ;  
  ex:mainTitle "The Hobbit, or There and Back Again"@en ;  
  ex:isbn "978-0-261-10221-7" .
```

Example of non-conforming data

```
ex:TheHobbit a ex:Book ;  
  ex:mainTitle "The Hobbit"@en ;  
  ex:mainTitle "The Hobbit, or There and Back Again"@en ;  
  ex:isbn "123" .
```

SHACL Core Constraint Components

Property pair	Compare two IRIs where,
sh:equals	$x \equiv y$
sh:disjoint	$x \cap y = \emptyset$
sh:lessThan	$x < y$
sh:lessThanOrEquals	$x \leq y$

```
rule:Person
  a sh:NodeShape ;
  sh:targetClass ex:Person ;
  sh:property [
    sh:path ex:birthDate ;
    sh:lessThanOrEquals
      ex:deathDate ;
  ] .
```

SHACL Core Constraint Components — example

Example of conforming data

```
ex:JRR Tolkien a ex:Person ;  
  ex:birthDate "1892-01-03"^^xsd:date ;  
  ex:deathDate "1973-09-02"^^xsd:date .
```

Example of non-conforming data

```
ex:JRR Tolkien a ex:Person ;  
  ex:birthDate "1992-01-03"^^xsd:date ;  
  ex:deathDate "1973-09-02"^^xsd:date .
```

SHACL Core Constraint Components

Logical	List of value nodes that,
sh:not	Cannot conform to given shape.
sh:and	Conforms to all provided shapes.
sh:or	Conforms to at least one of the provided shapes.
sh:xone	Conforms to exactly one of the provided shapes.

```
rule:Person
  a sh:NodeShape ;
  sh:targetClass ex:Person ;
  sh:xone (
    [
      sh:path ex:birthDate ;
      sh:datatype xsd:date ;
      sh:minCount 1 ;
    ]
    [
      sh:path ex:birthYear ;
      sh:datatype xsd:gYear ;
      sh:minCount 1 ;
    ]
  ) .
```

SHACL Core Constraint Components — example

Example of conforming data

```
ex:JRR Tolkien a ex:Person ;  
  ex:birthDate "1982-01-03"^^xsd:date .
```

Example of non-conforming data

```
ex:JRR Tolkien a ex:Person .
```

SHACL Core Constraint Components

Shape-based

sh:node

Each value node,

Conforms to the given node shape.

sh:property

Has a given property shape.

Other

sh:closed

Boolean signalling a complete shape.

sh:ignoredProperties

List of properties to ignore.

sh:hasValue

At least one value node is equal to the given term.

sh:in

Value node is member of given list.

```
rule:TheHobbit
  a sh:NodeShape ;
  sh:targetNode ex:TheHobbit ;
  sh:closed true ;
  sh:ignoredProperties (rdf:type) ;
  sh:property [
    sh:path ex:author ;
    sh:hasValue ex:JRRRTolkien ;
  ] .
```


SHACL Core Constraint Components — example

Example of conforming data

```
ex:TheHobbit a ex:Book ;  
  ex:author ex:JRRRTolkien .
```

Example of non-conforming data

```
ex:TheHobbit a ex:Book ;  
  ex:genre ex:Fantasy ;  
  ex:author ex:JRRRTolkien .
```

Oooops! Targets!

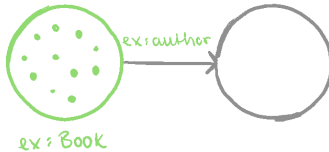
TARGET CLASS



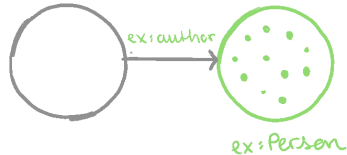
TARGET NODE



TARGET SUBJECTS OF



TARGET OBJECTS OF



Quirks!

```
rule:Book
  a sh:NodeShape, owl:Class ;
  sh:property [
    sh:path ex:author ;
    sh:or (
      [ sh:class foaf:Person ; ]
      [ sh:datatype xsd:string ; ]
    )
  ] .
```

Other nice to know about SHACL

For property shapes:

`sh:name` A label. Can have multiple values, one per language tag.

`sh:description` Description or comment. Can have multiple values, one per language tag.

`sh:order` Relative order as a decimal number.

`sh:group` Have range to an URI instance of `sh:PropertyGroup`.

`sh:defaultValue` Contains no fixed semantics. Should align with `sh:datatype` or `sh:class` for given shape.

SHACL Core Constraints not enough for you?



SHACL-SPARQL

sh:message Annotation used for validation result.
sh:prefixes IRI to prefix mapping.
sh:select String containing the SPARQL query.

```
ex:Book
  a sh:NodeShape ;
  sh:message "The mainTitle must be in Nynorsk."@en ;
  sh:sparql [
    a sh:SPARQLConstraint ;
    sh:prefixes ex: ;
    sh:select """
      SELECT $this ?mainTitle WHERE {
        $this ex:mainTitle ?mainTitle .
        FILTER(LANG(?mainTitle) = "nn")
      }
      """ ;
  ] ;
.
```

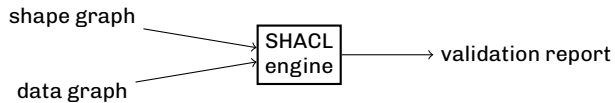
SHACL-SPARQL

```
ex:
  a owl:Ontology ;
  owl:imports sh: ;
  sh:declare [
    sh:prefix "ex" ;
    sh:namespace "http://data.veronahe.no/collection/"^^xsd:anyURI ;
  ] ;
  sh:declare [
    sh:prefix "rule" ;
    sh:namespace "http://rules.veronahe.no/collection/"^^xsd:anyURI ;
  ] .
```

SHACL-SPARQL

- › Queries cannot contain a `MINUS` clause.
- › Queries cannot be a federated query (`SERVICE`).
- › Queries cannot contain a `VALUES` clause.
- › Queries cannot use the syntax form `AS ?x` for any prebound variable.

SHACL engine



Validation report

Each instance of *sh:ValidationReport* has exactly one value of *sh:conforms*.

sh:conforms is true iff the validation did not produce any **validation results**, and false otherwise.

Iff validation conforms false, the report will contain an instance of **sh:ValidationResult**.

```
[  
  a sh:ValidationReport ;  
  sh:conforms true ;  
] .
```

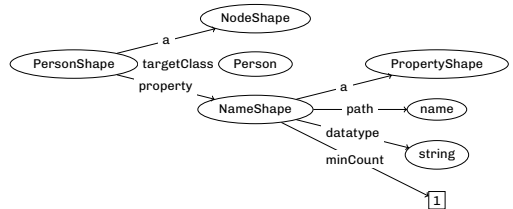
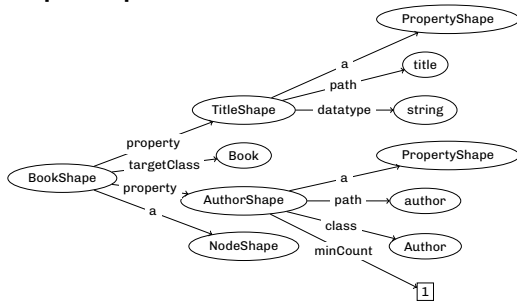
Validation result

All properties described can be specified in a validation result.

sh:focusNode	Node that caused the result.
sh:resultPath	Pointing to value of sh:path
sh:value	Value node that violated constraint.
sh:sourceShape	Shape that given focus node validated against.
sh:sourceConstraintComponent	Constraint component that caused the result.
sh:detail	Parent result containing more details about the violation.
sh:message	Annotation property with textual details.
sh:severity	Default sh:Violation .

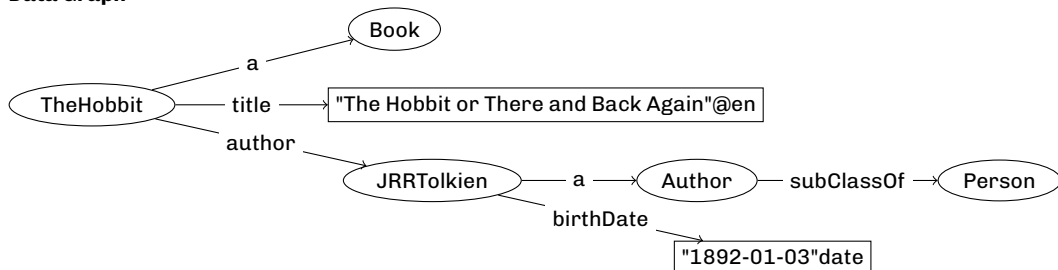
Validation example

Shapes Graph



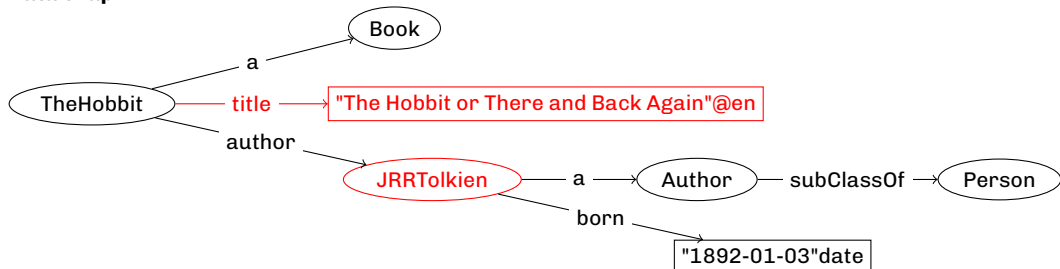
Validation example

Data Graph



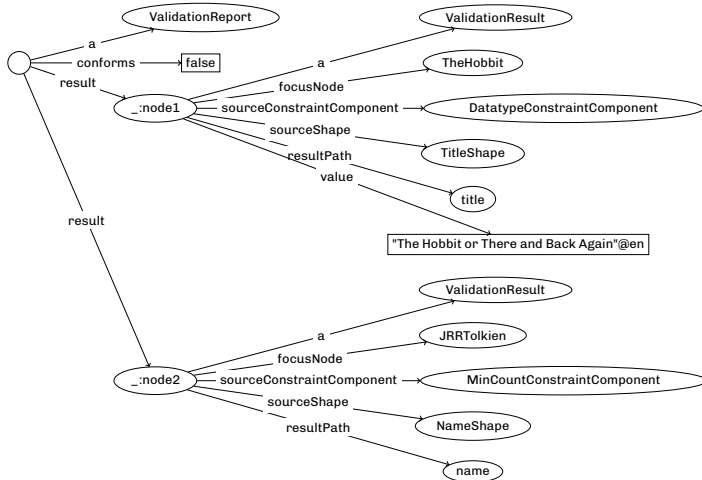
Validation example

Data Graph

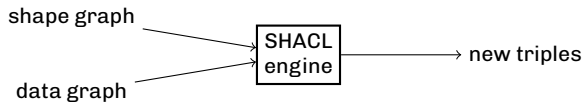


Validation example

Validation Result (RDF4J)



SHACL Advanced Features (SHACL-AF): SHACL Rules



- › Not standardised (yet!)
- › Possible to declare **triple rules** (`sh:TripleRule`) and **SPARQL rules** (`sh:SPARQLRule`).
- › Lightweight vocabulary.
- › Used to derive inferred statements from existing asserted statements.

SHACL Implementations

Frameworks

maplib	https://github.com/DataTreehouse/maplib
ruby-rdf/shacl	https://github.com/ruby-rdf/shacl
dotNetRDF	https://dotnetrdf.org/docs/stable/api/VDS.RDF.Shacl.html
pySHACL	https://github.com/RDFLib/pySHACL
RDF4J	https://rdf4j.org/
Jena	https://jena.apache.org/

Vendors

Most vendors support SHACL, but TopQuadrant, Hanami and Metaphacts are SHACL-first

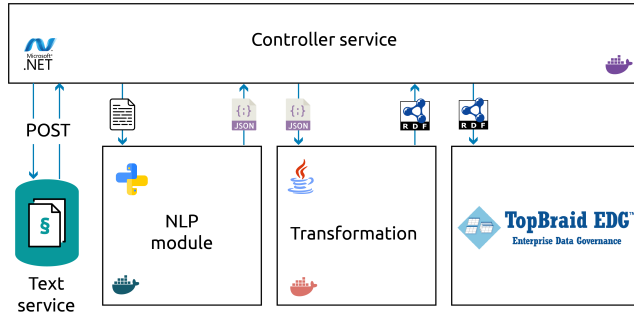
Web playgrounds

SHACL Play!	https://shacl-play.sparna.fr/play/
SHACL Playground	https://shacl.org/playground/
Zazuko SHACL Playground	https://shacl-playground.zazuko.com/

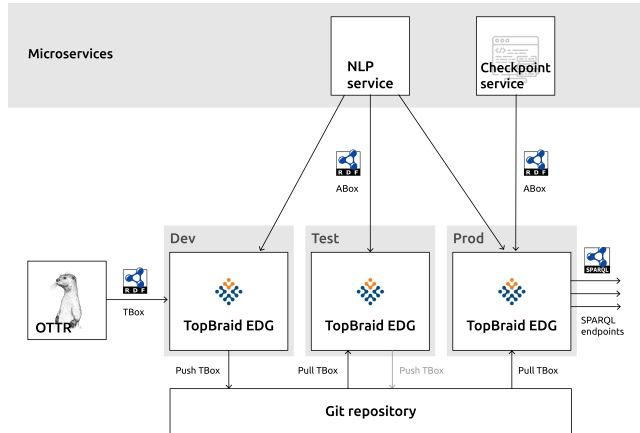
...and more: <https://github.com/w3c-cg/awesome-semantic-shapes>

SHACL Stories

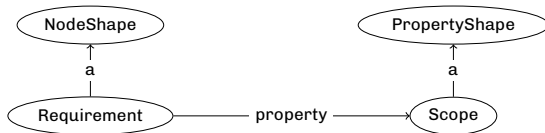
The Norwegian Maritime Authority



The Norwegian Maritime Authority



Regulatory Requirements



- › The **Requirement** is the core at every activity by the Norwegian Maritime Authority.
- › SHACL as verbose vocabulary for describing machine-readable requirements.
- › CWA for the domain of law.

More on reasons why in *Using the Shapes Constraint Language for modelling regulatory requirements* by Veronika Heimsbakk and Kristian Torkelsen, <https://arxiv.org/abs/2309.02723>

Requirement

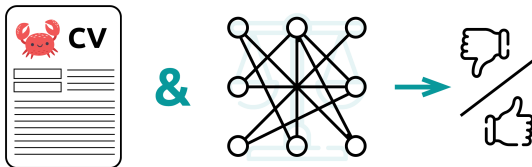
```
sdir:REG20140701955S1P1v0
  a sh:NodeShape ;
  a sdir:Requirement ;
  rdfs:label "Virkeområde"@no ;
  sh:property scope:PS_CargoShip ;
  sh:property scope:PS_ForeignTrade_PassengerShip ;
  sh:property scope:PS_MobileOffshoreUnit ;
  sh:property scope:PS_minLOA_24_LeisureBoat ;
  sdir:chapterTitle "Forskrift om radiokommunikasjonsutstyr for norske skip og flyttbare innretninger"@no ;
  sdir:eliReference "regulation/2014/07/01/955/part/1/chapter/1/section/1/subsection/1" ;
  sdir:generalScopes sdir:REG20140701955S1v0 ;
  sdir:regulationReference "https://lovdata.no/dokument/SF/forskrift/2014-07-01-955/§1" ;
  sdir:regulationTitle "Forskrift om radiokommunikasjonsutstyr for norske skip og flyttbare innretninger"@no ;
  sdir:takeEffect "2024-02-14"^^xsd:date ;
```

Scope

```
scope:PS_CargoShip
  a sh:PropertyShape ;
  a scope:Scope ;
  sh:path sdir:vesselType ;
  sh:description "Scope of vessel type cargo ship"@en ;
  sh:description "Virkeområde fartøytype lasteskip"@no ;
  sh:group scope:Scopes ;
  sh:hasValue vesseltype:CargoShip ;
  sh:name "Fartøytype lasteskip"@en ;
  sh:name "Fartøytype lasteskip"@no ;
  sh:name "Vessel type cargo ship"@en ;
.

scope:PS_minLOA_24_LeisureBoat
  a scope:CompoundScope ;
  sh:description "Compound scope of minLOA 24 LeisureBoat"@en ;
  sh:group scope:CompoundScopes ;
  sh:name "Length overall more than 24 m; vessel type leisure boat"@en ;
  sh:name "Største lengde større enn 24 m; fartøytype fritidsbåt"@no ;
  sh:property scope:PS_LeisureBoat ;
  sh:property scope:PS_minLOA_24 ;
.
```

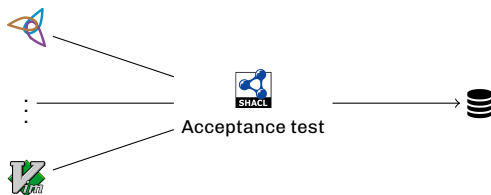
Issue Certificates



```
...
sh:or (
  [ sh:and ( # first alternative
    [ sh:or ( cert:D2A0 cert:D2B0 cert:D3A0
      cert:D3B0 cert:D4B0 cert:D4F0 ) ]
    [ sh:path nma:hasSeagoingServiceRequirement ;
      sh:hasValue nma:SGS_500_1080_D0 ; ]
  )]

  [ sh:and ( # second alternative
    [ sh:or ( cert:D2A0 cert:D2B0
      cert:D3A0 cert:D3B0 ) ]
    [ sh:path nma:hasSeagoingServiceRequirement ;
      sh:hasValue nma:SGS_500_720_D0 ; ]
    [ sh:path nma:hasSeagoingServiceRequirement ;
      sh:hasValue nma:SGS_500_360_C0 ; ]
  )]
) ;
...
```

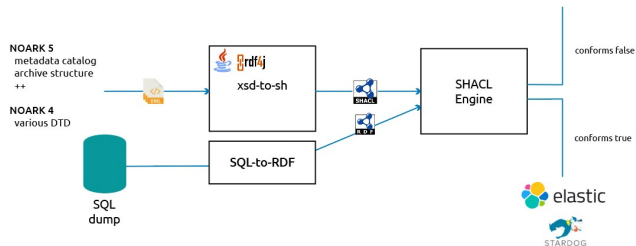

Acceptance Testing



```
:RDFSClassShape a sh:NodeShape ;  
  sh:targetClass rdfs:Class ;  
  sh:property :RDFSLabelShape .  
  
:RDFSLabelShape a sh:PropertyShape ;  
  sh:path rdfs:label ;  
  sh:minCount 1 ; sh:maxCount 1 ;  
  sh:datatype rdf:langString .
```

- › Shapes to validate the *structure* of the TBox.
- › Included in the commit-pipeline, or outside if git is not used.
- › Does not validate the *content* of the graph.

Schema



Journal post snippet

```
<jp/123> a :JournalPost ;  
  :numAttachments 1 ;  
  :documentMedium :ElectronicArchive ;  
  :journalPostNumber "12/123-4"^^xsd:string ;  
  . . . .
```

SHACL snippet

```
[  
  sh:datatype xsd:integer ;  
  sh:maxCount 1 ;  
  sh:predicate :numAttachments ;  
]
```

Fun fact! The SHACL Engine implemented at eInnsyn led to the SHACL Engine for rdf4j.

Demo and data 📌

<https://github.com/veleda/shacl-masterclass>

References & resources

Images

By the author

Around the web

W3C Recommendation

Ivo Velitchkov and Veronika Heimsbakk

Holger Knublauch

W3C Working Group Note

TopQuadrant

\$this

Shape Constraint Language

SHACL Wiki

SHACL and OWL Compared

SHACL Advanced Features

DASH Data Shapes

<https://www.w3.org/TR/shacl/>

<https://kvistgaard.github.io/shacl/>

<https://spinrdf.org/shacl-and-owl.html>

<https://w3c.github.io/shacl/shacl-af/>

<http://datashapes.org/>

<https://github.com/veleda/shacl-masterclass>

Book

Jose Emilio labra Gayo, Eric Prud'hommeaux, Iovka Boneva, Dimitris Kontokostas, *Validating RDF Data*, 2018.

Veronika Heimsbakk, *SHACL for the Practitioner*, pre-print 2025.