ShExStatements

Mar 06, 2021

Table of Contents:

ShExStatements allows the users to generate shape expressions from simple CSV statements, CSV files and Spreadsheets. `shexstatements` can be used from the command line as well as from the web interface.

CHAPTER 1

Quick start

1.1 Using pip

Set up a virtual environment and install `shexstatements`.

```
$ python3 -m venv .venv
$ source ./.venv/bin/activate
$ pip3 install shexstatements
```

Run the following command with an example CSV file. The file contains an example description of a language on Wikidata. This file uses comma as a delimiter to separate the values.

```
$ shexstatements.sh examples/language.csv
```

CHAPTER 2

Build from source

2.1 Terminal

Clone the **ShExStatements** repository.

```
$ git clone https://github.com/johnsamuelwrites/ShExStatements.git
```

Go to **ShExStatements** directory.

```
$ cd ShExStatements
```

Install modules required by **ShExStatements** (here: installing into a virtual environment).

```
$ python3 -m venv .venv
$ source ./.venv/bin/activate
$ pip3 install .
```

Run the following command with an example CSV file. The file contains an example description of a language on Wikidata. This file uses comma as a delimiter to separate the values.

```
$ ./shexstatements.sh examples/language.csv
```

CSV file can use delimiters like *;*. Take for example, the following command works with a file using semi-colon as a delimiter.

```
$ ./shexstatements.sh examples/languagedelimsemicolon.csv --delim ";"
```

But sometimes, users may like to specify the header. In that case, they can make use of `-s` or `--skipheader` to tell the generator to skip the header (first line of CSV).

```
$ ./shexstatements.sh --skipheader examples/header/languageheader.csv
```

It is also possible to work with Spreadsheet files like .ods, .xls or .xlsx.

```
$ shexstatements.sh examples/language.ods
```

```
$ shexstatements.sh examples/language.xls
```

```
$ shexstatements.sh examples/language.xlsx
```

In all the above cases, the shape expression generated by **ShExStatements** will look like

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
start = @<language>
<language> {
  wdt:P31 [ wd:Q34770  ] ;# instance of a language
  wdt:P1705 LITERAL ;# native name
  wdt:P17 .+ ;# spoken in country
  wdt:P2989 .+ ;# grammatical cases
  wdt:P282 .+ ;# writing system
  wdt:P1098 .+ ;# speakers
  wdt:P1999 .* ;# UNESCO language status
  wdt:P2341 .+ ;# indigenous to
}
```

Use `-j` or `--shexj` to generate ShEx JSON Syntax (ShExJ) instead of default ShEx Compact syntax (ShExC).

```
$ ./shexstatements.sh --shexj examples/language.csv
```

The output will be similiar to:

```
{
  "type": "Schema",
  "start": "language",
  "shapes": [
    {
      "type": "Shape",
      "id": "language",
      "expression": {

      }
    }
  ]
}
```

It's also possible to use application profiles of the following form

```
Entity_name,Property,Property_label,Mand,Repeat,Value,Value_type,Annotation
```

and Shape expressions can be generated using the following form

```
$ ./shexstatements.sh -ap --skipheader examples/languageap.csv
```

CHAPTER 3

Objectives

- Easily generate shape expressions (ShEx) from CSV files and Spreadsheets
- Simple syntax

CHAPTER 4

Documentation and examples

A detailed documentation can be found here. with a number of example CSV files in the examples folder.

CHAPTER 5

Test cases and coverage

All the test cases can be run in the following manner

```
$ python3 -m tests.tests
```

Code coverage report can also be generated by running the unit tests using the coverage tool.

```
$ coverage run --source=shexstatements -m unittest tests.test
$ coverage report -m
```

5.1 Web interface

`shexstatements` can also be accessed from a web interface. Clone the **ShExStatements** repository.

```
$ git clone https://github.com/johnsamuelwrites/ShExStatements.git
```

Go to **ShExStatements** directory.

```
$ cd ShExStatements
```

Install modules required by **ShExStatements** (here: installing into a virtual environment).

```
$ python3 -m venv .venv
$ source ./.venv/bin/activate
$ pip3 install .
```

Now run the application.

```
$ ./shexstatements.sh -r
```

Check the URL `http://127.0.0.1:5000/`

CHAPTER 6

API

ShExStatements also has an API to generate ShEx from CSV and is described here.

CHAPTER 7

Demonstration

Online demonstrations are also available:

- https://shexstatements.toolforge.org/
- https://tools.wmflabs.org/shexstatements/

CHAPTER 8

Author

- John Samuel
- Contributors

CHAPTER 9

Acknowledgements

- Wikidata Community

CHAPTER 10

Archives and Releases

- Zenodo
- Software Heritage
- Release Notes

CHAPTER 11

Licence

All code are released under GPLv3+ licence. The associated documentation and other content are released under CC-BY-SA.

CHAPTER 12

Contents

12.1 ShExStatements: Documentation

ShExStatements allows the users to generate shape expressions from simple CSV statements and files. `shexstatements` can be also be used from the command line.

12.1.1 Objectives

- Easily generate shape expressions (ShEx) from CSV files
- Simple syntax, with 5 columns
- Node name
- Property
- Allowed values
- Cardinality (optional)
- Comments (optional)

12.1.2 Quick start

Clone the **ShExStatements** repository.

```
$ git clone https://github.com/johnsamuelwrites/ShExStatements.git
```

Go to **ShExStatements** directory.

```
$ cd ShExStatements
```

Install modules required by **ShExStatements** (here: installing into a virtual environment).

```
$ python3 -m venv .venv
$ source ./.venv/bin/activate
$ pip3 install .
```

Run the following command with an example CSV file. The file contains an example description of a language on Wikidata. This file uses comma as a delimiter to separate the values.

```
$ ./shexstatements.sh examples/language.csv
```

There are five columns in the CSV file. Column 1 is used for specifying the node name, 2 for specifying the property value, 3 for possible values, 4 for cardinality (+,*) and column 5 for comments. Comments start with #. Columns 1, 2, 3 are mandatory. Column 3 can be a special value like . (period to say 'any' value). Columns 3,4 and 5 are empty for prefixes.

Cardinality can be any one of the following values

- * : zero or more values
- + : one or more values
- m : m number of values
- m,n : any number of values between m and n (including m and n).

CSV file can use delimiters like *;*. Take for example, the following command works with a file using semi-colon as a delimiter.

```
$ ./shexstatements.sh examples/languagedelimsemicolon.csv --delim ";"
```

But sometimes, users may like to specify the header. In that case, they can make use of `-s` or `--skipheader` to tell the generator to skip the header (firsrt line of CSV).

```
$ ./shexstatements.sh --skipheader examples/header/languageheader.csv
```

In all the above cases, the shape expression generated by **ShExStatements** will look like

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
start = @<language>
<language> {
  wdt:P31 [ wd:Q34770  ] ;# instance of a language
  wdt:P1705 LITERAL ;# native name
  wdt:P17 .+ ;# spoken in country
  wdt:P2989 .+ ;# grammatical cases
  wdt:P282 .+ ;# writing system
  wdt:P1098 .+ ;# speakers
  wdt:P1999 .* ;# UNESCO language status
  wdt:P2341 .+ ;# indigenous to
}
```

Use `-j` or `--shexj` to generate ShEx JSON Syntax (ShExJ) instead of default ShEx Compact syntax (ShExC).

```
$ ./shexstatements.sh --shexj examples/language.csv
```

The outpul will be similiar to:

```
{
  "type": "Schema",
  "start": "language",
  "shapes": [
    {
      "type": "Shape",
      "id": "language",
      "expression": {
        "type": "EachOf",
        "expressions": [
          {
            "type": "TripleConstraint",
            "predicate": "http://www.wikidata.org/prop/direct/P31",
            "valueExpr": {
              "type": "NodeConstraint",
              "values": [
                "http://www.wikidata.org/entity/Q34770"
              ]
            }
          },
          {
            "type": "TripleConstraint",
            "predicate": "http://www.wikidata.org/prop/direct/P1705",
            "valueExpr": {
              "type": "NodeConstraint",
              "nodeKind": "literal"
            }
          },
          {
            "type": "TripleConstraint",
            "predicate": "http://www.wikidata.org/prop/direct/P17",
            "min": 1,
            "max": -1
          },
          {
            "type": "TripleConstraint",
            "predicate": "http://www.wikidata.org/prop/direct/P2989",
            "min": 1,
            "max": -1
          },
          {
            "type": "TripleConstraint",
            "predicate": "http://www.wikidata.org/prop/direct/P282",
            "min": 1,
            "max": -1
          },
          {
            "type": "TripleConstraint",
            "predicate": "http://www.wikidata.org/prop/direct/P1098",
            "min": 1,
            "max": -1
          },
          {
            "type": "TripleConstraint",
            "predicate": "http://www.wikidata.org/prop/direct/P1999",
            "min": 0,
            "max": -1
```


(continues on next page)

(continued from previous page)

```
          },
          {
            "type": "TripleConstraint",
            "predicate": "http://www.wikidata.org/prop/direct/P2341",
            "min": 1,
            "max": -1
          }
        ]
      }
    }
  ]
}
```

It's also possible to use application profiles of the following form

```
Entity_name,Property,Property_label,Mand,Repeat,Value,Value_type,Annotation
```

and Shape expressions can be generated using the following form

```
$ ./shexstatements.sh -ap --skipheader examples/languageap.csv
```

There are example CSV files in the examples folder.

12.2 ShExStatements API

12.2.1 Operations

ShExStatements has also a public API that can be easily accessible both on a local installation as well as on the public interface. It has one operation that takes as input a JSON array with two elements as given below:

- delimiter
- CSV (every line should be terminated by \n)

It returns a JSON array with one element containing the ShEx (shape expression).

12.2.2 Example JSON input

Take for example the file `tvseries.json` (also present in `examples/api/tvseries.json`). It is an array with two elements.

```
[
"|",
"wd|<http://www.wikidata.org/entity/>|||\n
wdt|<http://www.wikidata.org/prop/direct/>|||\n
xsd|<http://www.w3.org/2001/XMLSchema#>|||\n
\n
@tvseries|wdt:P31|wd:Q5398426|# instance of a tvseries\n
@tvseries|wdt:P136|@genre|*|# genre\n
@tvseries|wdt:P495|.|+|#country of origin\n
@tvseries|wdt:P57|.|+|#director\n
@tvseries|wdt:P58|.|+|#screenwriter\n
@genre|wdt:P31|wd:Q201658,wd:Q15961987|#instance of genre\n"
]
```

12.2.3 Calling ShExStatements API

Following is the way to call the ShExStatements API

```
$ curl -s http://127.0.0.1:5000/ -X POST -H "Accept: application/json"  --data ␣
↪@examples/api/tvseries.json |sed 's/\\n/\n/g'
```

or

```
$ curl -s https://shexstatements.toolforge.org/ -X POST -H "Accept: application/json"␣
↪ --data  @examples/api/tvseries.json |sed 's/\\n/\n/g'
```

12.2.4 Example JSON output response

It gives the following output. For the output, the above command makes use of `sed`.

```
"PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
start = @<tvseries>
<tvseries> {
  wdt:P31 [ wd:Q5398426  ] ;# instance of a tvseries
  wdt:P136 @<genre>* ;# genre
  wdt:P495 . ;#country of origin
  wdt:P57 . ;#director
  wdt:P58 . ;#screenwriter
}
<genre> {
  wdt:P31 [ wd:Q201658 wd:Q15961987  ] ;#instance of genre
}
"
```

12.3 shexstatements package

It's also possible to use `shexstatements` in Python programs. This page gives a complete detail of the different modules that can be used in the programs.

In our first example, we take a look at the method `generate_shex_from_csv`, which takes as input a CSV file containing shexstatements and a delimiter. In this example, we use "," as a delimiter.

```
from shexstatements.shexfromcsv import CSV

shex = CSV.generate_shex_from_csv("language.csv", delim=",")
print(shex)
```

In our second example, we use a data string consisting of shexstatements and make use of the function `generate_shex_from_data_string`. Note here, that we use "|" as a delimiter.

```
from shexstatements.shexfromcsv import CSV

shexstatements="""
wd|<http://www.wikidata.org/entity/>
wdt|<http://www.wikidata.org/prop/direct/>
```

(continues on next page)

(continued from previous page)

```
xsd|<http://www.w3.org/2001/XMLSchema#>

@language|wdt:P31|wd:Q34770|# instance of a language
@language|wdt:P1705|LITERAL|# native name
@language|wdt:P17|.|+|# spoken in country
@language|wdt:P2989|.|+|# grammatical cases
@language|wdt:P282|.|+|# writing system
@language|wdt:P1098|.|+|# speakers
@language|wdt:P1999|.|*|# UNESCO language status
@language|wdt:P2341|.|+|# indigenous to
"""

shex = CSV.generate_shex_from_data_string(shexstatements)
print(shex)
```

12.3.1 Submodules

shexstatements.shexfromcsv

class shexstatements.shexfromcsv.**CSV**

This class contains functions that can be used to generate ShEx from a data string or a CSV file.

static generate_shex_from_csv(*filepath*, *delim='\t'*, *skip_header=False*, *filename=True*)

This method can be used to generate ShEx from data string. However, the input data string must contain one or more lines. Each line contains 'l' separated values. If filepath is a string, filename should be set to false.

Parameters

- **filepath** (*str*) – This parameter can contain either a file path of a CSV file or shexstatements in CSV format.
- **delim** (*str*) – a delimiter. Allowed values include ',', 'l' and ';'
- **skip_header** (*bool*) – if the first line is a header, set this value to True. By default, the value is False.
- **filename** (*bool*) – if 'filepath' is a string, then this filename must be set to False

Returns shape expression

Return type shex

static generate_shex_from_data_string(*data*)

This method can be used to generate ShEx from data string. However, the input data string must contain one or more lines. Each line contains 'l' separated values.

Parameters **data** (*str*) – shexstatements in CSV format, using "l" as a delimiter.

Returns shape expression

Return type shex

shexstatements.shexjfromcsv

class shexstatements.shexjfromcsv.**ShExJCSV**

This class contains functions that can be used to generate ShExJ from a ShEx.

static generate_shexj_from_csv(*filepath*, *delim=','*, *skip_header=False*)
This method can be used to generate ShExJ from ShExStatements CSV file

Parameters

- **filepath** (*str*) – This parameter can contain either a file path of a CSV file or shexstatements in CSV format.
- **delim** (*str*) – a delimiter. Allowed values include ',', '|' and ';'
- **skip_header** (*bool*) – if the first line is a header, set this value to True. By default, the value is False.

Returns shape expression in JSON format (ShExJ)

Return type shexj

static generate_shexj_from_shexstament(*shexstatement*)
This method can be used to generate ShEx from data string. However, the input data string must contain one or more lines. Each line contains '|' separated values.

Parameters **shexstatement** (*str*) – shex

Returns shape expression in JSON (ShExJ)

Return type shexj

shexstatements.shexfromapplprofilecsv

class shexstatements.shexfromapplprofilecsv.**ApplicationProfile**
This class contains functions that can be used to generate ShEx from a data string or CSV application profile file.

generate_shex_from_csv(*delim=','*, *skip_header=False*)
This method can be used to generate ShEx from application profile CSV file. However, the input file must contain one or more lines. Each line contains '|' separated values. If filepath is a string, filename should be set to false.

Parameters

- **filepath** (*str*) – This parameter can contain either a file path of a CSV file or shexstatements in CSV format.
- **delim** (*str*) – a delimiter. Allowed values include ',', '|' and ';'
- **skip_header** (*bool*) – if the first line is a header, set this value to True. By default, the value is False.

Returns shape expression

Return type shex

shexstatements.errors

exception shexstatements.errors.**ParserError**(*message*)

exception shexstatements.errors.**UnrecognizedCharacterError**(*message*)

CHAPTER 13

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index