wensleydale Documentation

Release

Rishi Ramraj

May 23, 2017

Contents

Mr Wensleydale. Query Python, get the AST as JSON.

CHAPTER 1

Why?

The AST, or abstract syntax tree, is a set of data structures that describe your python script. By exposing the AST to json, it can be treated as data, which means it can be reported on.

Sample use cases include:

- Visualizing the routing tree of a web application.
- Visualizing a class hierarchy.
- Auditing large code bases using queries.

CHAPTER 2

Support

Currently only Python 3.5 is supported.

Upcoming features:

- *Recursive imports:* Trace into calls.
- *Pythonic paths:* Use python dotted paths instead of file paths.

CHAPTER 3

Contents

Changes

0.1.0

Changes:

- Add script to convert AST to JSON.

Documentation:

- Add getting started.
- Add samples.
- Add contributors.

Getting Started

Installation

Install wensleydale using pip:

```
$ pip install wensleydale
```

Usage

```
Usage: wensleydale [OPTIONS] PATH

  Mr Wensleydale. Query Python, get the AST as JSON.
```

```
Options:
  --version  Show the version and exit.
  --help     Show this message and exit.
```

If we have the following script in a file called *test.py*:

```
print('Hello world!')
```

We can run wensleydale to get the AST, and use *jq* to pretty print the result:

```
$ wensleydale test.py | jq
{
  "body": [
    {
      "col_offset": 0,
      "value": {
        "col_offset": 0,
        "args": [
          {
            "col_offset": 6,
            "s": "Hello world!",
            "lineno": 1,
            "classname": "Str"
          }
        ],
        "lineno": 1,
        "func": {
          "col_offset": 0,
          "lineno": 1,
          "id": "print",
          "ctx": {
            "classname": "Load"
          },
          "classname": "Name"
        },
        "keywords": [],
        "classname": "Call"
      },
      "lineno": 1,
      "classname": "Expr"
    }
  ],
  "classname": "Module"
}
```

Understanding the AST

The *classname* property of the reported dictionaries will map to the Abstract Grammar of Python's syntax tree.

To get a full list of class names, using the following jq query:

```
$ wensleydale test.py | jq '.. | .classname?' | sort | uniq
"Module"
"Expr"
"Call"
"Str"
```

```
"Name"
"Load"
```

You can then select details of individual grammars using:

```
$ wensleydale test.py | jq '.. | select(.classname? == "Call")'
{
  "args": [
    {
      "col_offset": 6,
      "s": "Hello world!",
      "lineno": 1,
      "classname": "Str"
    }
  ],
  "func": {
    "id": "print",
    "col_offset": 0,
    "lineno": 1,
    "ctx": {
      "classname": "Load"
    },
    "classname": "Name"
  },
  "keywords": [],
  "col_offset": 0,
  "classname": "Call",
  "lineno": 1
}
```

Samples

We are always looking for useful queries. If you find one, please shoot us a *pull request*:

Test Code

The samples in this document all query the following Python code:

```
import this as python


def main():
    '''
    Print Hello World!
    '''
    assert python
    print('Hello world!')


if __name__ == '__main__':
    main()
```

Finding Calls

To find the list of function calls in this code:

```
$ wensleydale test.py | jq '.. | select(.classname? == "Call") | {name: .func.id,␣
↪lineno: .lineno}'
{
  "name": "print",
  "lineno": 9
}
{
  "name": "main",
  "lineno": 13
}
```

Finding Imports

To find the list of function calls in this code:

```
$ wensleydale test.py | jq '.. | select(.classname? == "Import") | [{name: .names[].
↪name, alias: .names[].asname}]'
[
  {
    "name": "this",
    "alias": "python"
  }
]
```

Contributors

Wensleydale started during the PyCon 2017 sprints in Portland. Many thanks to all those who contributed. It was surprising how quickly the tool came together.

- Matthew Boehm
- George Hickman
- Efron Licht
- Rishi Ramraj