Министерство высшего и профессионального образования РФ

Южно Уральский государственный университет

Кафедра «Системного программирования»

Лабораторная работа №2

Веб-сервер на Python

Выполнил:

студенты группы ВМИ-312

Меньшиков А.П. \_\_\_\_\_\_\_\_\_\_\_\_

Проверил:

Громов И.А. \_\_\_\_\_\_\_\_\_\_\_

Челябинск

2013

## Цель работы

Изучить файловые операции в Python

Ознакомиться с принципами тестирования и библиотеками тестирования

## Техническое задание

1. Доработать сервер из предыдущей лабораторной работы так, чтобы он выдавал клиенту содержимое файловой системы. Корень сервера должен задаваться через параметры командной строки или через конфигурационный файл.
2. Дополнительное задание. Написать тест, проверяющий отображение полного списка файлов в папке. Написать тест, проверяющий соответствие выдачи сервера содержимому файловой системы. Написать тест, проверяющий статусы ответов (404 — файл не найден, 403 — ошибка доступа, 200 — ОК)
3. Сохранить результат работы в системе контроля версий.

## Ход работы

Взял за основу код из задания. И немного доработал.

*# response = '''HTTP/1.1 {status}*

*# Connection: Close*

*#*

*# {html}*

*# '''*

*#*

*# context = {}*

*#*

*# context['status'] = 200*

*# context['html'] = '<html></html>'*

*#*

*# response.format(\*\*context)*

Используется для формировании ссылки.

*LISTING = '<a href="{file}">{file}<a/><br>'*

*import SocketServer*

*import re*

*import os*

*from BaseHTTPServer import BaseHTTPRequestHandler, HTTPServer*

*ROOT = 'C:\\'*

Для реализации сервера использую класс-заготовку *BaseHTTPRequestHandler.*

*class HttpProcessor(BaseHTTPRequestHandler):*

*def do\_GET(self):*

*self.send\_response(200)*

*self.send\_header('content-type', 'text/html')*

*self.end\_headers()*

*self.wfile.write("hello !")*

*class MyTCPHandler(SocketServer.BaseRequestHandler):*

*def handle(self):*

*self.data = self.request.recv(1024).strip()*

*path = re.findall('^GET (.\*) HTTP\/1\.1$', self.data.splitlines()[0])[0]*

*target = os.path.join(ROOT, \*path.split('/'))*

*print path.split('/')*

Проверка на наличие директории, если её нет то посылаем в ответ 404.

*if os.path.exists(target):*

*print target*

*if os.path.isdir(target):*

*content = '\n'.join([LISTING.format(file=f) for f in os.listdir(target)])*

*#result = []*

*#for f in os.listdir(target):*

*# result.append.LISTING.format(file=f)*

*else:*

*content = open(target).read()*

*content = """HTTP/1.1 200*

*Content-Type: text/html*

*""" + content*

*#print '----'*

*#print content*

*#print '----'*

*self.request.sendall(content)*

*else:*

*self.request.sendall('404 not found')*

*if \_\_name\_\_ == "\_\_main\_\_":*

*HOST, PORT = "localhost", 8000*

*print 'listening on ', HOST, ':', PORT*

*server = SocketServer.TCPServer((HOST, PORT), MyTCPHandler)*

*# serv = HTTPServer((HOST, PORT), HttpProcessor)*

*server.serve\_forever()*

## Результат

В результате получился простой веб-сервер который выдает содержимое файловой системы.