robotpy-cppheaderparser Documentation

Release 5.1.0

Author

Dec 05, 2022

Contents

1 Documentation **3**

2 Install **5**

3 License **7**

4 Authors **9**

4.1 API 9

Python Module Index **15**

Index **17**

CppHeaderParser is a pure python C++ header parser that parses C++ headers and creates a data structure that you can use to do many types of things. We've found it particularly useful for creating programs that generate python wrappers around existing C++ programs.

robotpy-cppheaderparser is a fork of the CppHeaderParser library originally created by @senex. CppHeaderParser is an excellent library and critical to some of the stuff we do in the RobotPy project. Unfortunately, the maintainer seems to be busy, so robotpy-cppheaderparser was born.

We aim to maintain (some) compatibility with the existing code and make improvements and bugfixes as we need them – though some decisions made early on in this code's development means some compatibility may be broken as things get fixed.

If you find an bug, we encourage you to submit a pull request! New changes will only be accepted if there are tests to cover the change you made (and if they don't break existing tests).

Note: CppHeaderParser only does some very minimal interpretation of preprocessor directives – and we're looking at removing some of that from this library. If you need anything complex, you should preprocess the code yourself. You can use the excellent pure python preprocessor pcpp, or the preprocessing facilities provided by your favorite compiler.

CHAPTER 1

Documentation

Documentation can be found at https://cppheaderparser.readthedocs.io

CHAPTER 2

Install

```
pip install robotpy-cppheaderparser
```

CHAPTER 3

License

BSD License

CHAPTER 4

Authors

Originally developed by Jashua Cloutier, this fork is maintained by the RobotPy project.

Past contributors include:

- Jashua Cloutier
- Chris Love
- HartsAntler

4.1 API

To parse a header file and retrieve the resulting data structure, you'll want to use the *CppHeader* object:

```
import CppHeaderParser

header = CppHeaderParser.CppHeader("path/to/header.h")
```

Below is some documentation of the various object types that CppHeaderParser will generated after parsing a header file. The documentation is not yet currently comprehensive, so the best way to see what gets generated is by printing out the JSON representation:

```
python -m CppHeaderParser.tojson /path/to/header.h
```

Warning: CppHeaderParser is not safe to use from multiple threads

Note: CppHeaderParser only does some very minimal interpretation of preprocessor directives – and we're looking at removing some of that from this library. If you need anything complex, you should preprocess the code yourself. You can use the excellent pure python preprocessor pcpp, or the preprocessing facilities provided by your favorite compiler.

4.1.1 CppHeaderParser

class CppHeaderParser.CppHeaderParser.**CppBaseDecl**(*default_access*)

Bases: dict

Dictionary that contains the following

- access - Anything in supportedAccessSpecifier
- class - Name of the type, along with template specializations
- decl_name - Name of the type only
- decl_params - Only present if a template specialization (Foo<int>). Is a list of *CppTemplateParam*.
- decltype - True/False indicates a decltype, the contents are in decl_name
- virtual - True/False indicates virtual inheritance
- ... - True/False indicates a parameter pack

class CppHeaderParser.CppHeaderParser.**CppClass**(*nameStack*, *curTemplate*, *doxygen*, *location*, *defaultAccess*)

Bases: dict

Dictionary that contains at least the following keys:

- name - Name of the class
- doxygen - Doxygen comments associated with the class if they exist
- inherits - List of Classes that this one inherits. Values are *CppBaseDecl*
- methods - Dictionary where keys are from supportedAccessSpecifier and values are a lists of *CppMethod*
- namespace - Namespace of class
- properties - Dictionary where keys are from supportedAccessSpecifier and values are lists of *CppVariable*
- enums - Dictionary where keys are from supportedAccessSpecifier and values are lists of *CppEnum*
- nested_classes - Classes and structs defined within this class
- final - True if final
- abstract - True if abstract
- using - Using directives in this class scope: key is name for lookup, value is *CppVariable*
- parent - If not None, the class that this class is nested in

An example of how this could look is as follows:

```
{
    'name': ""
    'inherits':[]
    'methods':
    {
        'public':[],
        'protected':[],
        'private':[]
    },
    'properties':
    {
```

(continues on next page)

(continued from previous page)

```
        'public':[],
        'protected':[],
        'private':[]
    },
    'enums':
    {
        'public':[],
        'protected':[],
        'private':[]
    }
}
```

get_all_method_names()

get_all_methods()

get_all_pure_virtual_methods()

get_method_names(*type='public'*)

get_pure_virtual_methods(*type='public'*)

show()

Convert class to a string

class CppHeaderParser.CppHeaderParser.**CppEnum**(*name*, *doxygen*, *location*)

Bases: CppHeaderParser.CppHeaderParser._CppEnum

Contains the following keys:

- `name` - Name of the enum (ex. "ItemState")
- `namespace` - Namespace containing the enum
- `isclass` - True if created via 'enum class' or 'enum struct'
- `values` - List of values. The values are a dictionary with the following key/values:
  - `name` - name of the key (ex. "PARSING_HEADER"),
  - `value` - Specified value of the enum, this key will only exist if a value for a given enum value was defined

class CppHeaderParser.CppHeaderParser.**CppHeader**(*headerFileName*, *argType='file'*, *encoding=None*, *preprocessed=False*, ***kwargs*)

Bases: CppHeaderParser.CppHeaderParser._CppHeader

Parsed C++ class header

IGNORE_NAMES = ['__extension__']

classes = None

Dictionary of classes found in the header file. The key is the name of the class, value is *CppClass*

defines = None

List of #define directives found

defines_detail = None

List of #define directives found, with location information

enums = None

List of enums in this header as *CppEnum*

functions = None
List of free functions as *CppMethod*

headerFileNames = None
Filenames encountered in #line directives while parsing

includes = None
List of #include directives found

includes_detail = None
List of #include directives found with location information

nameSpaces = None
Namespaces in this header

pragmas = None
List of #pragma directives found as strings

pragmas_detail = None
List of pragmas with location information

show()

toJSON(*indent=4*, *separators=None*)
Converts a parsed structure to JSON

using = None
Using directives in this header outside of class scope: key is full name for lookup, value is *CppVariable*

variables = None
List of variables in this header as *CppVariable*

class CppHeaderParser.CppHeaderParser.**CppMethod**(*nameStack*, *curClass*, *methinfo*, *curTemplate*, *doxygen*, *location*)
Bases: CppHeaderParser.CppHeaderParser._CppMethod

Dictionary that contains at least the following keys:

- rtnType - Return type of the method (ex. "int")
- name - Name of the method
- doxygen - Doxygen comments associated with the method if they exist
- parameters - List of *CppVariable*
- parent - If not None, the class this method belongs to

show()

exception CppHeaderParser.CppHeaderParser.**CppParseError**(*msg*, *tok=None*)
Bases: Exception

class CppHeaderParser.CppHeaderParser.**CppTemplateParam**
Bases: dict

Dictionary that contains the following:

- decltype - If True, this is a decltype
- param - Parameter value or typename
- **params - Only present if a template specialization. Is a list of** *CppTemplateParam*
- ... - If True, indicates a parameter pack

class CppHeaderParser.CppHeaderParser.**CppUnion**(*nameStack*, *doxygen*, *location*)

Bases: *CppHeaderParser.CppHeaderParser.CppClass*

Dictionary that contains at least the following keys:

- `name` - Name of the union
- `doxygen` - Doxygen comments associated with the union if they exist
- `members` - List of members of the union

show()

Convert class to a string

transform_to_union_keys()

class CppHeaderParser.CppHeaderParser.**CppVariable**(*nameStack*, *doxygen*, *location*, *is_var=True*, ***kwargs*)

Bases: CppHeaderParser.CppHeaderParser._CppVariable

Dictionary that contains at least the following keys:

- `type` - Type for the variable (ex. "const string &")
- `name` - Name of the variable (ex. "numItems")
- `namespace` - Namespace
- `desc` - If a method/function parameter, doxygen description for this parameter (optional)
- `doxygen` - If a normal property/variable, doxygen description for this
- `default` - Default value of the variable, this key will only exist if there is a default value
- `extern` - True if its an extern, False if not
- `parent` - If not None, either the class this is a property of, or the method this variable is a parameter in
- `access` - Anything in supportedAccessSpecifier

Vars = []

class CppHeaderParser.CppHeaderParser.**TagStr**

Bases: `str`

Wrapper for a string that allows us to store the line number associated with it

CppHeaderParser.CppHeaderParser.**ignoreSymbols = ['Q_OBJECT']**

Symbols to ignore, usually special macros

Python Module Index

C

CppHeaderParser.CppHeaderParser, 10

Index

C

classes (*CppHeaderParser.CppHeaderParser.CppHeader attribute*), 11
CppBaseDecl (*class in CppHeaderParser.CppHeaderParser*), 10
CppClass (*class in CppHeaderParser.CppHeaderParser*), 10
CppEnum (*class in CppHeaderParser.CppHeaderParser*), 11
CppHeader (*class in CppHeaderParser.CppHeaderParser*), 11
CppHeaderParser.CppHeaderParser (*module*), 10
CppMethod (*class in CppHeaderParser.CppHeaderParser*), 12
CppParseError, 12
CppTemplateParam (*class in CppHeaderParser.CppHeaderParser*), 12
CppUnion (*class in CppHeaderParser.CppHeaderParser*), 12
CppVariable (*class in CppHeaderParser.CppHeaderParser*), 13

D

defines (*CppHeaderParser.CppHeaderParser.CppHeader attribute*), 11
defines_detail (*CppHeaderParser.CppHeaderParser.CppHeader attribute*), 11

E

enums (*CppHeaderParser.CppHeaderParser.CppHeader attribute*), 11

F

functions (*CppHeaderParser.CppHeaderParser.CppHeader attribute*), 11

G

get_all_method_names() (*CppHeaderParser.CppHeaderParser.CppClass method*), 11
get_all_methods() (*CppHeaderParser.CppHeaderParser.CppClass method*), 11
get_all_pure_virtual_methods() (*CppHeaderParser.CppHeaderParser.CppClass method*), 11
get_method_names() (*CppHeaderParser.CppHeaderParser.CppClass method*), 11
get_pure_virtual_methods() (*CppHeaderParser.CppHeaderParser.CppClass method*), 11

H

headerFileNames (*CppHeaderParser.CppHeaderParser.CppHeader attribute*), 12

I

IGNORE_NAMES (*CppHeaderParser.CppHeaderParser.CppHeader attribute*), 11
ignoreSymbols (*in module CppHeaderParser.CppHeaderParser*), 13
includes (*CppHeaderParser.CppHeaderParser.CppHeader attribute*), 12
includes_detail (*CppHeaderParser.CppHeaderParser.CppHeader attribute*), 12

N

nameSpaces (*CppHeaderParser.CppHeaderParser.CppHeader attribute*), 12

P

`pragmas` (*CppHeaderParser.CppHeaderParser.CppHeader attribute*), 12
`pragmas_detail` (*CppHeaderParser.CppHeaderParser.CppHeader attribute*), 12

S

`show()` (*CppHeaderParser.CppHeaderParser.CppClass method*), 11
`show()` (*CppHeaderParser.CppHeaderParser.CppHeader method*), 12
`show()` (*CppHeaderParser.CppHeaderParser.CppMethod method*), 12
`show()` (*CppHeaderParser.CppHeaderParser.CppUnion method*), 13

T

`TagStr` (*class in CppHeaderParser.CppHeaderParser*), 13
`toJSON()` (*CppHeaderParser.CppHeaderParser.CppHeader method*), 12
`transform_to_union_keys()` (*CppHeaderParser.CppHeaderParser.CppUnion method*), 13

U

`using` (*CppHeaderParser.CppHeaderParser.CppHeader attribute*), 12

V

`variables` (*CppHeaderParser.CppHeaderParser.CppHeader attribute*), 12
`Vars` (*CppHeaderParser.CppHeaderParser.CppVariable attribute*), 13