ShEx

Testing data with the Shape Expressions Language

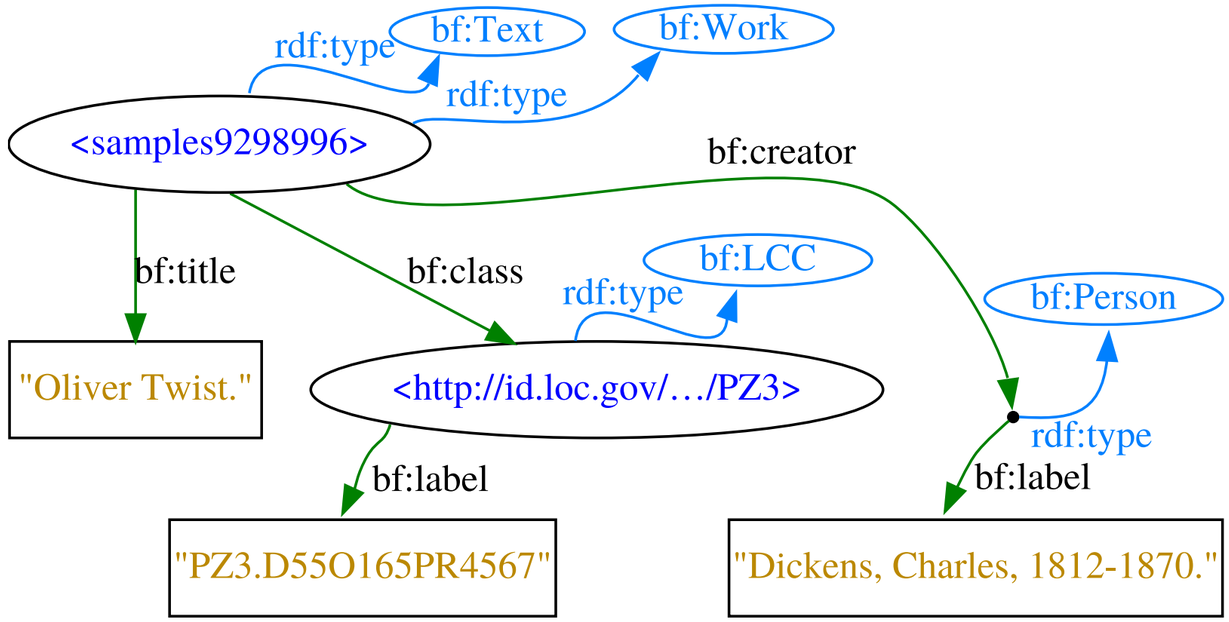
WikidataCon-2017
28 October 2017

http://shexspec.github.io/talks/2017/10-28-wikidatacon/

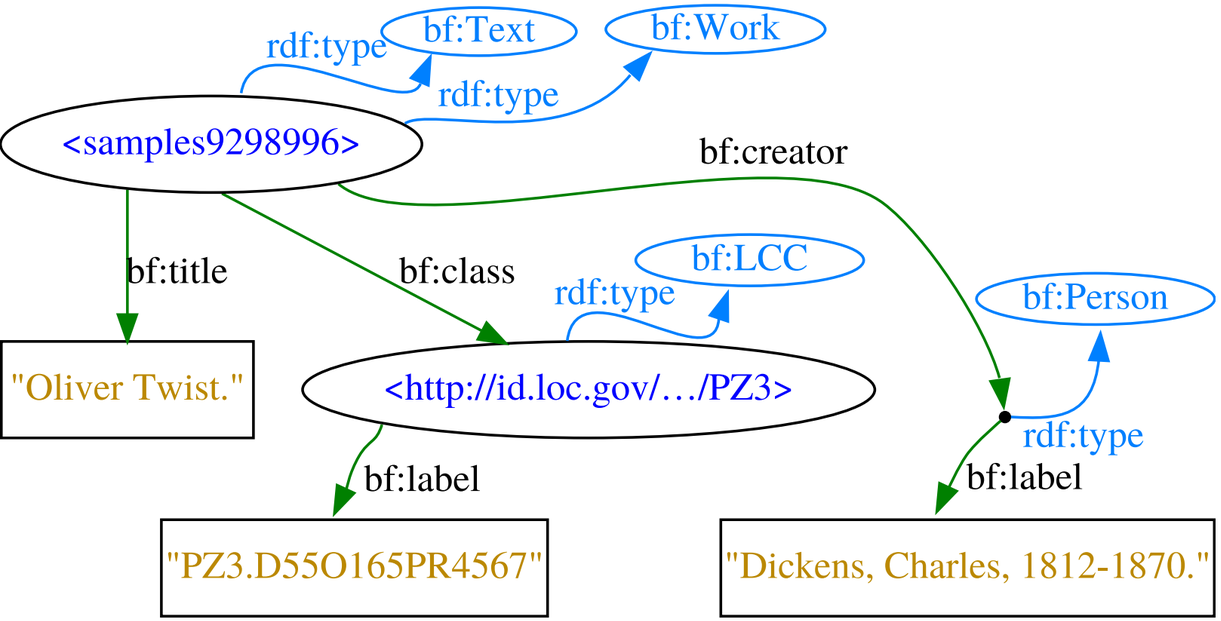
- Andra Waagmeester, <andra@micelio.be>
  - GeneWiki and Micelio
- Eric Prud'hommeaux, <eric@w3.org>
  - Sanitation Engineer, W3C
- Kat Thornton, <katherine.thornton@yale.edu>
  - Yale University
- Lucas Werkmeister, <lucas.werkmeister@wikimedia.de>
  - Wikimedia Deutschland e. V.
- Greg Stupp, <gstupp@scripps.edu>
  - The Scripps Research Institute

Shape Expressions (ShEx)

- a language for describing RDF data ("graphs")
- can be used for:
  - Model development – parallel development of a schema and examples/test cases makes the proposal | testing | refinement process much more concrete. This applies both to initial development and subsequent model refinement
  - Documentation – terse, human-readable representation helping contributors and maintainers quickly grok the model
  - Legacy review – develop punch lists for existing data issues that need fixing
  - Client pre-submission – submitters test their data before submission to make sure they're saying what they want to say and that the receiving schema can accommodate all of their data
  - Server pre-ingestion – submission process checks data as it comes in and either rejects or warns about non-conformant data
  - Generating user interfaces
  - Transforming RDF data into other data formats



RDF Data as graph and as text



Data (Turtle)

```
<samples9298996>
  rdf:type bf:Text ;
  rdf:type bf:Work ;
  bf:title "Oliver Twist." ;
  bf:class <id.loc.gov/.../PZ3> ;
  bf:creator [
    rdf:type bf:Person ;
    bf:label "Dickens, Charles, 1812-1870." ;
  ] .

<id.loc.gov/.../PZ3>
  rdf:type bf:LCC ;
  bf:label "PZ3.D55O165PR4567" .
```

ShEx Model: RDF Data is tested

- **RDF Data** ("graph") is tested
- against a **ShEx Schema**
- to produce a **Validation Result**

Data (Turtle)

```
<samples9298996>
  rdf:type bf:Text ;
  rdf:type bf:Work ;
  bf:title "Oliver Twist." ;
  bf:class <id.loc.gov/.../PZ3> ;
  bf:creator [
    rdf:type bf:Person ;
    bf:label "Dickens, Charles, 1812-1870." ;
  ] .
<id.loc.gov/.../PZ3>
  rdf:type bf:LCC ;
  bf:label "PZ3.D55O165PR4567" .
```

ShEx Model: ...against a ShEx Schema

Schema (ShExC)

```
<Work> EXTRA rdf:type {
  rdf:type [bf:Work] ? ;
  bf:title LITERAL ;
  bf:class @<Classification> * ;
  bf:creator @<Person> OR @<Organization> + ;
  bf:derivedFrom IRI * ;
}

<Classification>
  [<http://id.loc.gov/.../>~]
AND
  EXTRA rdf:type {
    rdf:type [bf:LCC] ? ;
    bf:label LITERAL ;
  }
```

- **RDF Data** ("graph") is tested
- against a **ShEx Schema**
  - which prescribes conditions that RDF graphs must meet:
    - which subjects, predicates, and objects may appear
    - ...in what combinations
    - ...with what cardinalities and datatypes
    - ...in order to produce:
- a **Validation Result**

ShEx Model: ...to produce a Validation Result

- **RDF Data** ("graph") is tested
- against a **ShEx Schema**
- to produce a **Validation Result**
  - ...which flags parts of the data that do not "conform"

```
validating samples9298996bad as Work:
  validating http://...oliverTwist:
    Error validating http://...oliverTwist
    as nodeKind literal:
      iri found when literal expected
```

ShEx Model: ...on the basis of a Shape Map

- **RDF Data** ("graph") is tested
- against a **ShEx Schema**
- to produce a **Validation Result**
- on the basis of a **Shape Map**
  - which designates nodes in **RDF Data** to be tested against **ShEx Schema**
  - and may be generated on the basis of a query.

Fixed Shape Map

```
inst:Alice @ school:Enrollee,
inst:Bob @ school:Enrollee,
inst:Claire @ school:Enrollee,
inst:Don @ school:Enrollee
```

Query Shape Map

```
{FOCUS, foaf:age, _} @ school:Enrollee
```

A ShEx Schema prescribes the "shape" of RDF Data

Schema (ShExC)

```
<Work> EXTRA rdf:type {
  rdf:type [bf:Work] ? ;
  bf:title LITERAL ;
  bf:class @<Classification> * ;
  bf:creator @<Person> OR @<Organization> + ;
  bf:derivedFrom IRI * ;
}

<Classification>
  [<http://id.loc.gov/.../>~]
AND
  EXTRA rdf:type {
    rdf:type [bf:LCC] ? ;
    bf:label LITERAL ;
  }
```

A resource in the RDF data matching the "Work" shape:

- has zero or one "rdf:type bf:Work" statements
- optionally has an extra "rdf:type" statement
- has exactly one "bf:title" statement with a literal value
- has zero or more "bf:class" statements taking objects that match the "Classification" shape
- has one or more "bf:creator" statements taking objects that match either the "Person" or the "Organization" shape
- and zero or more "bf:derivedFrom" statements taking an IRI as object

A ShEx Schema prescribes the "shape" of RDF Data

Schema (ShExC)

```
<Work> EXTRA rdf:type {
  rdf:type [bf:Work] ? ;
  bf:title LITERAL ;
  bf:class @<Classification> * ;
  bf:creator @<Person> OR @<Organization> + ;
  bf:derivedFrom IRI * ;
}

<Classification>
  [<http://id.loc.gov/.../>~]
AND
  EXTRA rdf:type {
    rdf:type [bf:LCC] ? ;
    bf:label LITERAL ;
  }
```

A resource matching the "Classification" shape:

- is identified with an IRI starting with "http://id.loc.gov..."
- has zero or one "rdf:type bf:LOC" statements
- optionally has an additional "rdf:type" statement
- has exactly one "bf:label" statement with a literal value."

A ShEx Schema is tested against RDF Data

Schema (ShExC)

```
<Work> EXTRA rdf:type {
  rdf:type [bf:Work] ? ;
  bf:title LITERAL ;
  bf:class @<Classification> * ;
  bf:creator @<Person> OR @<Organization> + ;
  bf:derivedFrom IRI * ;
}

<Classification>
  [<http://id.loc.gov/.../>~]
AND
  EXTRA rdf:type {
    rdf:type [bf:LCC] ? ;
    bf:label LITERAL ;
  }
```

Data (Turtle)

```
<samples9298996>
  rdf:type bf:Text ;
  rdf:type bf:Work ;
  bf:title "Oliver Twist." ;
  bf:class <id.loc.gov/.../PZ3> ;
  bf:creator [
    rdf:type bf:Person ;
    bf:label "Dickens, Charles, 1812-1870." ;
  ] .

<id.loc.gov/.../PZ3>
  rdf:type bf:LCC ;
  bf:label "PZ3.D55O165PR4567" .
```

Shapes can specify use of properties

A <Work> must have exactly one bf:title with value LITERAL:

Schema (ShExC)

```
<Work> {
  bf:title LITERAL ;
}
```

Data (Turtle)

```
<samples9298996>
  bf:title "Oliver Twist." .

<samples9298996bad>
  bf:title <http://...oliverTwist> .
```

try it

✓<samples9298996>@<Work>

✗<samples9298996bad>@!<Work>

validating samples9298996bad as Work:
validating http://...oliverTwist:
Error validating http://...oliverTwist
as nodeKind literal:
iri found when literal expected

one passed, one failed.

- <Work> { ... } defines a "shape".
- bf:title LITERAL expects an bf:title property with a literal value.
- The default cardinality is min: 1, max: 1.

Shapes can specify types

Permit some data to have a type arc identifying it as a <Work>:

Schema (ShExC)

```
<Work> {
  rdf:type [bf:Work] ? ;
  bf:label LITERAL ;
}
```

Data (Turtle)

```
<samples9298996> a bf:Work ;
  bf:title "Oliver Twist." .

<samples9298996b> a bf:Work ;
  bf:title "Oliver Twist." .

<samples9298996bad> a bf:Krow ;
  bf:title "Oliver Twist" .
```

try it

✓<samples9298996>@<Work>

✓<samples9298996b>@<Work>

✗<samples9298996bad>@!<Work>

validating samples9298996bad as Work:
validating http://bibframe.org/vocab/Krow:
Error validating http://bibframe.org/vocab/Krow
as values [<http://bibframe.org/vocab/Work>]:
value http://bibframe.org/vocab/Krow not found in set
[<http://bibframe.org/vocab/Work>]

- []s denote a value set.
- This value set as only one permissible value: bf:Work.
- The '?' code means min: 0, max: 1 (like regular expressions).

Shapes can reference other shapes

Add a reference to another shape:

Schema (ShExC)

```
<Work> {
  rdf:type [bf:Work] ? ;
  bf:label LITERAL ;
  bf:class @<Classification> * ;
}

<Classification> {
  rdf:type [bf:LCC] ? ;
  bf:label LITERAL ;
}
```

Data (Turtle)

```
<samples9298996>
  bf:title "Oliver Twist." ;
  bf:class <http://id.loc.gov/.../PZ3> .
<http://id.loc.gov/.../PZ3> a bf:LCC ;
  bf:label "PZ3.D55O165PR4567" .

<samples9298996b>
  bf:title "Oliver Twist." ;
  bf:class [ bf:label "PZ3.D55O165PR4567" ].

<samples9298996bad>
  bf:title "Oliver Twist." ;
  bf:class [ a bf:LCD ; bf:label "PZ3.D55O165PR4567" ].
```

try it

✓<samples9298996>@<Work>

✓<samples9298996b>@<Work>

✗<samples9298996bad>@!<Work>

validating samples9298996bad as Work:
validating _:b1:
validating http://bibframe.org/vocab/LCD:,
Error validating http://bibframe.org/vocab/LCD
as values: [<http://bibframe.org/vocab/LCC>]}:
value <http://bibframe.org/vocab/LCD> not found
in set [<http://bibframe.org/vocab/LCC>]

- @<Classification> references a different shape.
- The '*' code means min: 0, max: * (like regular expressions).

Shapes can specify value sets

Or a reference to an authority:

Schema (ShExC)

```
<Work> {
  rdf:type [bf:Work] ? ;
  bf:label LITERAL ;
  bf:class @<Classification> * ;
}

<Classification>
  [<http://id.loc.gov/.../>~]
```

Data (Turtle)

```
<samples9298996>
  bf:title "Oliver Twist." ;
  bf:class <http://id.loc.gov/.../PZ3> .

<samples9298996bad>
  bf:title "Oliver Twist." ;
  bf:class <http://id.loc.gov/authorities/PZ3> .

<samples9298996badb>
  bf:title "Oliver Twist." ;
  bf:class [ bf:label "PZ3.D55O165PR4567" ].
```

try it

✓<samples9298996>@<Work>

✗<samples9298996bad>@!<Work>

validating samples9298996bad as Work:
validating <http://id.loc.gov/authorities/PZ3>:
NodeConstraintError: expected to match
[<http://id.loc.gov/authorities/classification/>~]

✗<samples9298996badb>@!<Work>

validating samples9298996bad as Work:
validating _:b0:
NodeConstraintError: expected to match
[<http://id.loc.gov/authorities/classification/>~]

[]s denote a value set.

This value set includes every IRI starting with <http://id.loc.gov/.../>.

Shapes can combine constraints

Or a reference to a structure identifed by and authority:

Schema (ShExC)

```
<Work> {
  rdf:type [bf:Work] ? ;
  bf:label LITERAL ;
  bf:class @<Classification> * ;
}

<Classification>
  [<http://id.loc.gov/.../>~]
  AND {
    rdf:type [bf:LCC] ? ;
    bf:label LITERAL ;
  }
```

Data (Turtle)

```
<samples9298996>
  bf:title "Oliver Twist." ;
  bf:class <http://id.loc.gov/.../PZ3> .
<http://id.loc.gov/.../PZ3> a bf:LCC ;
  bf:label "PZ3.D55O165PR4567" .

<samples9298996bad>
  bf:title "Oliver Twist." ;
  bf:class <http://id.loc.gov/authorities/PZ3> .
<http://id.loc.gov/authorities/PZ3> a bf:LCC ;
  bf:label "PZ3.D55O165PR4567" .

<samples9298996badb>
  bf:title "Oliver Twist." ;
  bf:class <http://id.loc.gov/.../PZ3999> .
```

try it

✓<samples9298996>@<Work>

✗<samples9298996bad>@!<Work>

validating samples9298996bad as Work:
validating >http://id.loc.gov/authorities/PZ3>:
validating <http://id.loc.gov/authorities/PZ3> as Classification:
NodeConstraintError: expected to match [<http://id.loc.gov/.../>~]

✗<samples9298996badb>@!<Work>

validating samples9298996bad as Work:
validating <http://id.loc.gov/.../PZ3999>:
validating <http://id.loc.gov/.../PZ3999> as Classification:
Missing property: <http://bibframe.org/vocab/label>

- Constraints can be combined with AND, OR, NOT and ()s.

Shapes can specify choices

You may accept multiple forms of creator.

Schema (ShExC)

```
<Work> {
  rdf:type [bf:Work] ? ;
  bf:label LITERAL ;
  bf:class . * ;
  bf:creator @<Person> OR @<Organization> + ;
}

<Person> {
  rdf:type [bf:Person] ? ;
  bf:label LITERAL ;
}

<Organization> { EXTRA rdf:type
  rdf:type [bf:Organization] ;
  bf:label LITERAL ;
  org:member @<Person> OR @<Organization> *
}
```

Data (Turtle)

```
<samples9298996>
  bf:title "Oliver Twist." ;
  bf:creator [ rdf:type bf:Person ;
    bf:label "Dickens, Charles, 1812-1870."
  ] .

<wp-55-45-1>
  bf:title "Nixon blasts 'false charges'" ;
  bf:creator <WP> .

<WP> rdf:type bf:Organization , bf:Newpaper ;
  bf:label "Washington Post" ;
  org:member <BobWoodward>, <CarlBernstein> .

<BobWoodward> bf:label "Bob Woodward" .
<CarlBernstein> bf:label "Carl Bernstein" .
```

try it

✓<samples9298996>@<Work>

✓<wp-55-45-1>@<Work>

Diseases in Wikidata

Schema (ShExC)

```
# Shape Expression for Diseases in Wikidata
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX p: <http://www.wikidata.org/prop/>
PREFIX ps: <http://www.wikidata.org/prop/statement/>
PREFIX pq: <http://www.wikidata.org/prop/qualifier/>
PREFIX pr:  <http://www.wikidata.org/prop/reference/>
PREFIX prov: <http://www.w3.org/ns/prov#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX do: <http://purl.obolibrary.org/obo/DOID_>

start = @<wikidata-disease>

<wikidata-disease> {
  p:P31 { # instance of disease
    ps:P31 [ wd:Q12136 ]; # instance of disease
    $<has-do-reference> prov:wasDerivedFrom @<do-reference>;
  } ;
  p:P279 { # subclass of
    ps:P279 @<wikidata-disease>;
    &<has-do-reference>
  } * ;
  p:P2888 EXTRA prov:wasDerivedFrom { # exact match
    ps:P2888 [ do:~ ];
    prov:wasDerivedFrom @<do-reference> ?
  } + ;
}

<do-reference> {
  # stated in
  pr:P248 @<version-disease-ontology> ;
  # retrieved
  pr:P813 xsd:dateTime ;
  # Disease ontology ID
  pr:P699 @<disease-ontology-id> ;
}

<disease-ontology-id> LITERAL /^DOID:[0-9]+$/

<version-disease-ontology> {
  # edition or translation of Disease Ontology
  p:P629 { ps:P629 [ wd:Q5282129 ] } ;
```

Wikidata items on Cancer should have an NCI Thesaurus ID

Schema (ShExC)

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX prov: <http://www.w3.org/ns/prov#>
PREFIX p: <http://www.wikidata.org/prop/>
PREFIX pr: <http://www.wikidata.org/prop/reference/>
PREFIX ps: <http://www.wikidata.org/prop/statement/>

start = @<wikidata_item>

<wikidata_item> {
  p:P1748 {
    ps:P1748 LITERAL ;
    prov:wasDerivedFrom @<reference>
  }+
}

<reference> {
  pr:P248  IRI ;
  pr:P813  xsd:dateTime ;
  pr:P699  LITERAL
}
```

```
Endpoint: https://query.wikidata.org/bigdata/namespace/wdq/sparql

Query: SELECT ?item ?itemLabel
WHERE
{ ?item wdt:P279* wd:Q12078 .
  SERVICE wikibase:label { bd:serviceParam wikibase:language "en" }
} LIMIT 10
```

Try it!

Click “Wikidata item on Cancer should have a NCI Thesaurus ID” on the left,
then “Get all Wikidata items on Cancers (SPARQL)” on the right.

Three interchangeable concrete syntaxes

- **ShExC** (used in this talk)
  - a compact syntax meant for human eyes and fingers
- **ShExJ**
  - a JSON-LD Javascript syntax meant for machine processing
- **ShExR**
  - the RDF interpretation of ShExJ expressed in RDF Turtle syntax

Open-source implementations

- Javascript (mature)
- Ruby (mature)
- Scala (mature)
- Java
- Python
- Also:
  - Extensive test suite (2000+)
  - Sublime Text plugin
  - Jupyter Notebook kernel (in the pipeline)

Shapes Constraint Language (SHACL)

- similar to ShEx.
- both have logical expressions (**AND**, **OR**, **NOT**).
- uses a SPARQL rules language for constructing new tests.
- implemented in a commercial IDE (TopBraid Composer).

ShEx and SHACL compared

- ShEx is like XML Schema. SHACL is like Schematron.
- SHACL evaluates expressions individually.
- ShEx examines shape expressions as a whole for:
  - increased expressivity - cardinality on groups.
    (data:sourceURL IRI; data:contentType xsd:string)?
  - consistent interpretation of **OR**, **CLOSED**.
    foaf:name LITERAL | foaf:givenName LITERAL; foaf:familyName LITERAL
  - support for recursive shapes and data.
    <Person> { foaf:knows @<Person> }
  - simple constraints on repeated properties.
    fhir:component { fhir:code [sct:systolic]; fhir:value xsd:integer };
    fhir:component { fhir:code [sct:diastolic]; fhir:value xsd:integer };
- detailed comparison in book "Validating RDF Data" (http://a.co/fD3UbCw)
  - See examples at https://github.com/labra/validatingRDFBookExamples/

Join the ShEx Community!

W3C Shape Expressions Community Group

- ShEx homepage — http://shex.io
- community group page — https://www.w3.org/community/shex
- ShEx Primer — http://shex.io/shex-primer/
- ShEx Specification — http://shex.io/shex-semantics/
- ShapeMap Specification — shex.io/shape-map/
- Gitter discussions — https://gitter.im/shapeExpressions/Lobby