Министерство Образования, Культуры и Исследований
Технический Университет Молдовы Департамент Программной Инженерии и Автоматики

ОТЧЕТ

Лабораторная работа №3

По предмету: **Программирование в сети**

Тема: HTTP Client.

Выполнил студент гр.SI-212: Решетников Максим

Проверил: Лях Аркадий

Кишинев 2024 г.

Ссылка на GitHub

Визуальная часть

```python
class HTTPClientApp:
    def __init__(self, root):
        # Интерфейс для ввода URL, метода запроса и вывода результатов
        self.url_label = ttk.Label(root, text="URL:")
        self.url_entry = ttk.Entry(root, width=40)
        self.method_label = ttk.Label(root, text="Method:")
        self.method_combobox = ttk.Combobox(root, values=["GET", "POST", "HEAD", "OPTIONS"])
        self.result_text = tk.Text(root, height=10, width=50, state=tk.DISABLED)
        self.send_button = ttk.Button(root, text="Send Request", command=self.send_request)

        # Добавляем интерфейс для ввода IP и порта прокси
        self.proxy_ip_label = ttk.Label(root, text="Proxy IP:")
        self.proxy_ip_entry = ttk.Entry(root, width=15)
        self.proxy_port_label = ttk.Label(root, text="Proxy Port:")
        self.proxy_port_entry = ttk.Entry(root, width=5)

        # Размещаем элементы для прокси на форме
        self.proxy_ip_label.grid(row=4, column=0, padx=5, pady=5, sticky=tk.E)
        self.proxy_ip_entry.grid(row=4, column=1, padx=5, pady=5, sticky=tk.W)
        self.proxy_port_label.grid(row=4, column=2, padx=5, pady=5, sticky=tk.E)
        self.proxy_port_entry.grid(row=4, column=3, padx=5, pady=5, sticky=tk.W)

        # Размещение элементов на форме
        self.url_label.grid(row=0, column=0, padx=5, pady=5, sticky=tk.E)
        self.url_entry.grid(row=0, column=1, padx=5, pady=5, columnspan=2, sticky=tk.W)
        self.method_label.grid(row=1, column=0, padx=5, pady=5, sticky=tk.E)
        self.method_combobox.grid(row=1, column=1, padx=5, pady=5, sticky=tk.W)
        self.result_text.grid(row=2, column=0, columnspan=3, padx=5, pady=5)
        self.send_button.grid(row=3, column=0, columnspan=3, pady=5)
```

Отсылка запроса и получение результат

```
def make_request(self, url, method, proxies):
    try:
        # Выполняем HTTP-запрос с прокси
        response = requests.request(method, url, proxies=proxies)

        # Отображаем результат в текстовом поле
        result_text = f"Response Code: {response.status_code}\n\n{response.text}"
        self.display_result(result_text)
    except Exception as e:
        self.display_result(f"An error occurred: {str(e)}")

def display_result(self, text):
    # Выводим результат в текстовое поле
    self.result_text.config(state=tk.NORMAL)
    self.result_text.delete(1.0, tk.END)
    self.result_text.insert(tk.END, text)
    self.result_text.config(state=tk.DISABLED)
```

Прокси

```
def send_request(self):
    # Получаем данные из интерфейса
    url = self.url_entry.get()
    method = self.method_combobox.get()
    proxy_ip = self.proxy_ip_entry.get()
    proxy_port = self.proxy_port_entry.get()

    # Проверка валидности URL и метода
    if not url or not method:
        self.display_result("Invalid URL or method")
        return

    # Подготавливаем параметр proxies для запроса
    proxies = {}
    if proxy_ip and proxy_port:
        proxy_url = f"http://{proxy_ip}:{proxy_port}"
        proxies = {"http": proxy_url, "https": proxy_url}
```

Листинг

```
import tkinter as tk
from tkinter import ttk
import threading
import requests

class HTTPClientApp:
    def __init__(self, root):
        self.root = root
        self.root.title("HTTP Client App")

        # Интерфейс для ввода URL, метода запроса и вывода результатов
        self.url_label = ttk.Label(root, text="URL:")
        self.url_entry = ttk.Entry(root, width=40)
        self.method_label = ttk.Label(root, text="Method:")
        self.method_combobox = ttk.Combobox(root, values=["GET", "POST", "HEAD",
"OPTIONS"])
        self.result_text = tk.Text(root, height=10, width=50, state=tk.DISABLED)
        self.send_button = ttk.Button(root, text="Send Request",
command=self.send_request)

        # Добавляем интерфейс для ввода IP и порта прокси
        self.proxy_ip_label = ttk.Label(root, text="Proxy IP:")
        self.proxy_ip_entry = ttk.Entry(root, width=15)
        self.proxy_port_label = ttk.Label(root, text="Proxy Port:")
        self.proxy_port_entry = ttk.Entry(root, width=5)

        # Размещаем элементы для прокси на форме
        self.proxy_ip_label.grid(row=4, column=0, padx=5, pady=5, sticky=tk.E)
        self.proxy_ip_entry.grid(row=4, column=1, padx=5, pady=5, sticky=tk.W)
        self.proxy_port_label.grid(row=4, column=2, padx=5, pady=5, sticky=tk.E)
        self.proxy_port_entry.grid(row=4, column=3, padx=5, pady=5, sticky=tk.W)

        # Размещение элементов на форме
        self.url_label.grid(row=0, column=0, padx=5, pady=5, sticky=tk.E)
        self.url_entry.grid(row=0, column=1, padx=5, pady=5, columnspan=2,
sticky=tk.W)
        self.method_label.grid(row=1, column=0, padx=5, pady=5, sticky=tk.E)
        self.method_combobox.grid(row=1, column=1, padx=5, pady=5, sticky=tk.W)
        self.result_text.grid(row=2, column=0, columnspan=3, padx=5, pady=5)
        self.send_button.grid(row=3, column=0, columnspan=3, pady=5)

    def send_request(self):
        # Получаем данные из интерфейса
        url = self.url_entry.get()
        method = self.method_combobox.get()
        proxy_ip = self.proxy_ip_entry.get()
        proxy_port = self.proxy_port_entry.get()
```

```
        # Проверка валидности URL и метода
        if not url or not method:
            self.display_result("Invalid URL or method")
            return

        # Подготавливаем параметр proxies для запроса
        proxies = {}
        if proxy_ip and proxy_port:
            proxy_url = f"http://{proxy_ip}:{proxy_port}"
            proxies = {"http": proxy_url, "https": proxy_url}

        # Выполняем запрос в отдельном потоке с использованием прокси
        threading.Thread(target=self.make_request, args=(url, method, proxies),
daemon=True).start()

    def make_request(self, url, method, proxies):
        try:
            # Выполняем HTTP-запрос с прокси
            response = requests.request(method, url, proxies=proxies)

            # Отображаем результат в текстовом поле
            result_text = f"Response Code:
{response.status_code}\n\n{response.text}"
            self.display_result(result_text)
        except Exception as e:
            self.display_result(f"An error occurred: {str(e)}")

    def display_result(self, text):
        # Выводим результат в текстовое поле
        self.result_text.config(state=tk.NORMAL)
        self.result_text.delete(1.0, tk.END)
        self.result_text.insert(tk.END, text)
        self.result_text.config(state=tk.DISABLED)

if __name__ == "__main__":
    root = tk.Tk()
    app = HTTPClientApp(root)
    root.mainloop()
```

Вывод: В ходе работы мы научились создавать HTTPS-клиент, подключать прокси и выполнять запросы GET,POST, HEAD, OPTIONS.