zope.deferredimport Documentation

Release 4.1

Zope Foundation and Contributors

Dec 10, 2021

Contents

Contents:

CHAPTER 1

Deferred Imports

Often, especially for package modules, you want to import names for convenience, but not actually perform the imports until necessary. The zope.deferredimport package provided facilities for defining names in modules that will be imported from somewhere else when used. You can also cause deprecation warnings to be issued when a variable is used, but we'll get to that later.

The `zope.deferredimport.define()` function can be used to define one or more names to be imported when they are accessed. Simply provide names as keyword arguments with import specifiers as values. The import specifiers are given as strings of the form "module:name", where module is the dotted name of the module and name is a, possibly dotted, name of an object within the module.

To see how this works, we'll create some sample modules within the zope.deferredimport package. We'll actually use a helper function specific to this document to define the modules inline so we can easily see what's in them. Let's start by defining a module, sample1, that defined some things to be imported:

```
>>> create_module(sample1 = '''
... print("Sampe 1 imported!")
...
... x = 1
...
...
... class C:
...     y = 2
...
...
... z = 3
... q = 4
... ''')
```

Note that the module starts by printing a message. This allows us to see when the module is actually imported. Now, let's define a module that imports some names from this module:

```
>>> create_module(sample2 = '''
... import zope.deferredimport
...
```

(continues on next page)

(continued from previous page)

```
... zope.deferredimport.define(
...     sample1='zope.deferredimport.sample1',
...     one='zope.deferredimport.sample1:x',
...     two='zope.deferredimport.sample1:C.y',
...     )
...
... three = 3
... x = 4
...
...
... def getx():
...     return x
... ''')
```

In this example, we defined the name 'sample1' as the module zope.deferredimport.sample1. The module isn't imported immediately, but will be imported when needed. Similarly, the name 'one' is defined as the 'x' attribute of sample1.

The sample1 module prints a message when it is imported. When we import sample2, we don't see a message until we access a variable:

```
>>> import zope.deferredimport.sample2
>>> print(zope.deferredimport.sample2.one)
Sampe 1 imported!
1

>>> import zope.deferredimport.sample1

>>> zope.deferredimport.sample2.sample1 is zope.deferredimport.sample1
True
```

Note that a deferred attribute appears in a module's dictionary *after* it is accessed the first time:

```
>>> 'two' in zope.deferredimport.sample2.__dict__
False

>>> zope.deferredimport.sample2.two
2

>>> 'two' in zope.deferredimport.sample2.__dict__
True
```

When deferred imports are used, the original module is replaced with a proxy.

```
>>> type(zope.deferredimport.sample2)
<class 'zope.deferredimport.deferredmodule.ModuleProxy'>
```

But we can use the proxy just like the original. We can even update it.

```
>>> zope.deferredimport.sample2.x=5
>>> zope.deferredimport.sample2.getx()
5
```

And the inspect module thinks it's a module:

```
>>> import inspect
>>> inspect.ismodule(zope.deferredimport.sample2)
True
```

In the example above, the modules were fairly simple. Let's look at a more complicated example.

```
# >>> create_module(sample3 = '''
# ...
# ... import zope.deferredimport
# ... import zope.deferredimport.sample4
# ...
# ... zope.deferredimport.define(
# ...     sample1 = 'zope.deferredimport.sample1',
# ...     one = 'zope.deferredimport.sample1:x',
# ...     two = 'zope.deferredimport.sample1:C.y',
# ...     )
# ...
# ... x = 1
# ...
# ... ''')

# >>> create_module(sample4 = '''
# ... import sample3
# ...
# ... def getone():
# ...     return sample3.one
# ...
# ... ''')
```

Here, we have a circular import between sample3 and sample4. When sample3 is imported, it imports sample 4, which then imports sample3. Let's see what happens when we use these modules in an unfortunate order:

```
# XXX: Relative imports like this are not possible on Python 3 anymore.
# PY2
#
#    >>> import zope.deferredimport.sample3
#    >>> import zope.deferredimport.sample4
#
#    >>> zope.deferredimport.sample4.getone()
#    Traceback (most recent call last):
#    ...
#    AttributeError: 'module' object has no attribute 'one'
#
#Hm.  Let's try accessing one through sample3:
#
#    >>> zope.deferredimport.sample3.one
#    1
#
#Funny, let's try getone again:
#
#    >>> zope.deferredimport.sample4.getone()
#    1
```

The problem is that sample4 obtained sample3 before sample4 was replaced by a proxy. This example is slightly pathological because it requires a circular import and a relative import, but the bug introduced is very subtle. To guard against this, you should define deferred imports before importing any other modules. Alternatively, you can call the initialize function before importing any other modules, as in:

```
>>> create_module(sample5 = '''
... import zope.deferredimport
... zope.deferredimport.initialize()
...
... import zope.deferredimport.sample6  # noqa: E402 import not at top
...
... zope.deferredimport.define(
...     sample1='zope.deferredimport.sample1',
...     one='zope.deferredimport.sample1:x',
...     two='zope.deferredimport.sample1:C.y',
...     )
...
... x = 1
... ''')

>>> create_module(sample6 = '''
... import zope.deferredimport.sample5
...
...
... def getone():
...     return zope.deferredimport.sample5.one
... ''')

>>> import zope.deferredimport.sample5
>>> import zope.deferredimport.sample6

>>> zope.deferredimport.sample6.getone()
1
```

1.1 Deprecation

Deferred attributes can also be marked as deprecated, in which case, a message will be printed the first time they are accessed.

Lets define a module that has deprecated attributes defined as deferred imports:

```
>>> create_module(sample7 = '''
... import zope.deferredimport
... zope.deferredimport.initialize()
...
... zope.deferredimport.deprecated(
...     "Import from sample1 instead",
...     x='zope.deferredimport.sample1:x',
...     y='zope.deferredimport.sample1:C.y',
...     z='zope.deferredimport.sample1:z',
...     )
... ''')
```

Now, if we use one of these variables, we'll get a deprecation warning:

```
>>> import zope.deferredimport.sample7
>>> zope.deferredimport.sample7.x
docs/narrative.rst:1: DeprecationWarning:
            x is deprecated. Import from sample1 instead
```

(continues on next page)

(continued from previous page)

```
  ==================
1
```

but only the first time:

```
>>> zope.deferredimport.sample7.x
1
```

1.2 Importing multiple names from the same module

Sometimes, you want to get multiple things from the same module. You can use `defineFrom()` or `deprecatedFrom()` to do that:

```
>>> create_module(sample8 = '''
... import zope.deferredimport
...
... zope.deferredimport.deprecatedFrom(
...     "Import from sample1 instead",
...     'zope.deferredimport.sample1',
...     'x', 'z', 'q',
... )
...
... zope.deferredimport.defineFrom(
...     'zope.deferredimport.sample9',
...     'a', 'b', 'c',
... )
... ''')

>>> create_module(sample9 = '''
... print('Imported sample 9')
... a, b, c = range(10, 13)
... ''')

>>> import zope.deferredimport.sample8
>>> zope.deferredimport.sample8.q
docs/narrative.rst:1: DeprecationWarning:
        q is deprecated. Import from sample1 instead
  ==================
4

>>> zope.deferredimport.sample8.c
Imported sample 9
12
```

Note, as in the example above, that you can make multiple deferred-import calls in a module.

CHAPTER 2

`zope.deferredimport` API

CHAPTER 3

Indices and tables

- genindex
- modindex
- search