Vamei

编程，数学，设计

Python网络02 Python服务器进化

作者：Vamei 出处：http://www.cnblogs.com/vamei 欢迎转载，也请保留这段声明。谢谢！

**注意，在Python 3.x中，BaseHTTPServer, SimpleHTTPServer, CGIHTTPServer整合到http.server包，SocketServer改名为socketserver，请注意查阅官方文档。

在上一篇文章中(用socket写一个Python服务器)，我使用socket接口，制作了一个处理HTTP请求的Python服务器。任何一台装有操作系统和Python解释器的计算机，都可以作为HTTP服务器使用。我将在这里不断改写上一篇文章中的程序，引入更高级的Python包，以写出更成熟的Python服务器。

支持POST

我首先增加该服务器的功能。这里增添了表格，以及处理表格提交数据的"POST"方法。如果你已经读过用socket写一个Python服务器，会发现这里只是增加很少的一点内容。

原始程序：

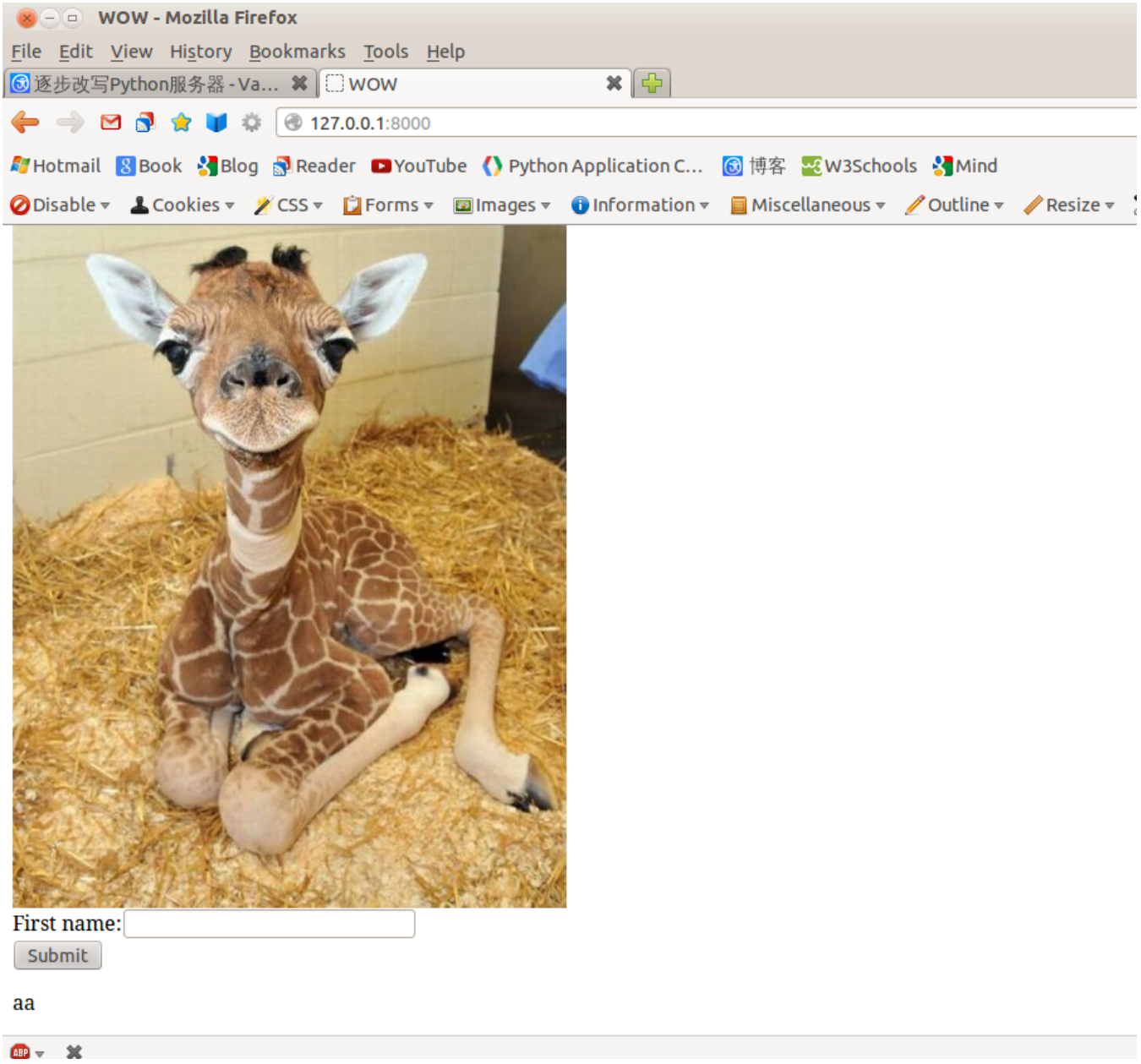
```
# Written by Vamei
# A messy HTTP server based on TCP socket

import socket

# Address
HOST = ''
PORT = 8000

text_content = '''
```

```
HTTP/1.x 200 OK
Content-Type: text/html

<head>
<title>WOW</title>
</head>
<html>
<p>Wow, Python Server</p>
<IMG src="test.jpg"/>
<form name="input" action="/" method="post">
First name:<input type="text" name="firstname"><br>
<input type="submit" value="Submit">
</form>
</html>
'''

f = open('test.jpg','rb')
pic_content = '''
HTTP/1.x 200 OK
Content-Type: image/jpg

'''
pic_content = pic_content + f.read()

# Configure socket
s    = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((HOST, PORT))

# Serve forever
while True:
    s.listen(3)
    conn, addr = s.accept()
    request    = conn.recv(1024)          # 1024 is the
receiving buffer size
    method     = request.split(' ')[0]
    src        = request.split(' ')[1]

    print 'Connected by', addr
    print 'Request is:', request
```

```
    # if GET method request
    if method == 'GET':
        # if ULR is /test.jpg
        if src == '/test.jpg':
            content = pic_content
        else: content = text_content
        # send message
        conn.sendall(content)
    # if POST method request
    if method == 'POST':
        form = request.split('\r\n')
        idx = form.index('')              # Find the empty
line
        entry = form[idx:]                # Main content of
the request

        value = entry[-1].split('=')[-1]
        conn.sendall(text_content + '\n <p>' + value +
'</p>')
        ######
        # More operations, such as put the form into
database
        # ...
        ######
    # close connection
    conn.close()
```

服务器进行的操作很简单，即从POST请求中提取数据，再显示在屏幕上。

运行上面Python服务器，像上一篇文章那样，使用一个浏览器打开。



页面新增了表格和提交(submit)按钮。在表格中输入aa并提交，页面显示出aa。

我下一步要用一些高级包，来简化之前的代码。

使用SocketServer

首先使用SocketServer包来方便的架设服务器。在上面使用socket的过程中，我们先设置了socket的类型，然后依次调用bind(),listen(),accept()，最后使用while循环来让服务器不断的接受请求。上面的这些步骤可以通过

SocketServer包来简化。

SocketServer:

```
# Written by Vamei
# use TCPServer

import SocketServer

HOST = ''
PORT = 8000

text_content = '''
HTTP/1.x 200 OK
Content-Type: text/html

<head>
<title>WOW</title>
</head>
<html>
<p>Wow, Python Server</p>
<IMG src="test.jpg"/>
<form name="input" action="/" method="post">
First name:<input type="text" name="firstname"><br>
<input type="submit" value="Submit">
</form>
</html>
'''

f = open('test.jpg','rb')
pic_content = '''
HTTP/1.x 200 OK
Content-Type: image/jpg

'''
pic_content = pic_content + f.read()

# This class defines response to each request
class MyTCPHandler(SocketServer.BaseRequestHandler):
    def handle(self):
```

```
        # self.request is the TCP socket connected to the
client
        request = self.request.recv(1024)

        print 'Connected by',self.client_address[0]
        print 'Request is', request

        method     = request.split(' ')[0]
        src        = request.split(' ')[1]

        if method == 'GET':
            if src == '/test.jpg':
                content = pic_content
            else: content = text_content
            self.request.sendall(content)

        if method == 'POST':
            form = request.split('\r\n')
            idx = form.index('')              # Find the
empty line
            entry = form[idx:]                # Main content
of the request

            value = entry[-1].split('=')[-1]
            self.request.sendall(text_content + '\n <p>' +
value + '</p>')
            ######
            # More operations, such as put the form into
database
            # ...
            ######


# Create the server
server = SocketServer.TCPServer((HOST, PORT), MyTCPHandler)
# Start the server, and work forever
server.serve_forever()
```

我建立了一个TCPServer对象，即一个使用TCP socket的服务器。在建立TCPServe的同时，设置该服务器的IP地址和端口。使用server_forever()方法来让服务器不断工作(就像原始程序中的while循环一样)。

我们传递给TCPServer一个MyTCPHandler类。这个类定义了如何操作socket。MyTCPHandler继承自BaseRequestHandler。改写handler()方法，来具体规定不同情况下服务器的操作。

在handler()中，通过self.request来查询通过socket进入服务器的请求 (正如我们在handler()中对socket进行recv()和sendall()操作)，还使用self.address来引用socket的客户端地址。

经过SocketServer的改造之后，代码还是不够简单。 我们上面的通信基于TCP协议，而不是HTTP协议。因此，我们必须手动的解析HTTP协议。我们将建立基于HTTP协议的服务器。

SimpleHTTPServer: 使用静态文件来回应请求

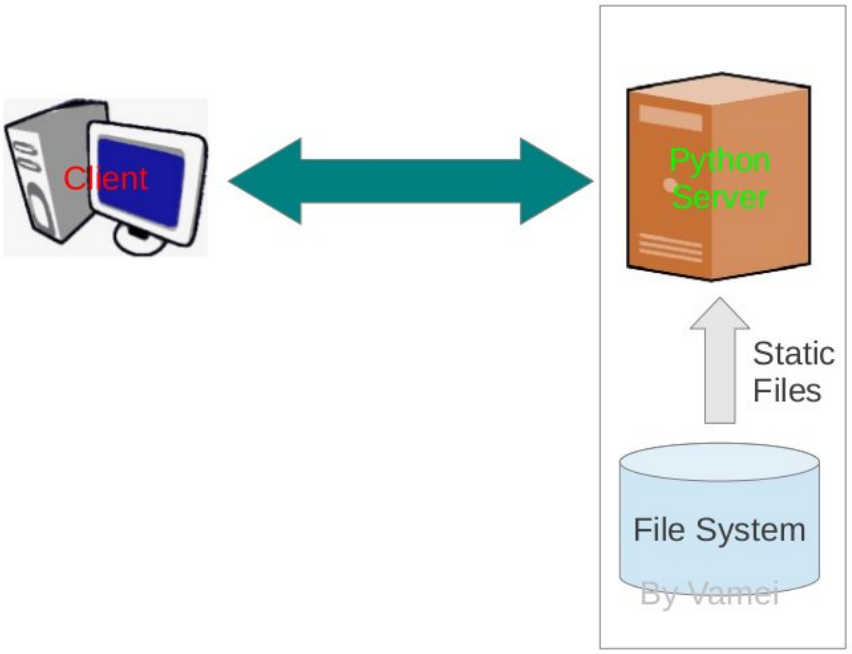
HTTP协议基于TCP协议，但增加了更多的规范。这些规范，虽然限制了TCP协议的功能，但大大提高了信息封装和提取的方便程度。

对于一个HTTP请求(request)来说，它包含有两个重要信息：请求方法和URL。

请求方法(request method)	URL	操作
GET	/	发送text_content
GET	/text.jpg	发送pic_content
POST	/	分析request主体中包含的value(实际上是我们填入表格的内容)； 发送text_content和value

根据请求方法和URL的不同，一个大型的HTTP服务器可以应付成千上万种不同的请求。在Python中，我们可以使用SimpleHTTPServer包和CGIHTTPServer包来

规定针对不同请求的操作。其中，SimpleHTTPServer可以用于处理GET方法和HEAD方法的请求。它读取request中的URL地址，找到对应的静态文件，分析文件类型，用HTTP协议将文件发送给客户。



SimpleHTTPServer

我们将text_content放置在index.html中，并单独存储text.jpg文件。如果URL指向index_html的母文件夹时，SimpleHTTPServer会读取该文件夹下的index.html文件。

我在当前目录下生成index.html文件：

```
<head>
<title>WOW</title>
</head>
<html>
<p>Wow, Python Server</p>
<IMG src="test.jpg"/>
<form name="input" action="/" method="post">
```

```
First name:<input type="text" name="firstname"><br>
<input type="submit" value="Submit">
</form>
</html>
```

改写Python服务器程序。使用SimpleHTTPServer包中唯一的类SimpleHTTPRequestHandler:

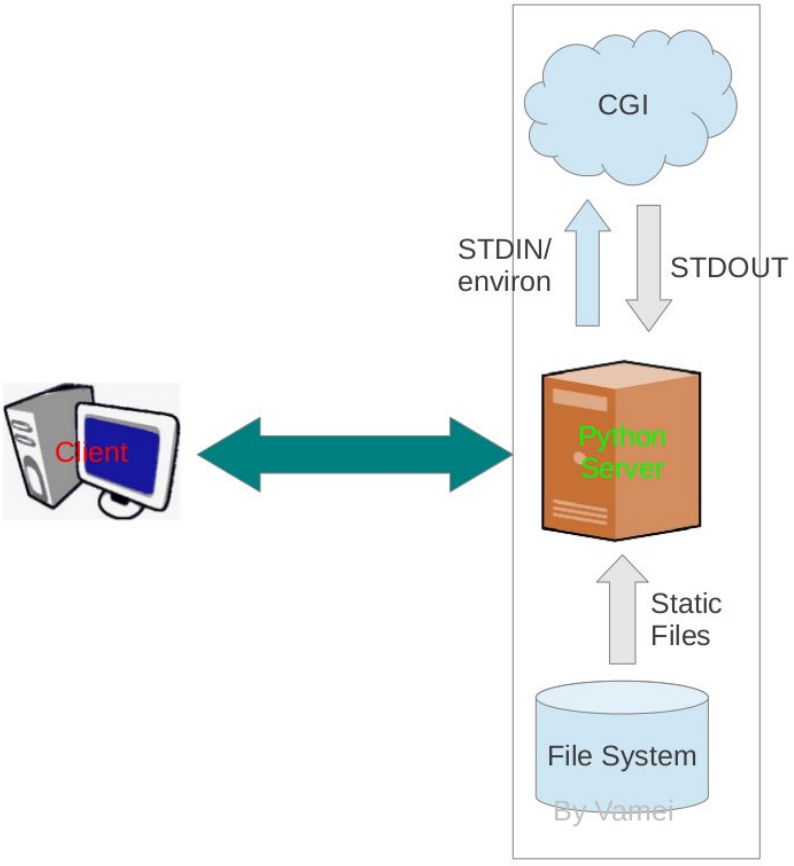
```
# Written by Vamei
# Simple HTTPsERVER

import SocketServer
import SimpleHTTPServer

HOST = ''
PORT = 8000

# Create the server, SimpleHTTPRequestHander is pre-defined handler in SimpleHTTPServer package
server = SocketServer.TCPServer((HOST, PORT), SimpleHTTPServer.SimpleHTTPRequestHandler)
# Start the server
server.serve_forever()
```

这里的程序不能处理POST请求。我会在后面使用CGI来弥补这个缺陷。值得注意的是，Python服务器程序变得非常简单。将内容存放于静态文件，并根据URL为客户端提供内容，这让内容和服务器逻辑分离。每次更新内容时，我可以只修改静态文件，而不用停止整个Python服务器。

这些改进也付出代价。在原始程序中，request中的URL只具有指导意义，我可以规定任意的操作。在SimpleHTTPServer中，操作与URL的指向密切相关。我用自由度，换来了更加简洁的程序。

CGIHTTPServer：使用静态文件或者CGI来回应请求

CGIHTTPServer包中的CGIHTTPRequestHandler类继承自SimpleHTTPRequestHandler类，所以可以用来代替上面的例子，来提供静态文件的服务。此外，CGIHTTPRequestHandler类还可以用来运行CGI脚本。



CGIHTTPServer

先看看什么是CGI (Common Gateway Interface)。CGI是服务器和应用脚本之间的一套接口标准。它的功能是让服务器程序运行脚本程序，将程序的输出作为response发送给客户。总体的效果，是允许服务器动态的生成回复内容，而不必局限于静态文件。

支持CGI的服务器程接收到客户的请求，根据请求中的URL，运行对应的脚本文件。服务器会将HTTP请求的信息和socket信息传递给脚本文件，并等待脚本的输出。脚本的输出封装成合法的HTTP回复，发送给客户。CGI可以充分发挥服务器的可编程性，让服务器变得"更聪明"。

服务器和CGI脚本之间的通信要符合CGI标准。CGI的实现方式有很多，比如说使用

Apache服务器与Perl写的CGI脚本，或者Python服务器与shell写的CGI脚本。

为了使用CGI，我们需要使用BaseHTTPServer包中的HTTPServer类来构建服务器。Python服务器的改动很简单。

CGIHTTPServer:

```
# Written by Vamei
# A messy HTTP server based on TCP socket

import BaseHTTPServer
import CGIHTTPServer

HOST = ''
PORT = 8000

# Create the server, CGIHTTPRequestHandler is pre-defined handler
server = BaseHTTPServer.HTTPServer((HOST, PORT), CGIHTTPServer.CGIHTTPRequestHandler)
# Start the server
server.serve_forever()
```

CGIHTTPRequestHandler默认当前目录下的cgi-bin和ht-bin文件夹中的文件为CGI脚本，而存放于其他地方的文件被认为是静态文件。因此，我们需要修改一下index.html，将其中form元素指向的action改为cgi-bin/post.py。

```
<head>
<title>WOW</title>
</head>
<html>
<p>Wow, Python Server</p>
<IMG src="test.jpg"/>
<form name="input" action="cgi-bin/post.py" method="post">
```

```
First name:<input type="text" name="firstname"><br>
<input type="submit" value="Submit">
</form>
</html>
```

我创建一个cgi-bin的文件夹，并在cgi-bin中放入如下post.py文件，也就是我们的CGI脚本：

```
#!/usr/bin/env python
# Written by Vamei
import cgi
form = cgi.FieldStorage()

# Output to stdout, CGIHttpServer will take this as
response to the client
print "Content-Type: text/html"     # HTML is following
print                               # blank line, end of
headers
print "<p>Hello world!</p>"         # Start of content
print "<p>" +  repr(form['firstname']) + "</p>"
```

(post.py需要有执行权限，见评论区)

第一行说明了脚本所使用的语言，即Python。 cgi包用于提取请求中包含的表格信息。脚本只负责将所有的结果输出到标准输出(使用print)。CGIHTTPRequestHandler会收集这些输出，封装成HTTP回复，传送给客户端。

对于POST方法的请求，它的URL需要指向一个CGI脚本(也就是在cgi-bin或者ht-bin中的文件)。CGIHTTPRequestHandler继承自SimpleHTTPRequestHandler，所以也可以处理GET方法和HEAD方法的请求。此时，如果URL指向CGI脚本时，服务器将脚本的运行结果传送到客户端；当此时URL指向静态文件时，服务器将文件的内容传送到客户端。

更进一步，我可以让CGI脚本执行数据库操作，比如将接收到的数据放入到数据库中，以及更丰富的程序操作。相关内容从略。

总结

我使用了Python标准库中的一些高级包简化了Python服务器。最终的效果分离静态内容、CGI应用和服务器，降低三者之间的耦合，让代码变得简单而容易维护。

希望你享受在自己的电脑上架设服务器的过程。

分类: Python

标签: Python

Vamei
关注 - 26
粉丝 - 4985

« 上一篇：Python网络01 原始Python服务器
» 下一篇：Python补充02 Python小技巧

posted @ 2012-10-31 17:57 Vamei 阅读(13637) 评论(12) 编辑 收藏

评论列表

#1楼 2012-11-01 17:54 HackerVirus

顶楼主

支持(0) 反对(0)

#2楼 2013-01-16 23:22 bells

这里要注意了，post.py 要加个"x"权限

支持(1) 反对(0)

#3楼[楼主] 2013-01-17 10:33 Vamei

@ bells

嗯，应该要加的。

支持(0) 反对(0)

#4楼 2013-02-20 15:20 猪在飞啊

楼主，一步一步连过来，前面教程都没有问题，这里算是卡住了.....
到了CGI例子，执行post后，页面显示Error response

Error code 501.

Message: Can only POST to CGI scripts.

Error code explanation: 501 = Server does not support this operation.

请问这是什么原因，新手，网查了下还是没有搞定，windows系统。
BTW，楼主的教程很好，希望多写些py的教程。
请楼主赐教。
还有有代码洁癖的人觉得HTML文件内容应该是下面这样啊

```
<html>
<head>
    <title>WOW</title>
</head>
<body>
    <form name="input" action="cgi-bin/post.py" method="post">
        <p>Wow, Python Server</p>
        <img src="test.jpg" />
        First name:<input type="text" name="firstname"><br>
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

支持(0) 反对(0)

#5楼[楼主] 2013-02-20 17:28 Vamei

@ 中国大兵

下面是官方文档中的解释：
class CGIHTTPServer.CGIHTTPRequestHandler(request, client_address,

server)

...

do_POST()

This method serves the 'POST' request type, only allowed for CGI scripts. Error 501, “Can only POST to CGI scripts”, is output when trying to POST to a non-CGI url.

所以，并没有指向一个CGI脚本。有可能是权限的问题。作为一个CGI脚本，必须要有执行权限。对于Windows，我不知道怎么来修改权限。你沿着这个方向搜索一下吧。

支持(0) 反对(0)

#6楼 2013-11-10 08:54 szxiaosong

post.py需要执行权限

chmod +x cgi-bin/post.py

支持(0) 反对(0)

#7楼[楼主] 2013-11-10 18:51 Vamei

@ szxiaosong

谢谢提醒啊！

支持(0) 反对(0)

#8楼 2014-03-09 02:17 1万小时理论

我是一个python初学者，以后有问题向你请教啊。关注你了。

支持(0) 反对(0)

#9楼 2014-03-19 10:23 gitbuild

底层socket模块，学习完了之后，写了一个例子，支持GET POST。

```
+ View Code
```

支持(0) 反对(0)

#10楼 2014-03-19 14:13 ray007great

你好，我想问下，如果我在本机搭好了这么一个服务器环境，理论上，挂上域名，连上宽带，就可以提供访问外网访问功能了？

支持(0) 反对(0)

#11楼[楼主] 2014-03-19 14:38 Vamei

@ ray007great

如果你有一个wifi，可以试验一下。只要找出自己电脑的IP地址就好。用另一台电脑或者手机访问。

但一般我们不这样。原因是现在有更好的工具，比如apache。可以更安全，更严密的工作。

支持(0) 反对(0)

#12楼[楼主] 2014-03-19 14:38 Vamei

@ 1万小时理论

加油！

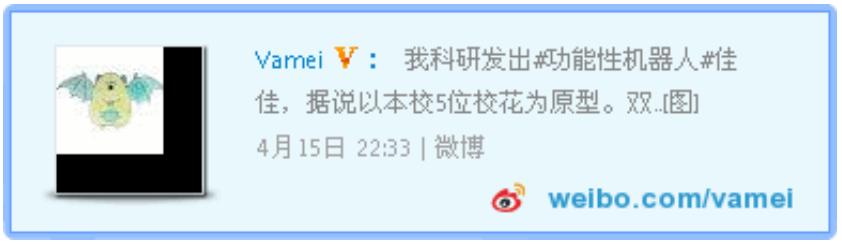
支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

公告

你好，这里是Vamei，一名编程爱好者。我在博客里写了**Python/Linux/**网络协议**/**算法**/Java/**数据科学系列文章，从这里开始阅读。非常期待和你的交流。



我的微博

下列教程已经做成电子出版物，内容经过修订，也方便离线阅读：

协议森林

欢迎阅读我写的其他书籍：

现代小城的考古学家

天气与历史的相爱相杀

随手拍光影

昵称：Vamei
园龄：4年1个月
荣誉：推荐博客
粉丝：4985
关注：26
+加关注

常用链接

我的随笔

我的评论

我的参与

最新评论

我的标签

我的标签

Python(61)

Java(42)

大数据(22)

Linux(17)

网络(16)

算法(15)

文青(14)

技普(9)

系列索引(6)

开发工具(4)

更多

系列文章

Java快速教程

Linux的概念与体系

Python快速教程

数据科学

协议森林

纸上谈兵: 算法与数据结构

积分与排名

积分 - 659668

排名 - 122

最新评论

1. Re:Java基础11 对象引用

受教!

--MisslLost

2. Re:Python快速教程

看评论区一片喝彩！看来我得在此扎营了！

--测试小蚂蚁

3. Re:Python进阶06 循环对象

好好地列表解析变成了表推导

--ashic

4. Re:“不给力啊，老湿！”：RSA加密与破解

感谢楼主精彩分享

--worldball

5. Re:概率论04 随机变量

你写的这一系列太棒了，刚加入博客园就在你这里学到了，我要转载过去学习一下

--yixius

6. Re:Python基础03 序列

挺好的教程、、、、

--王小拽的号

7. Re:Python进阶07 函数对象

def func(x,y): print x**ydef test(f,a,b): print 'test' print f(a,b)test (func,3,2)

输出的内容:tes.....

--M-edea

8. Re:Python进阶02 文本文件的输入输出

@coderXT换行符：\n...

--行者之印

9. Re:数据科学

博主啊，这里是一枚即将大二的计算机新人，大一学了python，java，还有一些算法，数据结构，图论了，感觉我对数学又一些反感，但是听说离散数学对计算机专业的很重要，不知道怎么去学比较好呢，我想像您写.....

--Acokil

10. Re:为什么要写技术博

楼主是用自己自定义的模板吗？在博客园里找不到这种风格的blog模板？

--行者之印

11. Re:来玩Play框架01 简介

挖煤哥,我补充了一下Windows下的搭建play框架,希望有点帮助,谢谢!

--Sungeek

12. Re:来玩Play框架07 静态文件

@helper.form(action = routes.Application.upload, 'enctype -> "multipart/form-data") {--action = rout.....

--quxiaozha

13. Re:来玩Play框架07 静态文件

该记录将/assets/下的URL，对应到项目的/public文件夹内的文件。比如在项目的/public/images/test.jpg，就可以通过/assests/images/test.jpg这一.....

--quxiaozha

14. Re:来玩Play框架06 用户验证

支持挖煤哥~~~

--quxiaozha

15. Re:“不给力啊，老湿！”：RSA加密与破解

@maanshancss请你仔细阅读了这个文章再来评价。...

--Vamei

推荐排行榜

1. “不给力啊，老湿！”：RSA加密与破解(218)
2. Python快速教程(140)
3. 野蛮生长又五年(91)