

## 8.18 利用Mixins扩展类功能¶

### 问题¶

你有很多有用的方法，想使用它们来扩展其他类的功能。但是这些类并没有任何继承的关系。因此你不能简单的将这些方法放入一个基类，然后被其他类继承。

### 解决方案¶

通常当你想自定义类的时候会碰上这些问题。可能是某个库提供了一些基础类， 你可以利用它们来构造你自己的类。

假设你想扩展映射对象，给它们添加日志、唯一性设置、类型检查等等功能。下面是一些混入类：

```
class LoggedMappingMixin:
    """
    Add logging to get/set/delete operations for debugging.
    """
    __slots__ = () # 混入类都没有实例变量，因为直接实例化混入类没有任何意义

    def __getitem__(self, key):
        print('Getting ' + str(key))
        return super().__getitem__(key)

    def __setitem__(self, key, value):
        print('Setting {} = {!r}'.format(key, value))
        return super().__setitem__(key, value)

    def __delitem__(self, key):
        print('Deleting ' + str(key))
        return super().__delitem__(key)

class SetOnceMappingMixin:
    """
    Only allow a key to be set once.
    """
    __slots__ = ()

    def __setitem__(self, key, value):
        if key in self:
            raise KeyError(str(key) + ' already set')
        return super().__setitem__(key, value)

class StringKeysMappingMixin:
    """
    Restrict keys to strings only
    """
    __slots__ = ()

    def __setitem__(self, key, value):
        if not isinstance(key, str):
            raise TypeError('keys must be strings')
        return super().__setitem__(key, value)
```

这些类单独使用起来没有任何意义，事实上如果你去实例化任何一个类，除了产生异常外没什么作用。它们是用来通过多继承来和其他映射对象混入使用的。例如：

```
class LoggedDict(LoggedMappingMixin, dict):
    pass

d = LoggedDict()
d['x'] = 23
print(d['x'])
del d['x']
```

```

from collections import defaultdict

class SetOnceDefaultDict(SetOnceMappingMixin, defaultdict):
    pass

d = SetOnceDefaultDict(list)
d['x'].append(2)
d['x'].append(3)
# d['x'] = 23 # KeyError: 'x already set'

```

这个例子中，可以看到混入类跟其他已存在的类(比如dict、defaultdict和OrderedDict)结合起来使用，一个接一个。结合后就能发挥正常功效了。

## 讨论

混入类在标准库中很多地方都出现过，通常都是用来像上面那样扩展某些类的功能。它们也是多继承的一个主要用途。比如，当你编写网络代码时候，你会经常使用 socketserver 模块中的 ThreadingMixIn 来给其他网络相关类增加多线程支持。例如，下面是一个多线程的XML-RPC服务：

```

from xmlrpc.server import SimpleXMLRPCServer
from socketserver import ThreadingMixIn
class ThreadedXMLRPCServer(ThreadingMixIn, SimpleXMLRPCServer):
    pass

```

同时在一些大型库和框架中也会发现混入类的使用，用途同样是增强已存在的类的功能和一些可选特征。

对于混入类，有几点需要记住。首先是，混入类不能被实例化使用。其次，混入类没有自己的状态信息，也就是说它们并没有定义 \_\_init\_\_() 方法，并且没有实例属性。这也是为什么我们在上面明确定义了 \_\_slots\_\_ = ()。

还有一种实现混入类的方式就是使用类装饰器，如下所示：

```

def LoggedMapping(cls):
    """第二种方式：使用类装饰器"""
    cls.__getitem__ = cls.__getitem__
    cls.__setitem__ = cls.__setitem__
    cls.__delitem__ = cls.__delitem__

    def __getitem__(self, key):
        print('Getting ' + str(key))
        return cls.__getitem__(self, key)

    def __setitem__(self, key, value):
        print('Setting {} = {}'.format(key, value))
        return cls.__setitem__(self, key, value)

    def __delitem__(self, key):
        print('Deleting ' + str(key))
        return cls.__delitem__(self, key)

    cls.__getitem__ = __getitem__
    cls.__setitem__ = __setitem__
    cls.__delitem__ = __delitem__
    return cls

@LoggedMapping
class LoggedDict(dict):
    pass

```

这个效果跟之前的是一样的，而且不再需要使用多继承了。参考9.12小节获取更多类装饰器的信息，参考8.13小节查看更多混入类和类装饰器的例子。