Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования «Рязанский государственный радиотехнический
университет имени В.Ф. Уткина»
Кафедра ВПМ

Отчёт
о лабораторной работе №2
по дисциплине
«Распределённые системы обработки информации»
Тема:
«Вычисления на основе взаимодействия сервисов»

Выполнила
студентка гр. 343М
Торжкова А.О.
Проверил
доц. Князьков П.А.

Рязань 2023

Цель работы

Получение практических навыков реализация взаимодействия сервисов.

Выполнение

Задание 1. Создать тетрадь «xmlrpc_stats_server.ipynb» с отдельным сервером сервиса статистики. В этот сервис должны приходить события работы функций сервера «xmlrpc_server.ipynb».

Код тетради xmlrpc_stats_server.ipynb (рис. 1-2):

```
from xmlrpc.server import SimpleXMLRPCServer
from xmlrpc.server import SimpleXMLRPCRequestHandler
import datetime
import os


class RequestHandler(SimpleXMLRPCRequestHandler):
    rpc_paths = ('/RPC2',)
server = SimpleXMLRPCServer(("localhost", 8018),
                            requestHandler=RequestHandler)

# Настройка максимального размера лога
max_log_size = 10
current_log_size = None
# Данные файла лога
log_path = '../../resources/lr2/logs/'

# Добавление строки в лог
def add_log(sname):
    global log_path
    global current_log_size
    if current_log_size is None:
        f = open(log_path + 'log.csv','w+')
        current_log_size = len(f.read().split('\n')) - 1
        f.close()
    f = open(log_path + 'log.csv','a')
    f.write(str(sname)+','+ datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S") +'\n')
    f.close()
    current_log_size += 1
    check_log_size()
    return True
server.register_function(add_log, 'add_log')
```

Рисунок 1 – Сервер статистики (начало)

```
# Проверка размера лога, сохранение лога в отдельный файл
def check_log_size():
    global log_path
    global max_log_size
    global current_log_size
    if current_log_size == max_log_size:
        os.rename(log_path + 'log.csv', log_path + 'log_' + datetime.datetime.now().strftime("%Y%m%d_%H%M%S") + '.csv')
        current_log_size = 0

def get_logs_slice(p_type, s_time, e_time):
    global log_path
    f = open(log_path + 'log.csv','r')
    text = f.read().split('\n')
    logs = []
    for line in text:
        if line != '':
            log = line.split(',')
            if (p_type == 'all' or log[0] == p_type) and (s_time == '' or log[1] >= s_time) and (e_time == '' or log[1] <= e_time):
                logs.append((log[0], log[1]))
    f.close()
    return logs
server.register_function(get_logs_slice, 'get_logs_slice')

print ("Listening on port 8018...")
server.serve_forever()
```

Рисунок 2 – Сервер статистики (окончание)

Код тетради xmlrpc_server.ipynb (рис. 3):

```
import xmlrpc.server as server
import xmlrpc.client as client
import datetime
import pandas as pd
import pickle
import numpy


class RequestHandler(server.SimpleXMLRPCRequestHandler):
    rpc_paths = ('/RPC2',)

server = server.SimpleXMLRPCServer(("localhost", 8008),
                           requestHandler=RequestHandler)
server.register_introspection_functions()
stats_server = client.ServerProxy("http://localhost:8018")

# Добавление в лог через сервер
def add_log(log_line):
    try:
        stats_server.add_log(log_line)
        return True
    finally:
        return False

# Получение данных из лога
def get_logs_slice(p_type, s_time, e_time):
    try:
        return stats_server.get_logs_slice(p_type, s_time, e_time)
    except:
        return 'Log server is not available'
server.register_function(get_logs_slice, 'get_logs_slice')

# Тест
def ping():
    add_log('ping')
    return True
server.register_function(ping, 'ping')
```

Рисунок 3 – Сервер, инициирующий события работы функций

Задание 2. Регистрироваться должны время события, тип события сервера (по типу выполняемой операции). Сохранять журнал событий в файл формата CSV. При отсутствии работы (выключенном) сервере «xmlrpc_stats_server.ipynb» работа функций сервера «xmlrpc_server.ipynb» должна выполняться.

Регистрация типа и времени события происходит в функции add_log() сервера статистики (рис. 4):

```
# Добавление строки в лог
def add_log(sname):
    global log_path
    global current_log_size
    if current_log_size is None:
        f = open(log_path + 'log.csv','w+')
        current_log_size = len(f.read().split('\n')) - 1
        f.close()
    f = open(log_path + 'log.csv','a')
    f.write(str(sname)+','+ datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S") +'\n')
    f.close()
    current_log_size += 1
    check_log_size()
    return True
server.register_function(add_log, 'add_log')
```

Рисунок 4 – Регистрация событий в сервере статистики

Корректная работа сервера при выключенном сервере статистики обеспечивается за счёт конструкции try-catch в функции вызова события в add_log() сервера (рис. 5):

```
# Добавление в лог через сервер
def add_log(log_line):
    try:
        stats_server.add_log(log_line)
        return True
    finally:
        return False
```

Рисунок 5 – Вызов события на сервере

Задание 3. Сервис статистики должен иметь настройку ограничения на максимальный размер журнала событий в количестве записей. При превышении размера сохранять существующий журнал событий под новым именем, содержащим дату сохранения в формате YYYMMDD_hhmmss. После этого создавать новый файл журнала событий и вести запись в него.

Настройка максимального размера лога обеспечивается переменными max_log_size и current_log_size. При поступлении первого события в сервер статистики с момента начала его работы в переменную current_log_size записывается текущий размер лога, который остался с прошлого запуска сервера. При каждой записи события в лог переменная инкрементируется и вызывается функция check_log_size(), которая при максимальном размере лога сохраняет его в отдельный файл и сбрасывает текущий размер (рис. 6):

```
# Настройка максимального размера лога
max_log_size = 10
current_log_size = None
# Данные файла лога
log_path = '../../resources/lr2/logs/'

# Добавление строки в лог
def add_log(sname):
    global log_path
    global current_log_size
    if current_log_size is None:
        f = open(log_path + 'log.csv','w+')
        current_log_size = len(f.read().split('\n')) - 1
        f.close()
    f = open(log_path + 'log.csv','a')
    f.write(str(sname)+','+ datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S") +'\n')
    f.close()
    current_log_size += 1
    check_log_size()
    return True
server.register_function(add_log, 'add_log')

# Проверка размера лога, сохранение лога в отдельный файл
def check_log_size():
    global log_path
    global max_log_size
    global current_log_size
    if current_log_size == max_log_size:
        os.rename(log_path + 'log.csv', log_path + 'log_' + datetime.datetime.now().strftime("%Y%m%d_%H%M%S") + '.csv')
        current_log_size = 0
```

Рисунок 6 – Ограничение размера лога в сервере статистики

Задание 4. В рамках клиента «xmlrpc_client.ipynb» реализовать получение содержимого журнала событий сервера статистики с возможностью получения среза по типу выполняемой операции (события) и времени.

Код тетради xmlrpc_stats_server.ipynb для получения среза лога (рис. 7):

```
def get_logs_slice(p_type, s_time, e_time):
    global log_path
    f = open(log_path + 'log.csv','r')
    text = f.read().split('\n')
    logs = []
    for line in text:
        if line != '':
            log = line.split(',')
            if (p_type == 'all' or log[0] == p_type) and (s_time == '' or log[1] >= s_time) and (e_time == '' or log[1] <= e_time):
                logs.append((log[0], log[1]))
    f.close()
    return logs
server.register_function(get_logs_slice, 'get_logs_slice')
```

Рисунок 7 – Функция получения среза лога в сервере статистики

Код тетради xmlrpc_client.ipynb и вывод результатов получения среза лога (рис. 8):

```
# Лабораторная 2. Задание 4. Получение содержимого журнала событий сервера статистики
# с возможностью получения среза по типу выполняемой операции (события) и времени

import datetime

print('All logs:\n' + '\n'.join(map(str,server.get_logs_slice('all', '', ''))))
print('\nSliced logs:\n' + '\n'.join(map(str,server.get_logs_slice('type', '2023-09-26 19:18:33', '2023-09-26 22:27:11'))))
```

```
All logs:
['ping', '2023-09-26 19:18:12']
['now', '2023-09-26 19:18:16']
['type', '2023-09-26 19:18:20']
['type', '2023-09-26 19:18:24']
['type', '2023-09-26 19:18:28']
['type', '2023-09-26 19:18:33']
['type', '2023-09-26 19:18:37']
['type', '2023-09-26 19:18:41']
['type', '2023-09-26 19:18:45']
['sum', '2023-09-26 19:18:49']

Sliced logs:
['type', '2023-09-26 19:18:33']
['type', '2023-09-26 19:18:37']
['type', '2023-09-26 19:18:41']
['type', '2023-09-26 19:18:45']
```

Рисунок 8 – Получение срезов лога на клиенте

Задание 5. Оформить отчет по результатам выполнения лабораторной работы.

Описание полученных навыков

В ходе лабораторной работы были получены практические навыки создания и настройки взаимодействия двух сервисов.

Вывод

Взаимодействие сервисов позволяет разделить различные по типу задачи между разными сервисами таким образом, чтобы при сбое работы одного сервиса другом мог корректно продолжать свою работу.