



# Werkzeug(Flask)之Local、LocalStack和LocalProxy



在我们使用Flask以及Werkzeug框架的过程中，经常会遇到如下三个概念：Local、LocalStack和LocalProxy。尤其在学习Flask的Request Context和App Context的过程中，这几个概念出现的更加频繁，另外很多Flask插件都会使用这三个概念对应的技术。那么这三个东西到底是什么？我们为什么需要它们？以及如何使用呢？本篇文章主要就是来解答这些问题。

## Local

这部分我们重点介绍Local概念，主要分为以下几个部分：

- 为什么需要Local?
- Local的使用
- Local的实现

### 为什么需要Local?

在Python的标准库中提供了 `thread local` 对象用于存储thread-safe和thread-specific的数据，通过这种方式存储的数据只在线程中有效，而对于其它线程则不可见。正是基于这样的特性，我们可以把针对线程全局的数据存储进 `thread local` 对象，举个简单的例子

```
1 >>from threading import local
2 >>thread_local_data = local()
3 >>thread_local_data.user_name="Jim"
4 >>thread_local_data.user_name
5 'Jim'
```

使用 `thread local` 对象虽然可以基于线程存储全局变量，但是在Web应用中可能会存在如下问题：

1. 有些应用使用的是greenlet协程，这种情况下无法保证协程之间数据的隔离，因为不同的协程可以在同一个线程当中。
2. 即使使用的是线程，WSGI应用也无法保证每个http请求使用的都是不同的线程，因为后一个http请求可能使用的是之前的http请求的线程，这样的话存储于 `thread local` 中的数据可能是之前残留的数据。

为了解决上述问题，Werkzeug开发了自己的local对象，这也是为什么我们需要Werkzeug的local对象

### Local的使用

先举一个简单的示例：

```
1 from werkzeug.local import Local, LocalManager
2
3 local = Local()
4 local_manager = LocalManager([local])
```

#### 推荐阅读

swagger2-3.0.0[Unable to scan documentation context default]问...  
阅读 566

Vue 3 响应式数据本质  
阅读 273

关于在springboot中获取request这件事  
阅读 177

Redux源码阅读（一）——  
createStore、dispatch、subscribe  
阅读 379

JavaScript 手写 new 的实现  
阅读 310

prometheus\_client源码分析（一）



logo购买

# Werkzeug(Flask)之Local、LocalStack和LocalProxy

geekpy

关注

赞赏支持

```
11 | application = local_manager.make_middleware(application)
```

- 首先Local对象需要通过LocalManager来管理，初次生成LocalManager对象需要传一个list类型的参数，list中是Local对象，当有新的Local对象时，可以通过 `local_manager.locals.append()` 来添加。而当LocalManager对象清理的时候会将所有存储于locals中的当前context的数据都清理掉
- 上例中当local.request被赋值之后，其可以在当前context中作为全局数据使用
- 所谓当前context(the same context)意味着是在同一个greenlet(如果有)中，也就肯定是在同一个线程当中

那么Werkzeug的Local对象是如何实现这种在相同的context环境下保证数据的全局性和隔离性的呢？

## Local的实现

我们先来看下源代码

```
1  # 在有greenlet的情况下，get_indent实际获取的是greenlet的id，而没有greenlet的情况下获取的是thread id
2  try:
3      from greenlet import getcurrent as get_indent
4  except ImportError:
5      try:
6          from thread import get_ident
7      except ImportError:
8          from _thread import get_ident
9
10 class Local(object):
11     __slots__ = ('__storage__', '__ident_func__')
12
13     def __init__(self):
14         object.__setattr__(self, '__storage__', {})
15         object.__setattr__(self, '__ident_func__', get_indent)
16
17     def __iter__(self):
18         return iter(self.__storage__.items())
19
20     # 当调用Local对象时，返回对应的LocalProxy
21     def __call__(self, proxy):
22         """Create a proxy for a name."""
23         return LocalProxy(self, proxy)
24
25     # Local类中特有的method，用于清空greenlet id或线程id对应的dict数据
26     def __release_local__(self):
27         self.__storage__.pop(self.__ident_func__(), None)
28
29     def __getattr__(self, name):
30         try:
31             return self.__storage__[self.__ident_func__()][name]
32         except KeyError:
33             raise AttributeError(name)
34
35     def __setattr__(self, name, value):
36         ident = self.__ident_func__()
37         storage = self.__storage__
38         try:
39             storage[ident][name] = value
40         except KeyError:
41             storage[ident] = {name: value}
42
43     def __delattr__(self, name):
44         try:
45             del self.__storage__[self.__ident_func__()][name]
46         except KeyError:
47             raise AttributeError(name)
```

### 推荐阅读

swagger2-3.0.0[Unable to scan documentation context default]问...  
阅读 566

Vue 3 响应式数据本质  
阅读 273

关于在springboot中获取request这件事  
阅读 177

Redux源码阅读（一）——  
createStore、dispatch、subscribe  
阅读 379

JavaScript 手写 new 的实现  
阅读 310



logo购买



写下你的评论...

评论13 赞55

## Werkzeug(Flask)之Local、LocalStack和LocalProxy

- 通过重新实现 `__getattr__` , `__setattr__` 等魔术方法, 我们在greenlet或者线程中使用local对象时, 实际会自动获取greenlet id(或者线程id), 从而获取到对应的dict存储空间, 再通过name key就可以获取到真正的存储的对象。这个技巧实际上在编写线程安全或协程安全的代码时是非常有用的, 即通过线程id(或协程id)来分别存储数据。
- 当我们需要释放local数据的内存时, 可以通过调用`release_local()`函数来释放当前context的local数据, 如下

```
1 >>> loc = Local()
2 >>> loc.foo = 42
3 >>> release_local(loc) # release_local实际调用local对象的__release_local__ method
4 >>> hasattr(loc, 'foo')
5 False
```

### LocalStack

LocalStack与Local对象类似, 都是可以基于Greenlet协程或者线程进行全局存储的存储空间(实际LocalStack是对Local进行了二次封装), 区别在于其数据结构是栈的形式。示例如下:

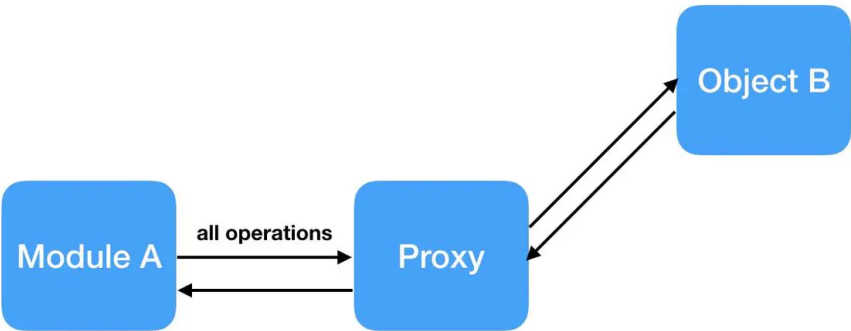
```
1 >>> ls = LocalStack()
2 >>> ls.push(42)
3 >>> ls.top
4 42
5 >>> ls.push(23)
6 >>> ls.top
7 23
8 >>> ls.pop()
9 23
10 >>> ls.top
11 42
```

- 从示例看出Local对象存储的时候是类似字典的方式, 需要有key和value, 而LocalStack是基于栈的, 通过push和pop来存储和弹出数据
- 另外, 当我们想释放存储空间的时候, 也可以调用`release_local()`

LocalStack在Flask框架中会频繁的出现, 其Request Context和App Context的实现都是基于LocalStack, 具体可以参考Github上的[Flask源码](#)

### LocalProxy

LocalProxy用于代理Local对象和LocalStack对象, 而所谓代理就是作为中间的代理人来处理所有针对被代理对象的操作, 如下图所示:



geekpy

关注

赞赏支持

- 推荐阅读
- swagger2-3.0.0[Unable to scan documentation context default]问...  
阅读 566
- Vue 3 响应式数据本质  
阅读 273
- 关于在springboot中获取request这件事  
阅读 177
- Redux源码阅读 (一) —— createStore、dispatch、subscribe  
阅读 379
- JavaScript 手写 new 的实现  
阅读 310





# Werkzeug(Flask)之Local、LocalStack和LocalProxy

接下来我们将重点讲下如下内容：

- LocalProxy的使用
- LocalProxy代码解析
- 为什么要使用LocalProxy

## LocalProxy的使用

初始化LocalProxy有三种方式：

### 1. 通过Local或者LocalStack对象的 `__call__` method

```
1 from werkzeug.local import Local
2 l = Local()
3
4 # these are proxies
5 request = l('request')
6 user = l('user')
7
8
9 from werkzeug.local import LocalStack
10 _response_local = LocalStack()
11
12 # this is a proxy
13 response = _response_local()
```

上述代码直接将对象像函数一样调用，这是因为Local和LocalStack都实现了 `__call__` method，这样其对象就是callable的，因此当我们将对象作为函数调用时，实际调用的是 `__call__` method，可以看看本文开头部分的Local的源代码，会发现 `__call__` method会返回一个LocalProxy对象

### 2. 通过LocalProxy类进行初始化

```
1 l = Local()
2 request = LocalProxy(l, 'request')
```

实际上这段代码跟第一种方式是等价的，但这种方式是最'原始'的方式，我们在Local的源代码实现中看到其 `__call__` method就是通过这种方式生成LocalProxy的

### 3. 使用callable对象作为参数

```
1 request = LocalProxy(get_current_request())
```

通过传递一个函数，我们可以自定义如何返回Local或LocalStack对象

那么LocalProxy是如何实现这种代理的呢？接下来看下源码解析

## LocalProxy代码解析

下面截取LocalProxy的部分代码，我们来进行解析

```
1 # LocalProxy部分代码
2
3 @implements_bool
4 class LocalProxy(object):
5     slots = ('_local', '_dict', '_name', '_wrapped')
```

## 推荐阅读

swagger2-3.0.0[Unable to scan documentation context default]问...  
阅读 566

Vue 3 响应式数据本质  
阅读 273

关于在springboot中获取request这件事  
阅读 177

Redux源码阅读（一）——  
createStore、dispatch、subscribe  
阅读 379

JavaScript 手写 new 的实现  
阅读 310



logo购买



# Werkzeug(Flask)之Local、LocalStack和LocalProxy

```
11         # "local" is a callable that is not an instance of Local or
12         # LocalManager: mark it as a wrapped function.
13         object.__setattr__(self, '__wrapped__', local)
14
15     def _get_current_object(self):
16         """Return the current object. This is useful if you want the real
17         object behind the proxy at a time for performance reasons or because
18         you want to pass the object into a different context.
19         """
20         # 由于所有Local或LocalStack对象都有__release_local__ method, 所以如果没有该属性就表明self.
21         if not hasattr(self.__local, '__release_local__'):
22             return self.__local()
23         try:
24             # 此处self.__local为Local或LocalStack对象
25             return getattr(self.__local, self.__name__)
26         except AttributeError:
27             raise RuntimeError('no object bound to %s' % self.__name__)
28
29     @property
30     def __dict__(self):
31         try:
32             return self._get_current_object().__dict__
33         except RuntimeError:
34             raise AttributeError('__dict__')
35
36     def __getattr__(self, name):
37         if name == '__members__':
38             return dir(self._get_current_object())
39         return getattr(self._get_current_object(), name)
40
41     def __setitem__(self, key, value):
42         self._get_current_object()[key] = value
43
44     def __delitem__(self, key):
45         del self._get_current_object()[key]
46
47     if PY2:
48         __getslice__ = lambda x, i, j: x._get_current_object()[i:j]
49
50         def __setslice__(self, i, j, seq):
51             self._get_current_object()[i:j] = seq
52
53         def __delslice__(self, i, j):
54             del self._get_current_object()[i:j]
55
56     # 截取部分操作符代码
57     __setattr__ = lambda x, n, v: setattr(x._get_current_object(), n, v)
58     __delattr__ = lambda x, n: delattr(x._get_current_object(), n)
59     __str__ = lambda x: str(x._get_current_object())
60     __lt__ = lambda x, o: x._get_current_object() < o
61     __le__ = lambda x, o: x._get_current_object() <= o
62     __eq__ = lambda x, o: x._get_current_object() == o
```

## 推荐阅读

swagger2-3.0.0[Unable to scan documentation context default]问...  
阅读 566

Vue 3 响应式数据本质  
阅读 273

关于在springboot中获取request这件事  
阅读 177

Redux源码阅读（一）——  
createStore、dispatch、subscribe  
阅读 379

JavaScript 手写 new 的实现  
阅读 310



logo购买

- 首先在 `__init__` method中传递的 `local` 参数会被赋予属性 `_LocalProxy__local`,该属性可以通过 `self.__local` 进行访问, 关于这一点可以看[StackOverflow的问题回答](#)
- LocalProxy通过 `_get_current_object` 来获取代理的对象。需要注意的是当初始化参数为callable对象时, 则直接调用以返回Local或LocalStack对象, 具体看源代码的注释。
- 重载了绝大多数操作符, 以便在调用LocalProxy的相应操作时, 通过 `_get_current_object` method来获取真正代理的对象, 然后再进行相应操作

## 为什么要使用LocalProxy

可是说了这么多, 为什么一定要用proxy, 而不能直接调用Local或LocalStack对象呢? 这主要是在有多个可供调用的对象的时候会出现问题, 如下图:

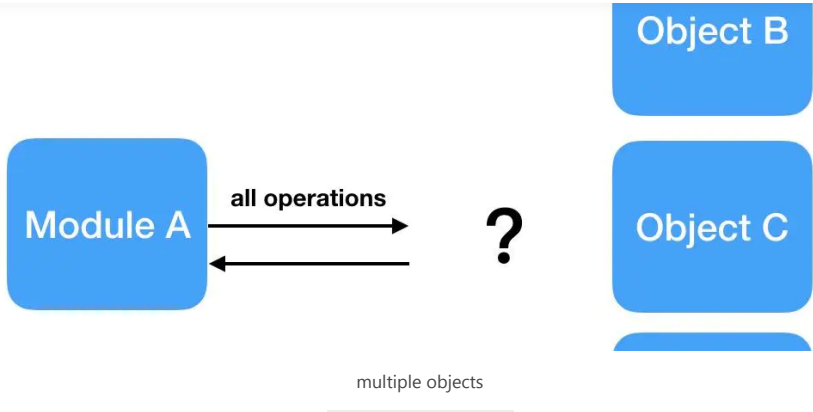


# Werkzeug(Flask)之Local、LocalStack和LocalProxy

geekpy

关注

赞赏支持



## 推荐阅读

swagger2-3.0.0[Unable to scan documentation context default]问...  
阅读 566

Vue 3 响应式数据本质  
阅读 273

关于在springboot中获取request这件事  
阅读 177

Redux源码阅读（一）——  
createStore、dispatch、subscribe  
阅读 379

JavaScript 手写 new 的实现  
阅读 310

我们再通过下面的代码也许可以看出来一二：

```
1 # use Local object directly
2 from werkzeug.local import LocalStack
3 user_stack = LocalStack()
4 user_stack.push({'name': 'Bob'})
5 user_stack.push({'name': 'John'})
6
7 def get_user():
8     # do something to get User object and return it
9     return user_stack.pop()
10
11
12 # 直接调用函数获取user对象
13 user = get_user()
14 print user['name']
15 print user['name']
```

打印结果是：

```
1 John
2 John
```

再看下使用LocalProxy

```
1 # use LocalProxy
2 from werkzeug.local import LocalStack, LocalProxy
3 user_stack = LocalStack()
4 user_stack.push({'name': 'Bob'})
5 user_stack.push({'name': 'John'})
6
7 def get_user():
8     # do something to get User object and return it
9     return user_stack.pop()
10
11 # 通过LocalProxy使用user对象
12 user = LocalProxy(get_user)
13 print user['name']
14 print user['name']
```

打印结果是：

```
1 John
2 Bob
```

怎么样，看出区别了吧，直接使用LocalStack对象，user一旦赋值就无法再动态更新了，而使用LocalProxy，每次调用操作该变量，该操作作用于该属性，都会重新获取user，从而实现了动态更新。



写下你的评论...

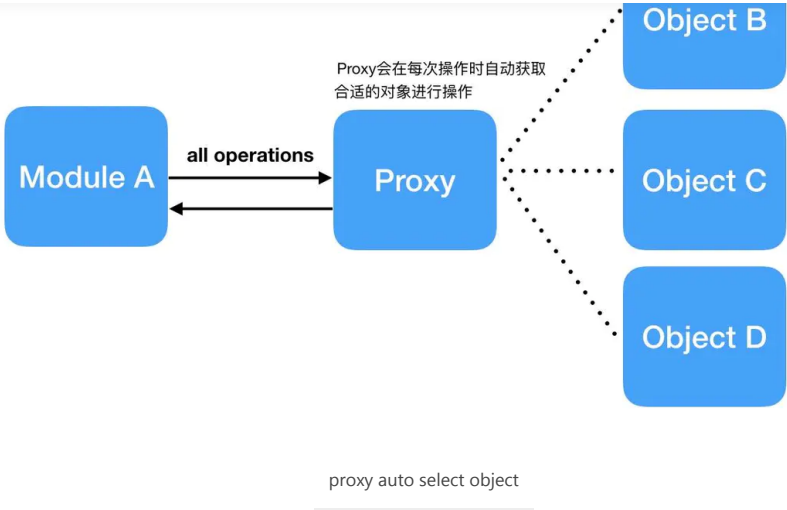
评论13 赞55

# Werkzeug(Flask)之Local、LocalStack和LocalProxy

geekpy

关注

赞赏支持



Flask以及Flask的插件很多时候都需要这种动态更新的效果，因此LocalProxy就会非常有用了。

至此，我们针对Local、LocalStack和LocalProxy的概念已经做了详细阐释，如果你觉得文章对你有帮助，不妨点个赞吧！

👍 55人点赞 >

👎

📄 Flask

...


更多精彩内容，就在简书APP



"小礼物走一走，来简书关注我"

赞赏支持

还没有人赞赏，支持一下

geekpy 一念嗔心起，百万障门开

总资产26 (约1.52元) 共写了10.8W字 获得730个赞 共422个粉丝

关注



logo购买

## 现在网络游戏排名前十的都有哪些游戏？



写下你的评论...






# Werkzeug(Flask)之Local、LocalStack和LocalProxy

geekpy

关注

赞赏支持

ADHDer\_830f

10楼 2019.08.08 16:58

Local 的类中 有slots\_ = ("\_storage\_", "\_ident\_func\_"), loc作为Local的实例，为何你例子中写loc.foo没有报错啊？

👍 赞

💬 回复

geekpy 作者

2019.08.23 19:53

这是因为Local类重写了\_\_getattr\_\_

💬 回复

✍ 添加新评论

dff873928282

9楼 2019.04.08 23:01


LocalProxy最后的举例很好！

👍 赞

💬 回复

## 怎么设计自己的签名



octocat

8楼 2018.10.31 09:47


首先在\_\_init\_\_ method中传递的local参数会被赋予属性\_LocalProxy\_\_local,该属性可以通过self.local进行访问

-----

这里应该是 self.\_\_local

👍 赞


💬 回复

ShukeZheng

2019.03.12 15:51

为什么参数名叫\_LocalProxy\_\_local 却可以被 self.\_\_local访问。这块没懂


💬 回复

铜锣湾洪爷

2019.03.21 15:31

@风在动云在飞 因为在类里面，双下划线标注的变量，会被自动转换为\_\_yourclass\_\_yourvariables的形式，你可以试下在某个类中定义一个双下划线变量，然后实例化出来输出\_\_dict\_\_查看

💬 回复

ShukeZheng

2019.03.25 15:21

@HxsLsh 这样啊，一直没看到过这个知识点，谢了

💬 回复

✍ 添加新评论 | 还有1条评论，[查看更多](#)

## 推荐阅读

swagger2-3.0.0[Unable to scan documentation context default]问...  
阅读 566

Vue 3 响应式数据本质  
阅读 273

关于在springboot中获取request这件事  
阅读 177

Redux源码阅读（一）—— createStore、dispatch、subscribe  
阅读 379

JavaScript 手写 new 的实现  
阅读 310



logo购买

IN4



Werkzeug(Flask)之Local、LocalStack和LocalProxy



geekpy

关注

赞赏支持

就是 user\_stack.pop()

1 回复



庞贝船长

6楼 2018.08.13 14:52

看Flask源码过来的，这篇帮助很大！

赞 回复



f8bd3d28c8fa

5楼 2018.06.23 06:32

写得挺好得，留名慢慢看。

赞 回复



NeXTCDO

4楼 2018.05.30 15:54

多谢楼主，LocalProxy确实挺神奇的

赞 回复



versus2017

3楼 2018.02.27 16:55

很棒，我看懂了

赞 回复

推荐阅读

swagger2-3.0.0[Unable to scan documentation context default]问...  
阅读 566

Vue 3 响应式数据本质  
阅读 273

关于在springboot中获取request这件事  
阅读 177

Redux源码阅读（一）——  
createStore、dispatch、subscribe  
阅读 379

JavaScript 手写 new 的实现  
阅读 310



logo购买

被以下专题收入，发现更多相似内容



Python/...



flask



AutoTes...



flask, p...



Python 运维

推荐阅读

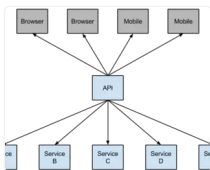
Spring Cloud

Spring Cloud为开发人员提供了快速构建分布式系统中一些常见模式的工具（例如配置管理，服务发现，断路器，智...



卡卡罗2017 阅读 117,588 评论 15 赞 132

更多精彩内容>

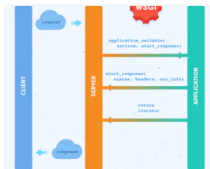


用尽洪荒之力学习Flask源码

[TOC]一直想做源码阅读这件事，总感觉难度太高时间太少，可望不可见。最近正好时间充裕，决定试试做一下，并记录一下...



何柯君 阅读 5,348 评论 3 赞 94



三亚必去的八大景点,你都去过了吗



写下你的评论...

评论13

赞55

Werkzeug(Flask)之Local、LocalStack和LocalProxy

geekpy

关注

赞赏支持

19年6月更新：这篇翻译已经被移到 TangYefei's Blog，相较于简书阅读体验会更好，如果喜欢请点个 ♥ ...

 tangyefei 阅读 33,132 评论 22 赞 257



iApp\_《ilua》速成开发手册3.0

《ilua》速成开发手册3.0 官方用户交流：iApp开发交流（1） 239547050iApp开发交流（2） 1...

 叶染柒丶 阅读 6,089 评论 0 赞 10

Android Sqlite字段重复则更新，不重复则添加

有时候会有这样的需求：更新数据库中的某一列，如果该表中有某个字段，就更新该列，如果没有则添加到表中。1. 先要在建...

 张老梦 阅读 4,642 评论 0 赞 2

推荐阅读

swagger2-3.0.0[Unable to scan documentation context default]问...  
阅读 566

Vue 3 响应式数据本质  
阅读 273

关于在springboot中获取request这件事  
阅读 177

Redux源码阅读（一）—— createStore、dispatch、subscribe  
阅读 379

JavaScript 手写 new 的实现  
阅读 310



logo购买