

[首页](#) [资讯](#) [文章](#) [资源](#) [小组](#) [❤ 相亲](#)[频道](#) [🔑 登录](#) [👤 注册](#) [?](#)

学会了面向对象编程, 却找不着对象

[首页](#)
[最新文章](#)
[IT 职场](#)
[前端](#)
[后端](#)
[移动端](#)
[数据库](#)
[运维](#)
[其他技术](#)

- 导航条 -

[伯乐在线](#) > [首页](#) > [所有文章](#) > [Python](#) > Python 2.7.x 和 3.x 版本的重要区别

Python 2.7.x 和 3.x 版本的重要区别

2014/08/02 · [Python](#), [开发](#) · [2 评论](#) · [Python](#)

分享到: 55 本文由 [伯乐在线](#) - [Daetalus](#) 翻译, [黄利民](#) 校稿。未经许可, 禁止转载!

英文出处: nbviewer.ipython.org。欢迎加入[翻译组](#)。

许多Python初学者都会问: 我应该学习哪个版本的Python。对于这个问题, 我的回答通常是“先选择一个最适合你的Python教程, 教程中使用哪个版本的Python, 你就用那个版本。等学得差不多了, 再来研究不同版本之间的差别”。

但如果想要用Python开发一个新项目, 那么该如何选择Python版本呢? 我可以负责任的说, 大部分Python库都同时支持Python 2.7.x和3.x版本的, 所以不论选择哪个版本都是可以的。但为了在使用Python时避开某些版本中一些常见的陷阱, 或需要移植某个Python项目时, 依然有必要了解一下Python两个常见版本之间的主要区别。

目录

- [使用 `future` 模块](#)
- [print函数](#)
- [整数除法](#)
- [Unicode](#)
- [xrange](#)
- [触发异常](#)
- [处理异常](#)
- [next\(\)函数和.next\(\)方法](#)
- [For循环变量与全局命名空间泄漏](#)
- [比较无序类型](#)
- [使用input\(\)解析输入内容](#)
- [返回可迭代对象, 而不是列表](#)
- [更多关于Python 2和Python 3的文章](#)

[首页](#) [future 模块](#) [文章](#) [资源](#) [小组](#) [❤ 相亲](#)[频道](#) ▾[➔ 登录](#)[👤 注册](#)[?](#)[\[回到目录\]](#)

Python 3.x引入了一些与Python 2不兼容的关键字和特性，在Python 2中，可以通过内置的__future__ 模块导入这些新内容。如果你希望在Python 2环境下写的代码也可以在Python 3.x中运行，那么建议使用__future__ 模块。例如，如果希望在Python 2中拥有Python 3.x的整数除法行为，可以通过下面的语句导入相应的模块。

```
Python
1 from __future__ import division
```

下表列出了__future__中其他可导入的特性：

特性	可选版本	强制版本	效果
nested_scopes	2.1.0b1	2.2	PEP 227 : Statically Nested Scopes
generators	2.2.0a1	2.3	PEP 255 : Simple Generators
division	2.2.0a2	3.0	PEP 238 : Changing the Division Operator
absolute_import	2.5.0a1	3.0	PEP 328 : Imports: Multi-Line and Absolute/Relative
with_statement	2.5.0a1	2.6	PEP 343 : The “with” Statement
print_function	2.6.0a2	3.0	PEP 3105 : Make print a function
unicode_literals	2.6.0a2	3.0	PEP 3112 : Bytes literals in Python 3000

（来源：<https://docs.python.org/2/library/future.html>）

示例：

```
Python
1 from platform import python_version
```

print函数

[\[回到目录\]](#)

虽然print语法是Python 3中一个很小的改动，且应该已经广为人知，但依然值得提一下：Python 2中的print语句被Python 3中的print()函数取代，这意味着在Python 3中必须用括号将需要输出的对象括起来。

在Python 2中使用额外的括号也是可以的。但反过来在Python 3中想以Python2的形式不带括号调用print函数时，会触发SyntaxError。

Python 2

首页	资讯	文章 ▾	资源	小组	❤ 相亲	频道 ▾	🔑 登录	👤 注册	Python ?
1	<code>print 'Python', python_version()</code>								
2	<code>print 'Hello, World!'</code>								
3	<code>print('Hello, World!')</code>								
4	<code>print "text", ; print 'print more text on the same line'</code>								

Python									
1	Python 2.7.6								
2	Hello, World!								
3	Hello, World!								
4	text print more text on the same line								

Python 3

Python									
1	<code>print('Python', python_version())</code>								
2	<code>print('Hello, World!')</code>								
3									
4	<code>print("some text,", end="")</code>								
5	<code>print(' print more text on the same line')</code>								

Python									
1	Python 3.4.1								
2	Hello, World!								
3	some text, print more text on the same line								

Python									
1	<code>print 'Hello, World!'</code>								

Python									
1	File "<ipython-input-3-139a7c5835bd>", line 1								
2	<code>print 'Hello, World!'</code>								
3	^								
4	SyntaxError: invalid syntax								

注意：

在Python中，带不带括号输出”Hello World”都很正常。但如果在圆括号中同时输出多个对象时，就会创建一个元组，这是因为在Python 2中，`print`是一个语句，而不是函数调用。

Python									
1	<code>print 'Python', python_version()</code>								
2	<code>print('a', 'b')</code>								
3	<code>print 'a', 'b'</code>								

Python									
1	Python 2.7.7								
2	('a', 'b')								
3	a b								

整数除法

[[回到目录](#)]

由于人们常常会忽视Python 3在整数除法上的改动（写错了也不会触发Syntax Error），所以在移植代码或在Python 2中执行Python 3的代码时，需要特别注意这个改动。

所以，我还是会在Python 3的脚本中尝试用float(3)/2或 3/2.0代替3/2，以此来避免代码在Python 2环境下可能导致的错误（或与之相反，在Python 2脚本中用from __future__ import division来使用Python 3的除法）。

Python 2

```
Python
1 | print 'Python', python_version()
2 | print '3 / 2 =', 3 / 2
3 | print '3 // 2 =', 3 // 2
4 | print '3 / 2.0 =', 3 / 2.0
5 | print '3 // 2.0 =', 3 // 2.0
```

```
Python
1 | Python 2.7.6
2 | 3 / 2 = 1
3 | 3 // 2 = 1
4 | 3 / 2.0 = 1.5
5 | 3 // 2.0 = 1.0
```

Python 3

```
Python
1 | print('Python', python_version())
2 | print('3 / 2 =', 3 / 2)
3 | print('3 // 2 =', 3 // 2)
4 | print('3 / 2.0 =', 3 / 2.0)
5 | print('3 // 2.0 =', 3 // 2.0)
```

```
Python
1 | Python 3.4.1
2 | 3 / 2 = 1.5
3 | 3 // 2 = 1
4 | 3 / 2.0 = 1.5
5 | 3 // 2.0 = 1.0
```

Unicode

[\[回到目录\]](#)

Python 2有基于ASCII的str()类型，其可通过单独的unicode()函数转成unicode类型，但没有byte类型。

而在Python 3中，终于有了Unicode（utf-8）字符串，以及两个字节类：bytes和bytearrays。

Python 2

```
Python
1 | print 'Python', python_version()
```

```
Python
1 | Python 2.7.6
```

```
Python
1 | print type(unicode('this is like a python3 str type'))
```

```
Python
1 | <type 'unicode'>
```

首页	资讯	文章 ▾	资源	小组	❤ 相亲	频道 ▾	🔑 登录	👤 注册	Python ?
1	print type(b'byte type does not exist')								

	Python
1	<type 'str'>

	Python
1	print 'they are really' + b' the same'

	Python
1	they are really the same

	Python
1	print type(bytearray(b'bytearray oddly does exist though'))

	Python
1	<type 'bytearray'>

Python 3

	Python
1	print('Python', python_version())
2	print('strings are now utf-8 u03B8nicou0394é!')

	Python
1	Python 3.4.1
2	strings are now utf-8 μnicoΔé!

	Python
1	print('Python', python_version(), end="")
2	print(' has', type(b' bytes for storing data'))

	Python
1	Python 3.4.1 has <class 'bytes'>

	Python
1	print('and Python', python_version(), end="")
2	print(' also has', type(bytearray(b'bytearrays')))

	Python
1	and Python 3.4.1 also has <class 'bytearray'>

	Python
1	'note that we cannot add a string' + b'bytes for data'

	Python
1	-----
2	TypeError Traceback (most recent call last)
3	<ipython-input-13-d3e8942ccf81> in <module>()
4	----> 1 'note that we cannot add a string' + b'bytes for data'

5 首页 资讯 文章 资源 小组 相亲 对象 to str implicitly
6 TypeError: Can't convert 'bytes' object to str implicitly

频道 ▾

登录

注册



xrange

[[回到目录](#)]

在Python 2.x中，经常会用xrange()创建一个可迭代对象，通常出现在“for循环”或“列表/集合/字典推导式”中。

这种行为与生成器非常相似（如“惰性求值”），但这里的xrange-iterable无尽的，意味着可能在这个xrange上无限迭代。

由于xrange的“惰性求知”特性，如果只需迭代一次（如for循环中），range()通常比xrange()快一些。不过不建议在多次迭代中使用range()，因为range()每次都会在内存中重新生成一个列表。

在Python 3中，range()的实现方式与xrange()函数相同，所以就不存在专用的xrange()（在Python 3中使用xrange()会触发NameError）。

Python

```
1 import timeit
2
3 n = 10000
4 def test_range(n):
5     return for i in range(n):
6         pass
7
8 def test_xrange(n):
9     for i in xrange(n):
10         pass
```

Python 2

Python

```
1 print 'Python', python_version()
2
3 print 'ntiming range()'
4 %timeit test_range(n)
5
6
7 print 'ntiming xrange()'
8 %timeit test_xrange(n)
```

Python

```
1 Python 2.7.6
2
3 timing range()
4 1000 loops, best of 3: 433 µs per loop
5
6 timing xrange()
7 1000 loops, best of 3: 350 µs per loop
```

Python 3

Python

```
1 print('Python', python_version())
2
3 print('ntiming range()')
4 %timeit test_range(n)
```

Python

1	Python 3.4.1	资源	小组	❤ 相亲	频道 ▾	🔑 登录	👤 注册	?
3	timing range()							
4	1000 loops, best of 3: 520 µs per loop							

		Python
1	print(xrange(10))	

		Python
1	-----	
2	NameError Traceback (most recent call last)	
3	in ()	
4	----> 1 print(xrange(10))	
5		
6	NameError: name 'xrange' is not defined	

Python 3中的range对象中的__contains__方法

另一个值得一提的是，在Python 3.x中，range有了一个新的__contains__方法。__contains__方法可以有效的加快Python 3.x中整数和布尔型的“查找”速度。

		Python
1	x = 10000000	
2	def val_in_range(x, val):	
3	return val in range(x)	
4		
5	def val_in_xrange(x, val):	
6	return val in xrange(x)	
7		
8	print('Python', python_version())	
9	assert(val_in_range(x, x/2) == True)	
10	assert(val_in_range(x, x//2) == True)	
11	%timeit val_in_range(x, x/2)	
12		
13	%timeit val_in_range(x, x//2)	

		Python
1	Python 3.4.1	
2	1 loops, best of 3: 742 ms per loop	
3	1000000 loops, best of 3: 1.19 µs per loop	

根据上面的timeit的结果，查找整数比查找浮点数要快大约6万倍。但由于Python 2.x中的range或xrange没有__contains__方法，所以在Python 2中的整数和浮点数的查找速度差别不大。

		Python
1	print 'Python', python_version()	
2		
3	assert(val_in_xrange(x, x/2.0) == True)	
4	assert(val_in_xrange(x, x/2) == True)	
5	assert(val_in_range(x, x/2) == True)	
6	assert(val_in_range(x, x//2) == True)	
7	%timeit val_in_xrange(x, x/2.0)	
8		
9	%timeit val_in_xrange(x, x/2)	
10		
11	%timeit val_in_range(x, x/2.0)	
12		
13	%timeit val_in_range(x, x/2)	

		Python
1	Python 2.7.7	

2 1 loops, best of 3: 285 ms per loop
 3 1 loops, best of 3: 179 ms per loop
 4 1 loops, best of 3: 658 ms per loop
 5 1 loops, best of 3: 556 ms per loop

频道

登录

注册

?

下面的代码证明了Python 2.x中没有__contains__方法:

Python

```
1 print('Python', python_version())
2 range.__contains__
```

Python

```
1 Python 3.4.1
2 <slot wrapper '__contains__' of 'range' objects
```

Python

```
1 print('Python', python_version())
2 range.__contains__
```

Python

```
1 Python 2.7.7
2 -----
3 AttributeError Traceback (most recent call last)
4 <ipython-input-7-05327350dafb> in <module>()
5 1 print 'Python', python_version()
6 ----> 2 range.__contains__
7
8 AttributeError: 'builtin_function_or_method' object has no attribute '__contains__'
```

Python

```
1 print('Python', python_version())
2 xrange.__contains__
```

Python

```
1 Python 2.7.7
2
3 -----
4 AttributeError Traceback (most recent call last)
5 in ()
6 1 print 'Python', python_version()
7 ----> 2 xrange.__contains__
8
9 AttributeError: type object 'xrange' has no attribute '__contains__'
```

关于Python 2中xrange()与Python 3中range()之间的速度差异的一点说明:

有读者指出了Python 3中的range()和Python 2中xrange()执行速度有差异。由于这两者的实现方式相同,因此理论上执行速度应该也是相同的。这里的速度差别仅仅是因为Python 3的总体速度就比Python 2慢。

Python

```
1 def test_while():
2     i = 0
3     while i < 20000:
4         i += 1
5     return
```

Python

```
1 print('Python', python_version())
```


2 `%timeit test_while()` 小组 相亲 频道 登录 注册 ?

```
Python
1 Python 3.4.1
2 %timeit test_while()
3
4 100 loops, best of 3: 2.68 ms per loop
```

```
Python
1 print 'Python', python_version()
2 %timeit test_while()
```

```
Python
1 Python 2.7.6
2 1000 loops, best of 3: 1.72 ms per loop
```

触发异常

[\[回到目录\]](#)

Python 2支持新旧两种异常触发语法，而Python 3只接受带括号的的语法（不然会触发SyntaxError）：

Python 2

```
Python
1 print 'Python', python_version()
```

```
Python
1 Python 2.7.6
```

```
Python
1 raise IOError, "file error"
```

```
Python
1 -----
2 IOError Traceback (most recent call last)
3 <ipython-input-8-25f049caebb0> in <module>()
4 ----> 1 raise IOError, "file error"
5
6 IOError: file error
```

```
Python
1 raise IOError("file error")
```

```
Python
1 -----
2 IOError Traceback (most recent call last)
3 <ipython-input-9-6f1c43f525b2> in <module>()
4 ----> 1 raise IOError("file error")
5
6 IOError: file error
```

Python 3

```
Python
```

首页	资讯	文章	资源	小组	问答	频道	登录	注册	?
Python									
1 Python 3.4.1									
Python									
1 raise IOError, "file error"									
Python									
1 File "<ipython-input-10-25f049caebb0>", line 1 2 raise IOError, "file error" 3 ^ 4 SyntaxError: invalid syntax 5 The proper way to raise an exception in Python 3:									
Python									
1 print('Python', python_version()) 2 raise IOError("file error")									
Python									
1 Python 3.4.1 2 3 ----- 4 OSError Traceback (most recent call last) 5 <ipython-input-11-c350544d15da> in <module>() 6 1 print('Python', python_version()) 7 ----> 2 raise IOError("file error") 8 9 OSError: file error									

异常处理

[\[回到目录\]](#)

Python 3中的异常处理也发生了一点变化。在Python 3中必须使用“as”关键字。

Python 2

Python									
1 print 'Python', python_version() 2 try : 3 let_us_cause_a_NameError 4 except NameError, err: 5 print err, '--> our error message'									
Python									
1 Python 2.7.6 2 name 'let_us_cause_a_NameError' is not defined --> our error message									

Python 3

Python									
1 print('Python', python_version()) 2 try : 3 let_us_cause_a_NameError 4 except NameError as err: 5 print(err, '--> our error message')									

首页	资讯	文章 ▾	资源	小组	❤ 相亲	频道 ▾	🔑 登录	👤 注册	Python ?
1	Python 3.4.1								
2	name 'let_us_cause_a_NameError' is not defined --> our error message								

next()函数和.next()方法

[\[回到目录\]](#)

由于会经常用到next()（.next()）函数（方法），所以还要提到另一个语法改动（实现方面也做了改动）：在Python 2.7.5中，函数形式和方法形式都可以使用，而在Python 3中，只能使用next()函数（试图调用.next()方法会触发AttributeError）。

Python 2

		Python
1	print 'Python', python_version()	
2	my_generator = (letter for letter in 'abcdefg')	
3	next(my_generator)	
4	my_generator.next()	

		Python
1	Python 2.7.6	
2	'b'	

Python 3

		Python
1	print('Python', python_version())	
2	my_generator = (letter for letter in 'abcdefg')	
3	next(my_generator)	

		Python
1	Python 3.4.1	
2	'a'	

		Python
1	my_generator.next()	

		Python
1	-----	
2	AttributeError Traceback (most recent call last)	
3	<ipython-input-14-125f388bb61b> in <module>()	
4	----> 1 my_generator.next()	
5		
6	AttributeError: 'generator' object has no attribute 'next'	

For循环变量与全局命名空间泄漏

[\[回到目录\]](#)

好消息是：在Python 3.x中，for循环中的变量不再会泄漏到全局命名空间中了！

这是Python 3.x中做的一个改动，在“What's New In Python 3.0”中有如下描述：

“列表推导不再支持[... for var in item1, item2, ...]这样的语法，使用[... for var in (item1, item2, ...)]代替。还要注意列表推导有不同的语义：现在列表推导更接近list()构造器中的生成器表达式这样的语法糖，特别要注意的是，循环控制变量不会再泄漏到循环周围的空间中了。”

Python 2

```
Python
1 print 'Python', python_version()
2
3 i = 1
4 print 'before: i =', i
5
6 print 'comprehension: ', [i for i in range(5)]
7
8 print 'after: i =', i
```

```
Python
1 Python 2.7.6
2 before: i = 1
3 comprehension: [0, 1, 2, 3, 4]
4 after: i = 4
```

Python 3

```
Python
1 print('Python', python_version())
2
3 i = 1
4 print('before: i =', i)
5
6 print('comprehension:', [i for i in range(5)])
7
8 print('after: i =', i)
```

```
Python
1 Python 3.4.1
2 before: i = 1
3 comprehension: [0, 1, 2, 3, 4]
4 after: i = 1
```

比较无序类型

[\[回到目录\]](#)

Python 3中另一个优秀的改动是，如果我们试图比较无序类型，会触发一个TypeError。

Python 2

```
Python
1 print 'Python', python_version()
2 print "[1, 2] > 'foo' = ", [1, 2] > 'foo'
3 print "(1, 2) > 'foo' = ", (1, 2) > 'foo'
4 print "[1, 2] > (1, 2) = ", [1, 2] > (1, 2)
```

```
Python
1 Python 2.7.6
2 [1, 2] > 'foo' = False
3 (1, 2) > 'foo' = True
4 [1, 2] > (1, 2) = False
```

Python 3

首页

资讯

文章 ▾

资源

小组

❤ 相亲

频道 ▾

🔑 登录

👤 注册

?

Python

```
1 print('Python', python_version())
2 print("[1, 2] > 'foo' = ", [1, 2] > 'foo')
3 print("(1, 2) > 'foo' = ", (1, 2) > 'foo')
4 print("[1, 2] > (1, 2) = ", [1, 2] > (1, 2))
```

Python

```
1 Python 3.4.1
2 -----
3 TypeError Traceback (most recent call last)
4 <ipython-input-16-a9031729f4a0> in <module>()
5   1 print('Python', python_version())
6   ----> 2 print("[1, 2] > 'foo' = ", [1, 2] > 'foo')
7   3 print("(1, 2) > 'foo' = ", (1, 2) > 'foo')
8   4 print("[1, 2] > (1, 2) = ", [1, 2] > (1, 2))
9   TypeError: unorderable types: list() > str()
```

通过input()解析用户的输入

[\[回到目录\]](#)

幸运的是, Python 3改进了input()函数, 这样该函数就会总是将用户的输入存储为str对象。在Python 2中, 为了避免读取非字符串类型会发生的一些危险行为, 不得不使用raw_input()代替input()。

Python 2

Python

```
1 Python 2.7.6
2 [GCC 4.0.1 (Apple Inc. build 5493)] on darwin
3 Type "help", "copyright", "credits" or "license" for more information.
4
5 >>> my_input = input('enter a number: ')
6
7 enter a number: 123
8
9 >>> type(my_input)
10 <type 'int'>
11
12 >>> my_input = raw_input('enter a number: ')
13
14 enter a number: 123
15
16 >>> type(my_input)
17 <type 'str'>
```

Python 3

Python

```
1 Python 3.4.1
2 [GCC 4.2.1 (Apple Inc. build 5577)] on darwin
3 Type "help", "copyright", "credits" or "license" for more information.
4
5 >>> my_input = input('enter a number: ')
6 enter a number: 123
7 >>> type(my_input)
8 <class 'str'>
```

返回可迭代对象, 而不是列表

[\[回到目录\]](#)

在 `xrange` 一节中可以看到, 某些函数和方法在Python中返回的是可迭代对象, 而不像在Python 2中返回列表。

由于通常对这些对象只遍历一次, 所以这种方式会节省很多内存。然而, 如果通过生成器来多次迭代这些对象, 效率就不高了。

此时我们的确需要列表对象, 可以通过`list()`函数简单的将可迭代对象转成列表。

Python 2

```
Python
1 print 'Python', python_version()
2
3 print range(3)
4 print type(range(3))
```

```
Python
1 Python 2.7.6
2 [0, 1, 2]
3 <type 'list'>
```

Python 3

```
Python
1 print('Python', python_version())
2 print(range(3))
3 print(type(range(3)))
4 print(list(range(3)))
```

```
Python
1 Python 3.4.1
2 range(0, 3)
3 <class 'range'>
4 [0, 1, 2]
```

下面列出了Python 3中其他不再返回列表的常用函数和方法:

- `zip()`
- `map()`
- `filter()`
- 字典的`.key()`方法
- 字典的`.value()`方法
- 字典的`.item()`方法

更多关于Python 2和Python 3的文章

[\[回到目录\]](#)

下面列出了其他一些可以进一步了解Python 2和Python 3的优秀文章,

// 迁移到 Python 3

- [Should I use Python 2 or Python 3 for my development activity?](#)
- [What's New In Python 3.0](#)
- [Porting to Python 3](#)
- [Porting Python 2 Code to Python 3](#)
- [How keep Python 3 moving forward](#)

// 对Python 3的褒与贬

- 首页
- 10 awesome features of Python that you can't use because you refuse to upgrade to Python 3
- 资源
- 文章
- 资源
- 小组
- 相亲
- 频道
- 登录
- 注册
- ?
- 关于你不想知道的所有Python3 unicode特性
 - Python 3 正在毁灭 Python
 - Python 3 能振兴 Python
 - Python 3 is fine

打赏支持我翻译更多好文章，谢谢！

¥ 打赏译者

2 赞

9 收藏

2 评论

关于作者：Daetalus



Pyton核心开源开发者。熟悉CPython实现，关注Python科学计算。

个人主页 · 我的文章 · 28 · 6



相关文章

- 为什么 Python 增长如此之快?
- 150 多个 ML、NLP 和 Python 相关的教程
- 27 个机器学习、数学、Python 速查表 · 2
- JSON 的正确用法: Pyhong、MongoDB、JavaScript与Ajax
- 系统管理员资源大全 · 6

可能感兴趣的话题

- 有哪些进入公司，你才发现的坑
- flask中如何触发request请求的?
- 怎样才能进 BAT? · 1
- 构建高可用、可扩展的redis集群
- 爬虫进阶书籍有哪些?
- 想开发一个类似ogame 的web网页游戏，前端用啥框架好? · 4

登录后评论

新用户注册

直接登录



最新评论



zleung

2015/02/11

函数中出现了 `return for i in range(n):` 这个语句, 哪位大神可以解释一下, 从来没有见过这种语法啊。而且我在 console 里面写的时候也提示出错了。

首页 资讯 文章 资源 小组 相亲 频道 登录 注册 ?

求教!!!

赞 回复



demoZ (10 · 8)

2015/06/27

那是错的。没有return

赞 回复



- [本周热门文章](#)
- [本月热门文章](#)
- [热门标签](#)

0 [惊叹! 这个盲人程序员是这样写代码的](#)

1 [为提高用户体验, Yelp 是如何无损...](#)

2 [每个程序员都该知道的五大定理](#)

3 [为什么 Python 增长如此之快?](#)

4 [MySQL 8.0.3 RC 版即将发...](#)

5 [C# 最佳工具集合: IDE、分...](#)

6 [数据异构的武器-BINLOG+MQ](#)

7 [MySQL 索引设计概要](#)

8 [HttpClient 获取 Cookie 的一...](#)

9 [Linux 26 周年, 来一场怀旧之旅](#)



业界热点资讯

[更多 »](#)



[苹果不再对个人收到的网络打赏抽成](#)

1 天前 · 6

[首页](#) | [资讯](#) | [文章](#) ▾ | [资源](#) | [小组](#) | [❤ 相亲](#)[频道](#) ▾[➔ 登录](#)[👤 注册](#)[微软将推动 Linux 创新？](#)7 小时前 · [👍 2](#)[因专利问题 WordPress 决定停止使用 React](#)17 小时前 · [👍 2](#)[IBM J9 Java 虚拟机正式开源，贡献给 Eclipse ...](#)17 小时前 · [👍 2](#)[Ubuntu 官方开发团队合力让用户更容易从 Unity 转换...](#)18 小时前 · [👍 2](#)[精选工具资源](#)[更多资源 »](#)[Whitewidow: SQL 漏洞自动扫描工具](#)[数据库](#) · [👍 2](#)[Caffe: 一个深度学习框架](#)[机器学习](#)[静态代码分析工具清单：公司篇](#)[静态代码分析](#)

[HotswapAgent: 支持无限次重定义运行时类与资源](#)
[开发流程增强工具](#)

[频道](#)[登录](#)[注册](#)[?](#)

[静态代码分析工具清单：开源篇（各语言）](#)
[静态代码分析](#)

关于伯乐在线博客

在这个信息爆炸的时代，人们已然被大量、快速并且简短的信息所包围。然而，我们相信：过多“快餐”式的阅读只会令人“虚胖”，缺乏实质的内涵。伯乐在线内容团队正试图以我们微薄的力量，把优秀的原创文章和译文分享给读者，为“快餐”添加一些“营养”元素。

快速链接

[网站使用指南](#) »
[问题反馈与求助](#) »
[加入我们](#) »
[网站积分规则](#) »
[网站声望规则](#) »

关注我们

新浪微博: [@伯乐在线官方微博](#)

RSS: [订阅地址](#)

推荐微信号



程序员的那些事



UI设计达人



极客范

合作联系

Email: bd@jobbole.com

QQ: 2302462408（加好友请注明来意）

更多频道

[小组](#) – 好的话题、有启发的回复、值得信赖的圈子

[头条](#) – 分享和发现有价值的内容与观点

[相亲](#) – 为IT单身男女服务的征婚传播平台

[资源](#) – 优秀的工具资源导航

[翻译](#) – 翻译传播优秀的外文文章

[文章](#) – 国内外的精选文章

[设计](#) – UI, 网页, 交互和用户体验

[iOS](#) – 专注iOS技术分享

[安卓](#) – 专注Android技术分享

[前端](#) – JavaScript, HTML5, CSS

[Java](#) – 专注Java技术分享

[Python](#) – 专注Python技术分享

© 2017 伯乐在线

[文章](#) [小组](#) [相亲](#) [加入我们](#) [反馈](#)

[沪ICP备14046347号-1](#)

