ctxvar Documentation

Release 1.0

Kay-Uwe (Kiwi) Lorenz

Sep 27, 2017

Contents

A module to access variables defined in calling frames.

Inspired by Perl6's `$*foo` variables, which are "dynamically overridable global variables" (Synopsis 2 on Twigils), I have created this module.

Basically this enables you to access local variables from calling frames:

```
>>> def a(): x = 1 ; return b()
>>> def b(): return ctx('x')
>>> a()
1
```

If you request to get a variable from calling contexts, there will be searched locals of each calling frame until requested variable is found, or there will be raised a *NameError* exception.

If you rely on this feature and you set a variable *x*, which should be accessed in a called function, you quickly hit the problem, that your *x* might be used in another called function locally and you do not get the expected value in your called function, but the one of the other *x*:

```
>>> def a(): x = 1 ; return b()
>>> def b(): x = 2 ; return c()
>>> def c(): # expects x from a
...     return ctx('x')
>>> a()
2
```

So it is good to brand context variables with some characters, which cannot be used in usual python syntax for variables:

```
>>> def a(): ctx('*x', 1) ; return b()
>>> def b(): x = 2 ; return c()
>>> def c(): # expects *x from a
...    return ctx('*x')
>>> a()
1
```

If you try to access a variable, which cannot be found, there will be raised a NameError:

```
>>> ctx('***does not exist***')
Traceback (most recent call last):
...
NameError: name '***does not exist***' not defined in calling frames
```

You can use *`ctxget()`* for specifying a default in such a case:

```
>>> ctxget('***does not exist***', 'default')
'default'
```

There are two convenience functions for normal import:

```
>>> import ctxvar
>>> ctxvar.set('*foo', 'bar')
>>> ctxvar.get('*foo', 'default')
'bar'
```

If you access a context var more than once within a function, it may be helpful to copy it to local frame, while getting the value:

```
>>> def a(): ctx('*x', 1) ; return b()
>>> def b(): x = 2 ; return c()
>>> def c(): # expects *x from a
...     first = ctx('*x', Here)
...     return ctx('*x') # this will access the local copy
>>> a()
1

>>> def a(): FOO = "Hello" ; return b()
>>> def b(): BAR = "World" ; return c()
>>> def c(): return ctxstr("${FOO} ${BAR}")
>>> a()
'Hello World'
```

This will go back the frames only once.

ctxvar.**ctx**(*name*, *value=Undef*, *caching=None*, *back=1*)

Set or get a context variable.

Parameters:

name name of variable

value Any value the variable shall be set to. Default is Undef, a local class, which indicates that rather a value shall be retrieved than stored.

You may also set the value to Here, this will still retrieve the value but will create a copy of the variable in current frame.

caching Set caching on, if there shall be created a copy in each frame between current one and the one holding the variable. Similar to *value = Here*.

back Number of frames initially go back. This usually needs not to be set and is rather for internal calls, see *set()*

Returns: value of variable

Raises: *NameError* - if top level frame is reached and variable could not be found.

ctxvar.**ctxget**(*name*, *default=None*)

get a context variable. If not exists return default value

ctxvar.**ctxstr**(*s*, *default=''*)

expand each ${...} s to its ctxvar. if default is None, replace by match

class ctxvar.**Here**

A Symbolic class for indicating, that you want to copy a context variable into local scope. See *ctx()*.

class ctxvar.**Undef**

A Symbolic class different from *None* for use as default value in a function, if *None* could be passed as value. See *ctx()*.

ctxvar.**set**(*name*, *value*)

set context variable name = value

ctxvar.**get**(*name*, *default=None*)

get a context variable. If not exists return default value

CHAPTER 1

Installation

From Sources

Get Mercurial repository from https://bitbucket.org/klorenz/ctxvar:

```
$ hg clone https://bitbucket.org/klorenz/ctxvar
```

And run python installation:

```
$ python setup.py install
```

CHAPTER 2

License

New BSD License

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of “Kay-Uwe (Kiwi) Lorenz” nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

C

Index