**python黑魔法---装饰器（decorator）**

<http://python.jobbole.com/87309/>

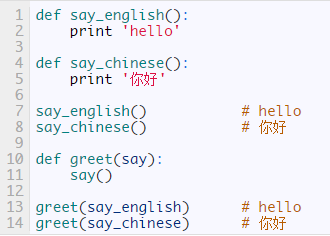
python 是一门优雅的语言，有些使用方法就像魔法一样。装饰器(decorator)就是一种化腐朽性为神奇的技巧。最近一直都在使用 Tornado 框架，一直还是念念不忘 Flask 。Flask 是我最喜欢的 Python 框架，最早被它吸引也是源自它使用装饰器这个语法糖（Syntactic sugar）来做 Router，让代码看上去就感觉甜甜的。

Tornado 中的 Router 略显平淡，怀念 Flask 的味道，于是很好奇的想知道 Flask 是如何使用这个魔法。通过阅读 Flask 的源码，我们也可以为 Tornado 实现了一个装饰器 Router。

当然对于刚接触 Python 的人，也许很容易理解装饰器本质是设计模式中的装饰器模式。可是 Python 通过@一个实现装饰器的语法糖。本文的目的就是让 @ 不再神秘。

**一切都是对象**

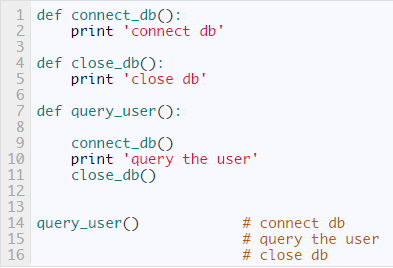
Python 里一切都是对象，当然这不代表一切都是女朋友。函数也是对象，因而可以当成参数传递，例如：



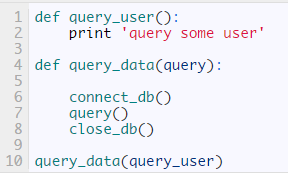
我们的 greet 函数的参数，也是一个函数对象。可以传递这个参数对象。我们调用greet的时候，greet 内部进行函数参数的调用。

**装饰模式**

装饰模式，顾名思义，就是在调用目标函数之前，对这个函数对象进行装饰。比如一个对数据库操作的方法，我们在查询数据之前，需要连接一下数据库，当查询结束之后，需要再把连接断开关闭。正常的逻辑如下：



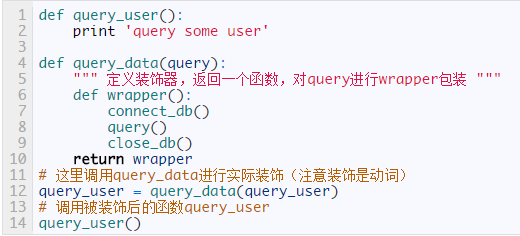
我们把 连接数据库(connect\_db) 和 关闭连接 (close\_db)都封装成了函数。 query\_data 方法执行我们查询的具体逻辑。这样需要不同的查询方法，只需要把查询的逻辑也封装成一个方法就Ok啦。



把查询的函数对象传进来，符合开篇说的一切都是对象。装饰器完成啦。对，就这么简单，**query\_data 就是对 query\_user 的装饰**，当然你还可以写出 query\_blog 等方法。

等等，设想一种情况，在我们使用装饰函数之前，项目的代码已经有了大量的 query\_user方法的调用。如果使用了query\_data 包装。我们就不得不把之前 query\_user() 的地方统统替换成 query\_data(query\_user)。怎么样才能减少对代码的改动呢？

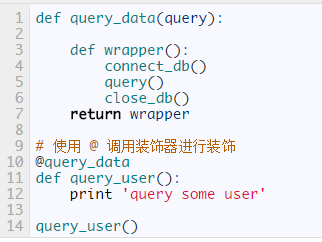
我们的出发点是为了保持之前的 query\_user() 不改动，现在实际情况是调用 query\_data(query\_user)。如果 query\_data 调用的时候，返回一个函数呢？例如下面的代码：



这样一个完整的装饰器就完成了，比起前面的版本，我们不需要改动之前写好的 query\_user 代码。一个关键点在于query\_data 调用的时候，返回了一个 wrapper 函数，而这个wrapper 函数执行 query 函数调用前后的一些逻辑。另外一个关键就是调用装饰器 query\_data 装饰函数。

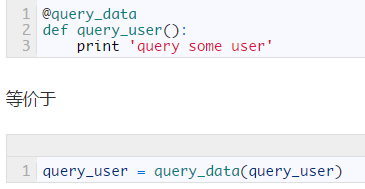
**语法糖@**

前面的代码，可以使用 python的装饰器语法糖@，如下：



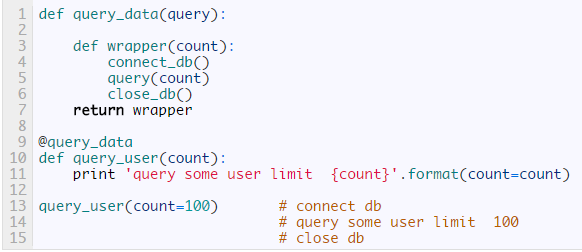
前面的 装饰器 调用进行装饰的时候，python 有一个语法糖。

如果给装饰器函数前面加一个@，我们可以理解为调用了一些装饰器函数，即 @query\_data 等于 query\_data()。当实际上，并不是这么使用，而是这么一个整体：



**被装饰函数参数**

我们被装饰的函数，往往带有参数，因此通过装饰器如何传递参数呢？回想一下，装饰器函数针对被装饰的函数进行装饰，使用的是返回一个 wrapper 函数。其实这个函数可以等同于被装饰的函数，只不过 wrapper 还做了更多的事情。被装饰的函数参数可以通过 wrapper 传递。如下：



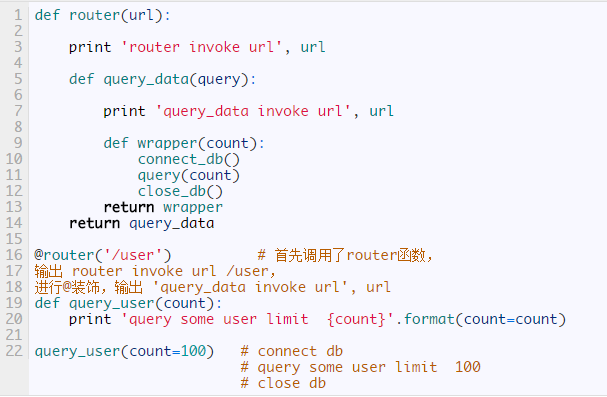
这样就实现了被装饰的函数传递参数。当然，位置参数和关键字参数，可变参数都可以。

**装饰器参数**

在 flask 中，对视图函数的装饰是装饰器中传递 url 正则，即在装饰器中传递参数，和被装饰器的参数还不一样。



我们如何定义router这个装饰器呢？其实只要在原先的装饰器外面再包裹一层，也就是针对装饰器进行装饰。

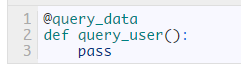


@router() 这个语法糖看上去让人迷惑，其实也很好理解。这里可以看成两个步骤

第一步是调用 router 这个函数:

[6I6R3F7P[5VTB(T2P}C4R7

第二步则进行装饰:



连起来的效果就是

5P[K)NFKF(Z4VTZUKT2Y~CJ

现在回想，@ 这个语法糖很甜吧。并且和python一样很好理解，也十分常用。

当然，我们使用 装饰器是为了实现一些需要包装的方法，例如前面提到的 flask 的 router

有人已经写了一篇很棒的 Tutorial: Things which aren’t magic – Flask and @app.route，

<http://ains.co/blog/things-which-arent-magic-flask-part-1.html>可以参考加深对装饰确定理解，装饰器还有很多用途.