modutil Documentation

Release

Brett Cannon

Apr 23, 2018

Contents

A library for working with Python modules.

CHAPTER 1

Module Contents

STANDARD_MODULE_ATTRS
A container of standard attribute names on modules.

exception ModuleAttributeError(*importer_name*, *attribute*)
A subclass of `AttributeError` with the attributes *importer_name* and *attribute* set to the module being searched on and the attribute being searched for, respectively.

lazy_import(*module__name*, *to_import*)
Returns the importing module and a callable for lazy importing.

The module named by *module_name* represents the module performing the import to help facilitate resolving relative imports.

to_import is an iterable of the modules to be potentially imported. Modules may be specified by either absolute or relative names. The attribute name that the specified module is ultimately bound to is specified in one of two ways. First, the general case is the end of the dotted name of the module is what the attribute name will be, e.g. `pkg.mod` will be bound to `mod` on the importing module. Second, the `as` format of importing is also supported, e.g. `"pkg.mod as spam"` leads to `pkg.mod` bound to the attribute `spam` on the importing module.

This function returns a two-item sequence. The first is the importing module itself for easy referencing. The second item is a callable which is expected to be set to `__getattr__()` within the importing module to allow for lazy importing. For instance:

```
mod, __getattr__ = lazy_import(__name__, {'sys', '.submodule',
                                           'importlib.abc as imp_abc'})

def func():
    return mod.imp_abc.__name__
```

> **Warning:** This function should only be used in code where start-up time is paramount (e.g. large CLI apps). Otherwise using this function will lead to import errors occurring lazily in unexpected points and with a less helpful traceback.

filtered_attrs(*module*, *, *modules=False*, *private=False*, *dunder=False*, *common=False*)
Return a collection of attribute names found on the *module* object.

If *modules* is false then attributes pointing to modules are excluded. If *private* is false then attributes starting with, but not ending in, _ will be excluded. With *dunder* set to false then attributes starting and ending with _ are left out. The *common* argument controls whether attributes found in *STANDARD_MODULE_ATTRS* are included.

calc___all__(*module_name*, ***kwargs*)
Return a sorted list of defined attributes on *module_name*.

All values specified in `**kwargs` are directly passed to *filtered_attrs()*.

Since the calculation of attributes is done eagerly, the function should be called as late as possible if it's used as a side-effect for importing. For example, the suggested usage is:

```
# __all__ is defined at the end of the module.

# ... the entire module except for the last line of ...

__all__ = module.calc___all__(__name__)
```

filtered_dir(*module_name*, *, *additions={}*, ***kwargs*)
Return a callable which returns the attributes of *module_name*.

All values specified in `**kwargs` get passed directly to *filtered_attrs()*. The *additions* argument should be an iterable which is added to the final results.

Expected usage is:

```
__dir__ = modutil.filtered_dir(__name__)
```

chained__getattr__(*importer_name*, **getattrs*)
Return a callable which calls the chain of `__getattr__()` functions in sequence.

If *ModuleAttributeError* is raised and matches *importer_name* and the attribute being searched, then the exception will be caught and the search will continue. All other exceptions will be allowed to propagate immediately. If no callable successfully returns a value, *ModuleAttributeError* will be raised.

Example usage is:

```
mod, import_getattr = modutil.lazy_import(__name__, {'mod'})
some_other_getattr = ...
__getattr__ = modutil.chained___getattr__(__name__, import_getattr, all_getattr)
del import_getattr, some_other_getattr
```

Index