第二十章 Web编程

Python

20.1 介绍

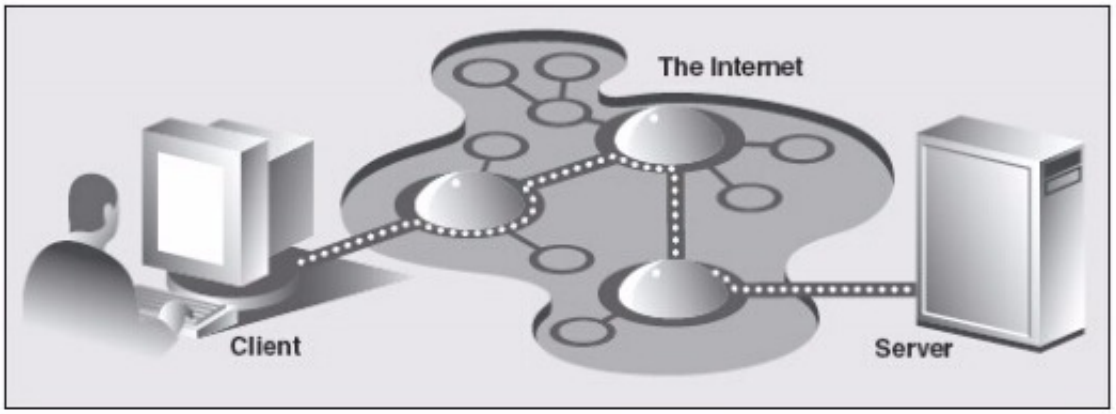


图 20-1 因特网上的 Web 客户端和 Web 服务器。在因特网上客户端向服务器端发送一个请求，然后服务器端响应这个请求并将相应的数据返回给客户端。

客户端可能向服务器端发出各种请求。这些请求可能获得一个网页视图或者提交一个数据表单。请求经过服务器端的处理，然后会以特定的格式HTML等返回给客户端浏览。

Web客户端和服务器端使用HTTP协议来交互。HTTP协议是TCP/IP协议的上层协议,由TCP/IP协议负责路由和传递消息,而HTTP协议负责发送、接收HTTP消息来处理客户端的请求。

HTTP 协议属于无状态协议，它不跟踪从一个客户端到另一个客户端的的请求信息。由于每个请求缺乏上下文背景，你可以注意到有些 URL 会有很长的变量和值作为请求的一部分，以便提供一些状态信息。

客户端最终连接到服务器的通路，实际包含了不定节点的连通。
这些不定点的连通被简化为从客户端到所访问的服务器端的直接连接。
这些不定点的连通有TCP/IP负责,而HTTP负责将用户的数据发送/接收 服务器的响应消息。

20.2 Python的Web应用

浏览器只是 Web 客户端的一种。任何一个通过向服务器端发送请求来获得
数据的应用程序都被认为是“客户端”。

URL(Web地址)用来在Web上定位一个文档,URL是URI(统一资源标识)的一部分。
Web地址的格式如下:

`prot_sch://net_loc/path;params?query#frag`

`proc_sch` –网络协议或下载规则
`net_loc` –服务器位置(或许也有用户信息)
`path` –斜杠(/)限定文件或者CGI应用程序的路径
`param` –可选参数
`query` –连接符(&)连接键值对
`frag` –拆分文档中的特殊锚

> https://docs.python.org/2/library/tkinter.html#tkinter-life-preserver

`net_loc` 可以进一步拆分为多个组成部分
`user:passwd@host:port`
`host` 为主机名
`port` 端口号只有在 Web 服务器运行其他非默认端口上时才会被使用。
`user:passwd` 户名和密码部分只有在使用 FTP 连接时候才有可能用到，因为即使是使用 FTP，大多数的连接都是使用匿名这时是不需要用户名和密码的。

Python 支持两种不同的模块，分别以不同的功能和兼容性来处理 URL。一种是 urlparse，一种是 urllib。

20.2.2 urlparse模块

urlparse提供了对URL字符串操作的基本功能,包括 `urlparse(),urlunparse(),urljoin()`

`urlparse()` 将 URL 字符串拆分成如上所描述的一些主要部件
`urlparse(urlstr, defProtSch=None, allowFrag=None)`

`urlparse()` 方法将 `urlstr` 解析成一个 6-元组
`(prot_sch, net_loc,path, params, query,frag)` .

`urlunparse()` 的功能与 `urlpase()` 完全相反
`urljoin()` 需要多个相关的 URL 时,如在一个 Web 页中生成的一系列页面的 URL

```
import urlparse

def web1():
    urlstr = urlparse.urlparse('https://docs.python.org/2/library/urllib2.html#openerdire
ctor-objects')
    print urlstr
    for ele in urlstr:
        if ele:
            print ele
    url2 = urlparse.urlunparse(urlstr) #拼合urltuple组成一个新的网址
    print url2
```

20.2.3 urllib模块

urllib 提供了了一个高级的 Web 交流库，支持 Web 协议， HTTP， FTP 和 Gopher 协议，同时也支持对本地文件的访问。urllib 模块的特殊功能是利用上述协议下载数据(从因特网、局域网、主机上下载)。

使用这个模块可以避免使用 httplib， ftplib 和 gopherlib 这些模块，除非你想用更低层的功能。

> 大多数以*lib 命名的模块用于客户端相关协议开发。

urllib.urlopen()

GET 和 POST 请求是向 Web 服务器上传数据的两种方法。
一旦连接成功， `urlopen()` 将会返回一个文件类型对象，就像在目标路径下打开了一个可读文件。例如返回的文件对象为f,则这个文件句柄f支持常用的文件方法。

`f.info()` 方法可以返回 MIME头文件,通知浏览器返回的可以用哪类应用程序打开。如浏览器本身可以查看HTML,文本文件,生成PNG文件,JPEG或者GIF文件。
常用方法

Table 20.4 urllib.urlopen() File-like Object Methods

urlopen() 对象方法	描述
f.read([bytes])	从 f 中读出所有或 bytes 个字节
f.readline()	从 f 中读出一行
f.readlines()	从 f 中读出所有行并返回一个列表
f.close()	关闭 f 的 URL 的连接
f.fileno()	返回 f 的文件句柄
f.info()	获得 f 的 MIME 头文件
f.geturl()	返回 f 所打开的真正的 URL

如果访问的url有基于数字权限验证,cookie,重定位等问题可以使用urllib2模块。

urllib.urlretrieve()

`urlretrieve(urlstr, localfile=None, downloadStatusHook=None)`
如果该文件已经被复制到本地或者已经是一个本地文件，后续的下载动作将不会发生。
`downloadStatusHook` 这个函数将会在每块数据下载或传输完成后被调用。

`urlretrieve()` 返回一个2-元组, `(filename,mime_headers)` ,第一个参数包含下载数据的本地文件名，第二个参数是对Web服务器响应后返回的一些列MIME文件头。可以用mimetool的Message类查看更多的信息。

urllib.quote(),urllib.quote_plus

`quote` 的功能:将一下不能被打印的或者不被Web服务器作为有效URL接收的特殊字符串转换。
另外，
那些不被允许的字符前边会被加上百分号(%)同时转换成 16 进制,例如：“%xx”，“xx”代表这个字母的 ASCII 码的十六进制值。
调用 quote*()时，urldata 字符串被转换成了一个可在 URL 字符串中使用的等价值。

> 这就是在URL中经常看见有%后面跟着几个数字或者一些字符的原因。
> `quote(urldata, safe='/')`
> safe中是不需要被转换的字符,默认为斜线。

urllib.unquote(),urllib.unquote_plus()

与 quote*()函数的功能安全相反—它将所有编码为 “%xx”式的字母都转换成它们的 ASCII 码值。

关于urllib的常用函数

在表 20.5 中可以看到关于本节讨论的 urllib 函数的概要总结。

Table 20.5 Core urllib Module Functions

urllib 函数	描述
urlopen(urlstr, postQuery- Data=None)	打开 URL urlstr,如果必要则通过 postQueryData 发送请求。
urlretrieve(urlstr, local- tusHook=None)	将 URL urlstr 定位的文件下载到 localfile 或临时文件中（当 localfile 没有给定时）；如果文件已经存在 downloaStatusHook 将会获得下载的统计信息。
quote(urldata, safe='/')	将 urldata 的无效的 URL 字符编码；在 safe 列的则不必编码。
quote_plus(urldata, safe='/')	将空格编译成加(+)号(并非%20)外，其他功能与 quote()相似。
unquote(urldata)	将 urldata 中编码后的字母解码
unquote_plus(urldata)	除了将加好转换成空格后其他功能与 unquote()相似。
urlencode(dict)	将字典键-值对编译成有效的 CGI 请求字符串，用 quote_plus()对键和值字符串分别编码。

20.2.4 urllib2模块

20.3 高级 Web 客户端

Web浏览器是基本的Web客户端，主要用来浏览和下载文件。高级的Web客户端还可以基于不同的目的在Internet上搜索和下载页面。

- Google搜索引擎建索引
- 脱机浏览文档(将文档下载到本地,重新设定了超链接)
- 下载并保存历史纪录
- Web页面的缓存，节省再次访问Web站点的时间

```
# 20.3 高级Web客户端
# 网络爬虫:抓取Web的开始页面地址,下载该页面和其他后面的连接也main,但是
# 仅限于哪些与开始页面有着相同域名的页面。

from sys import argv
from os import makedirs, unlink, sep
from os.path import  dirname, exists, isdir, splitext
from string import replace, find,lower
from htmllib import HTMLParser
from urllib import urlretrieve
from urlparse import urlparse, urljoin
from formatter import DumbWriter,AbstractFormatter
from cStringIO import StringIO


class Retriever(object): # download web pages
    def __init__(self, url):
        self.url = url
        self.file = self.filename(url)

    def filename(self, url, deffile='index.htm'):
        parsedurl = urlparse(url,'http:',0)## parse path
        path = parsedurl[1] + parsedurl[2]
        ext = splitext(path)
        if ext[1]=='':  pass # no file
        if path[-1]=='/':
            path += deffile
```

```
            else:
                path += '/' + deffile
                ldir = dirname(path) # local directory

            if sep != '/':
                ldir = replace(ldir,'/',sep)
                if not isdir(ldir):
                    makedirs(ldir)
            return path

        def download(self):
            try:
                retval = urlretrieve(self.url, self.file)
            except IOError:
                retval = ('*** ERROR: invalid URL "%s" ' % self.url)
            return retval

        def parseAndGetLinks(self): # parse HTML,save links
            self.parser = HTMLParser(AbstractFormatter(DumbWriter(StringIO())))
            self.parser.feed(open(self.file).read())
            self.parser.close()
            return self.parser.anchorlist

    class Crawler(object): # manage centire crawling process
        count = 0
        def __init__(self, url):
            self.q = [url]
            self.seen = []
```

```
        self.dom = urlparse(url)[1]

    def getPage(self, url):
        r = Retriever(url)
        retval = r.download()
        if retval[0] == '*': # error situation,do not parse
            print retval,'... skipping parse'
            return
        Crawler.count += 1
        print '\n(.',Crawler.count,')'
        print 'URL:',url
        print 'FILE:',retval[0]
        self.seen.append(url)


        links = r.parseAndGetLinks() # get and process links
        for eachLinks in links:
            if eachLinks[:4] != 'http' and find(eachLinks,'://') == -1:
                eachLinks = urljoin(url,eachLinks)
                print '*',eachLinks

            if find(lower(eachLinks),'mailto:') != -1:
                print '... discarded, mailto link'
                continue

            if eachLinks not in self.seen:
                if find(eachLinks, self.dom):
                    print '... discarded, not in domain'
```

```
                else:
                    if eachLinks not in self.q:
                        self.q.append(eachLinks)
                        print 'new, added to Q'
                    else:
                        print '... discarded, already in Q'
            else:
                print '... discarded, already processed'

    def go(self): # process links in queue
        while self.q:
            url = self.q.pop()
            self.getPage(url)

def main():
    if len(argv) > 1:
        url = argv[1]
    else:
        try:
            url = 'http://langeasy.com.cn/m/player?f=2012081220200810000239'#raw_input
('Enter starting URL:')
        except (KeyboardInterrupt, EOFError):
            url = ''
    if not url:
        return

    robot = Crawler(url)
    robot.go()
```

```
if __name__ == "__main__":
    main()
```

20.4 CGI:帮助Web服务器处理客户端数据

一个用户从提交表单到返回最终结果 Web 页面的整个执行过程和数据流。

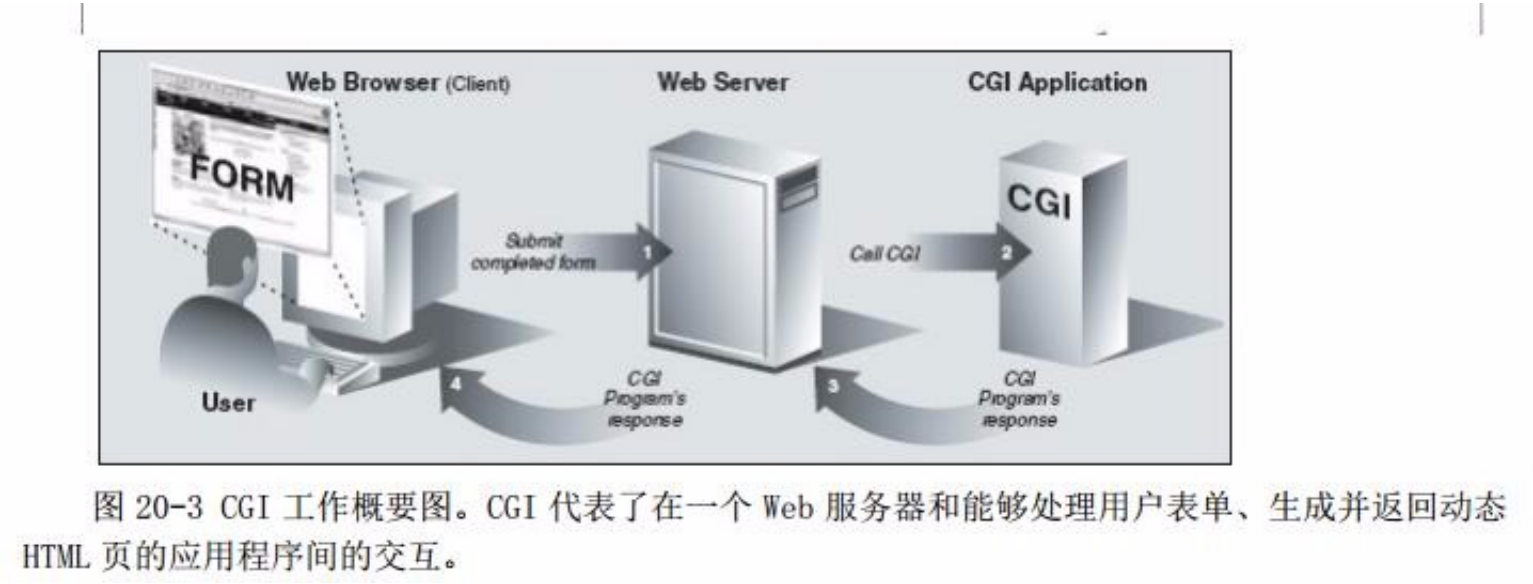
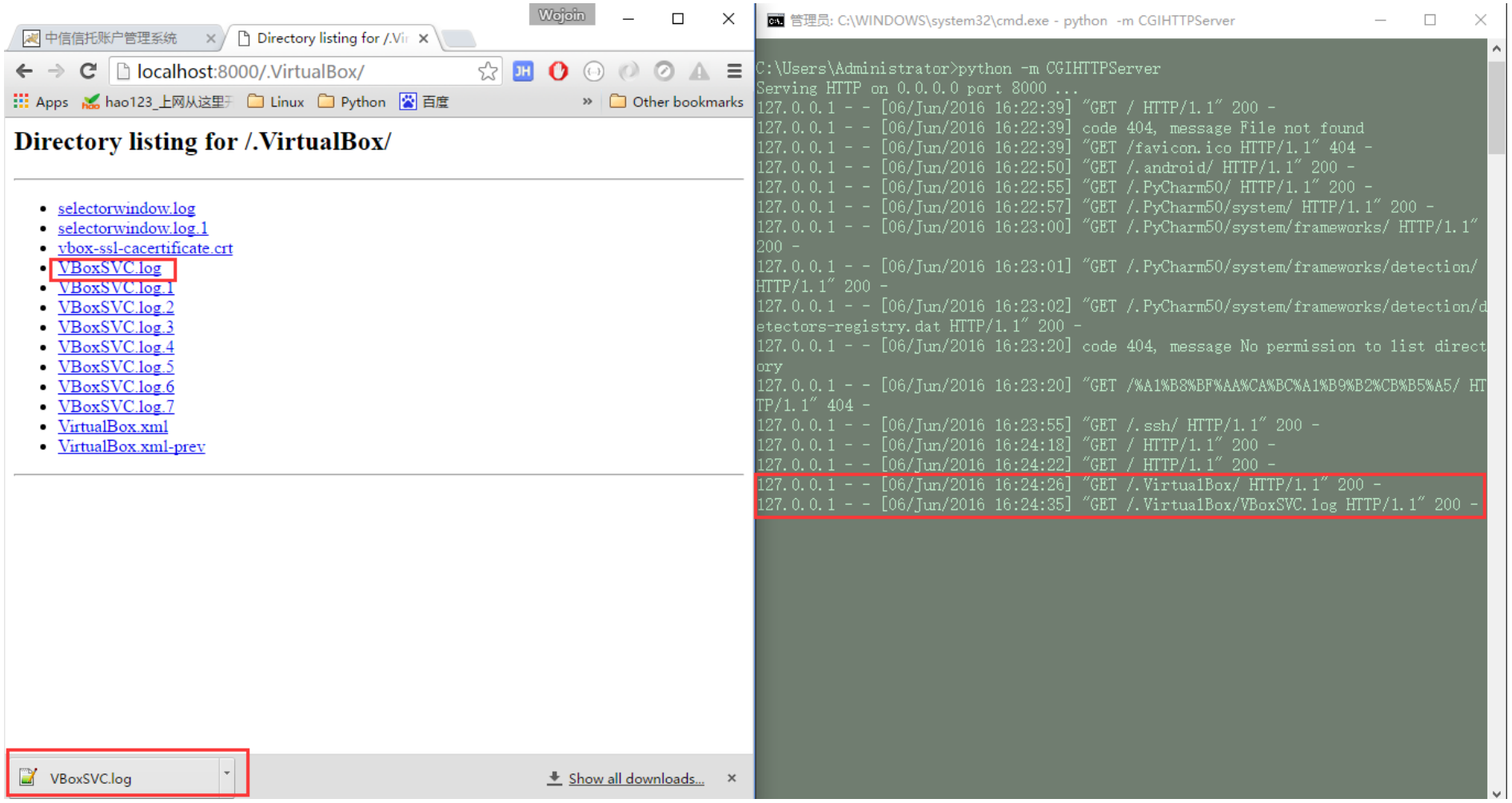


图 20-3 CGI 工作概要图。CGI 代表了在一个 Web 服务器和能够处理用户表单、生成并返回动态 HTML 页的应用程序间的交互。

CGI 是一个适用于小型 Web 网站开发的工具。

20.5 建立CGI应用程序

如何建立和配置简单的基于 Python 的 Web 服务器。 20.8



在启动Python自带的服务器CGIHTTPServer后

> 在浏览器中输入主目录(localhost:8000)后进入了主页面,然后点击了左半边的(.VirtualBox/–在前一个页面中),然后就会在右半边显示红线框中的第一调信息.
> 当点击页面中的(VBoxSVC.log)时,在浏览器中直接把这个文件下载下来了,而右半边的CGIHTTPServer服务器中显示的是红线中的第二条信息。

20.7 高级 CGI

最常见的情况就是你有一系列的复选框允许用户有多个选择。**在表单提交时，数据从用户端以键-值对形式发送到服务器端。**

一旦在客户端建立了 cookie，HTTP_COOKIE 环境变量会将那些 cookie 自动放到请求中发送给服务器。

Cookie **RFC2019**

```
from cgi import  FieldStorage
from os import environ
from cStringIO import StringIO
from urllib import quote,unquote
from string import capwords,strip,split,join

class AdvCGI(object):
    header = 'Content-Type:text/html\n\n'
    url = '/py/advcgi.py'

    # form的HTML
    formhtml = '''<HTML><HEAD><TITLE>
    Advanced CGI Demo</TITLE></HEAD>
    <BODY><H2>Advanced CGI Demo Form</H2>
    <FORM METHOD=post ACTION="%s" ENCTYPE="multipart/form-data">
    <H3>My Cookie Setting</H3>
    <LI> <CODE><B>CPPuser = %s</B></CODE>
    <H3>Enter cookie value<BR>
    <INPUT NAME=cookie value="%s"> (<I>optional</I>)</H3>
    <H3>Enter your name<BR>
    <INPUT NAME=person VALUE="%s"> (<I>required</I>)</H3>
    <H3>What languages can you program in?
    (<I>at least one required</I>)</H3>
    %s
    <H3>Enter file to upload</H3>
    <INPUT TYPE=file NAME=upfile VALUE="%s" SIZE=45>
    <P><INPUT TYPE=submit>
```

```
</FORM></BODY></HTML>'''

errhtml = '''<HTML><HEAD><TITLE>
Advanced CGI Demo</TITLE></HEAD>
<BODY><H3>ERROR</H3>
<B>%s</B><P>
<FORM><INPUT TYPE=button VALUE=Back
ONCLICK="window.history.back()"></FORM>
</BODY></HTML>'''

reshtml = '''<HTML><HEAD><TITLE>
Advanced CGI Demo</TITLE></HEAD>
<BODY><H2>Your Uploaded Data</H2>
<H3>Your cookie value is: <B>%s</B></H3>
<H3>Your name is: <B>%s</B></H3>
<H3>You can program in the following languages:</H3>
<UL>%s</UL>
<H3>Your uploaded file...<BR>
Name: <I>%s</I><BR>
Contents:</H3>
<PRE>%s</PRE>
Click <A HREF="%s"><B>here</B></A> to return to form.
</BODY></HTML>'''

langSet = ('Python','PERL','Java','C++','PHP','C','JavaScript')
langItem = '<IMPUT TYPE=checkbox NAME=lang VALUE="%s"%s> %s\n'

def getCPPCookies(self):
```

```
        if environ.has_key('HTTP_COOKIE'):
            for eachCookie in map(strip,split(environ['HTTP_COOKIE'],':')):
                if len(eachCookie) > 6 and eachCookie[:3] == 'CPP':
                    tag = eachCookie[3:7]
                    try:
                        self.cookies[tag] = eval(unquote(eachCookie[8:]))
                    except (NameError,SyntaxError):
                        self.cookies[tag] = unquote(eachCookie[8:])
                else:
                    self.cookies['info'] = self.cookies['user'] = ''

            if self.cookies['info'] != '':
                self.who,langSet,self.fn = split(self.cookies['info'],':')
                self.langs = split(langSet,',')
            else:
                self.who = self.fn = ' '
                self.langs = ['Python']

    def showForm(self):
        self.getCPPCookies()
        langStr = ''
        for eachLang in AdvCGI.langSet:
            if eachLang in self.langs:
                langStr +=AdvCGI.langItem % (eachLang, 'CHECKED', eachLang)
            else:
                langStr +=AdvCGI.langItem % (eachLang, '', eachLang)

        if not self.cookies.has_key('user') or self.cookies['user'] == '':
```

```
            cookStatus = '<I>(cookie has not been set yet)</I>'
            userCook = ''
        else:
            userCook = cookSatus = self.cookies['user']

        print AdvCGI.header + AdvCGI.formhtml % (AdvCGI.url,cookStatus, userCook, self.wh
o, langStr, self.fn)

    def showError(self):
        print AdvCGI.header + AdvCGI.errhtml % (self.error)

    def setCPPCookies(self):
        for eachCookie in self.cookies.keys():
             print 'Set-Cookie: CPP%s=%s; path=/' % (eachCookie, quote(self.cookies[eachC
ookie]))

    def doResults(self):
        MAXTYPES = 1024
        langlist=''
        for eachLang in self.langs:
            langlist = langlist + '<LI>%s<BR>' % eachLang

        filedata = ''
        while len(filedata) < MAXTYPES:
            data = self.fp.readline()
            if data == '':break
            filedata += data
        else:
```

```
            filedata += '... <B><I>(file truncated due to size)</I></B>'
            self.fp.close()

        if filedata == '':
            filedata = '<B><I>(file upload error or file not given)</I></B>'

        filename = self.fn

        if not self.cookies.has_key('user') or self.cookies['user']=='':
            cookStatus = '<I>(cookie has not been set yet)</I>'
            userCook = ''
        else:
            userCook = cookStatus = self.cookies['user']

        self.cookies['info'] = join([self.who,join(self.langs,','),filename,':'])
        self.setCPPCookies()
        print AdvCGI.header + AdvCGI.reshtml % (cookStatus, self.who, langlist,filename,
filedata, AdvCGI.url)


    def go(self):
        self.cookies = {}
        self.error = ''
        form = FieldStorage()
        if form.keys() == []:
            self.showForm()
            return
```

```
if form.has_key('person'):
    self.who = capwords(strip(form['person'].value))
    if self.who == '':
        self.error = 'Your name is required. (blank)'
    else:
        self.error = 'Your name is required. (missing)'

    if form.has_key('cookie'):
        self.cookies['user'] =unquote(strip(form['cookies'].value))
    else:
        self.cookies['user'] = ''

    self.langs = []
    if form.has_key('lang'):
        langdata = form['lang']
        if type(langdata) == type([]):
            for eachLang in langdata:
                self.langs.append(eachLang.value)
            else:
                self.langs.append(langdata.value)
        else:
            self.error = 'At least one language required.'

    if form.has_key('upfile'):
        upfile = form['upfile']
        self.fn = upfile.filename or ''
        if upfile.file:
            self.fp = upfile.file
```

```
                else:
                    self.fp = StringIO('(no data)')
            else:
                self.fp = StringIO('(no file)')
                self.fn = ''

            if not self.error:
                self.doResults()
            else:
                self.showError()


if __name__ == "__main__":
    page = AdvCGI()
    page.go()
```

20.8 Web(HTTP)服务器

模块	描述
`BaseHTTPServer`	提供基本的Web服务和处理器类,分别是 `HTTPServer` 和 `BaseHTTPResquestHandler`
`SimpleHTTPServer`	包含执行 `GET` 和 `HEAD` 请求的 `SimpleHTTPRequestHandler` 类
`CGIHTTPServer`	包含处理 `POST` 请求和执行 `CGICGIHTTPRequestHandler` 类

CGI–小结

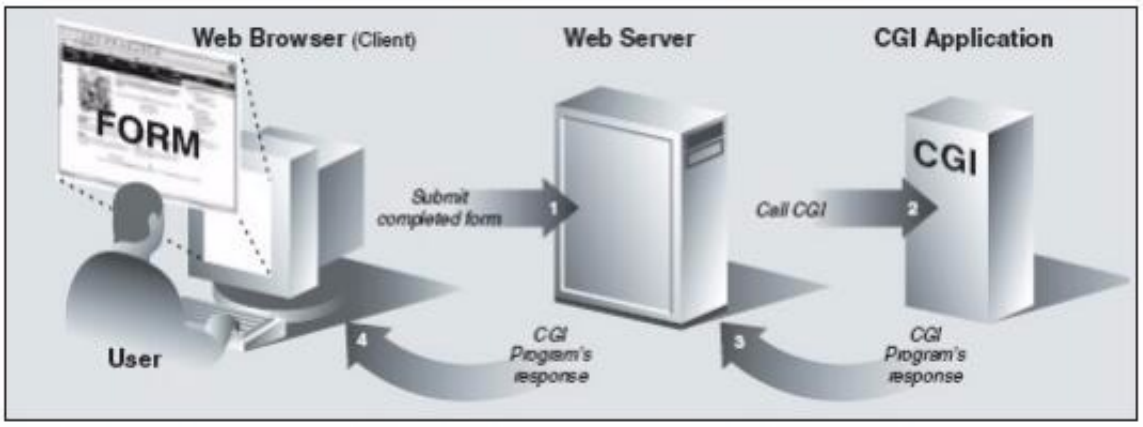


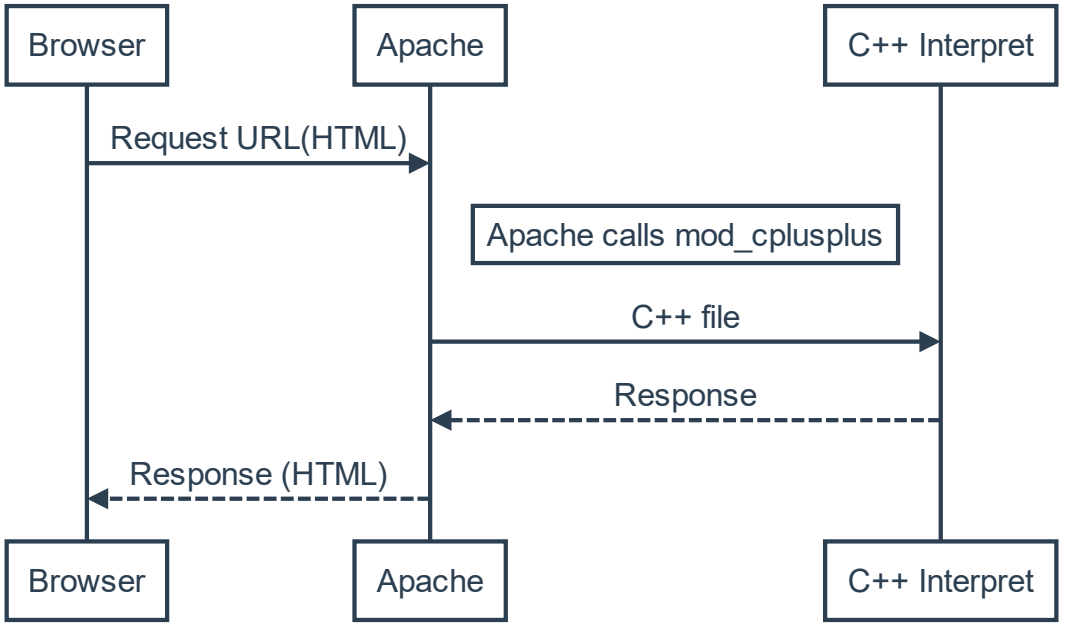
图 20-3 CGI 工作概要图。CGI 代表了在一个 Web 服务器和能够处理用户表单、生成并返回动态 HTML 页的应用程序间的交互。

1. 通过Internet把用户请求送到web服务器。
2. web服务器接收用户请求并交给CGI程序处理。
3. CGI程序把处理结果传送给web服务器。
4. web服务器把结果送回到用户。

CGI的工作方式，从Web服务器的角度看，是在特定的位置(比如：http://www.example.com/wiki.cgi)定义了可以运行CGI程序。当收到一个匹配URL的请求，相应的程序就会被调用，并将客户端发送的数据作为输入。程序的输出会由Web服务器收集，并加上合适的档头，再发送回客户端。

> 编程语言Perl是一个广泛被用来编写CGI程式的语言，但**CGI的一个目的是要独立于任何语言的**。Web服务器无须在这个问题上对语言有任何了解。
> 事实上，CGI程序可以用任何脚本语言或者是完全独立编程语言实现，只要这个语言可以在这个系统上运行。如C++,Python等
> 一般每次的CGI请求都需要新生成一个程序的副本来运行，这样大的工作量会很快将服务器压垮，因此一些更有效的技术像mod_perl，可以让脚本解释器直接作为模块集成在Web服务器（例如：Apache）中，这样就能避免重复载入和初始化解释器。
> 另一个方法是直接把解释器放在Web服务器中，这样就无须新建一个进程来执行脚本。Apache服务器有很多这样的模块，像mod_cplusplus、mod_perl、mod_php、mod_python、mod_ruby、和mod_mono。

关于用Python处理网页的部分还有**WSGI**(用Python语言定义的Web服务器)



`mod_cplusplus` 解释器

Apache中的C++解释器的类图

http://modcplusplus.sourceforge.net/docs/hierarchy.html