Simple Browser

استاد : آقای دکتر بستام

دانشجو : عمید اسدالهی مجد

شماره دانشجویی : ۹۴۱۲۲۱۱۰۲

دانشگاه مازندران

تابستان ۹۹

چکیده

در این پروژه سعی بر پیاده‌سازی یک مرورگر از صفر بود که یک مرورگر ساده با استفاده از سوکت ها (Sockets) و کتابخوانه PyQt5 پیاده‌سازی شد. درواقع با استفاده از سوکت ها پیامی بین دو هاست (کلاینت و سرور) ارسال و دریافت می‌شود و در محیط گرافیکی مرورگر پاسخ سرور به درخواست HTTP کلاینت نمایش داده می‌شود.
این مرورگر از گونه های مختلف آدرس و همچنین HTTPS (با تایپ پروتکل https:// قبل از آدرس) پشتیبانی می‌کند.

نصب و راه اندازی

برای راه اندازی پروژه تنها نیاز به نصب دو کتابخوانه زیر می‌باشد:

۱. `magic` : برای تشخیص فرمت فایل با استفاده از magic number (مثل jpg, text, html, ...)

۲. `pyqt5` : برای اجرای محیط گرافیکی

دسته بندی فایل‌های موجود در پروژه

پروژه شامل یک دو فایل اصلی limited_http_server و limited_http_client می‌باشد که به ترتیب سرور ساده و کلاینت پس زمینه مرورگر هستند. همچنین فایلی که در نهایت باید اجرا گردد تا مرورگر همراه با محیط گرافیکی اجرا شود main.py نام دارد.

ماژول `main`

برای اجرای مرورگر به صورت گرافیکی باید این فایل با دستور python main.py اجرا شود. این فایل شامل یک کلاس MainWindow است که در این کلاس یک آبجکت از محیط گرافیکی طراحی شده ساخته می‌شود و سپس با کلیک کاربر بر روی GO یا فشردن کلید Enter پس از تایپ آدرس مورد نظر، ماژول limited_http_client در پس زمینه اجرا شده و آدرس کاربر به این ماژول داده می‌شود تا قسمت محتوای نتیجه درخواست (پاسخ سرور) را در محیط مرورگر نمایش دهد. اگر آدرس وارد شده بسیار کوچک باشد (کمتر از سه حرف) یا مشکلی در دریافت پاسخ وجود داشته باشد، به ترتیب پیغام های Invalid Address و Requested Address is not Reachable روی مرورگر مشاهده خواهند شد.

ماژول limited_http_client

این ماژول درواقع قسمت اصلی مرورگر است که آدرس درج شده در محیط گرافیکی را به عنوان درخواست ارسال کرده و پاسخ درخواست را به محیط گرافیکی بر می‌گرداند.
این ماژول با صدا زدن متود send_request که آدرس درخواست ورودی این تابع است.

متود tokenize_address

این متود URL وارد شده کاربر را دریافت کرده و دیکشنری address را بر می‌گرداند که شامل host به عنوان آدرس پایه سرور، port به عنوان پورت درخواستی کاربر، path به عنوان مسیر درخواستی کاربر و پروتکل می باشد.
کاربر امکان وارد کردن آدرس به گونه های مختلف را دارد. مثلاً همراه با http:// یا بدون آن، با شماره پورت یا بدون آن و ..
همچنین این متود اگر پورتی وارد نشده باشد پورت ۸۰ را برای پروتکل HTTP و پورت ۴۴۳ را برای پروتکل HTTPS در نظر می‌گیرد که برای درخواست https باید https:// قبل از آدرس تایپ شود.

متود tokenize_response

این متود پاسخ دریافت شده از سمت سرور را جدا سازی کرده و در نهایت یک دیکشنری شامل کلید هایی مثل status_code – status – protocol – content و ... برمی‌گرداند.

متود response_decode

این متود وظیفه دیکود کردن اطلاعات با فرمت UTF-8 را بر عهده دارد.

متود create_get_request

این متود بدنه درخواست را با مقادیر ساخته شده می‌سازد و به باینری دیکود می‌کند.

متود send_request

این متود درواقع متود اصلی کلاس Client است که مراحل زیر را برای هر آدرس ورودی اجرا می‌کند:

1. ساختن یک سوکت با پروتکل TCP و IP از ورژن ۴
2. اجرای جدا ساز آدرس بر روی ورودی کاربر (tokenize_address)
3. با توجه به پروتکل معین شده در مرحله دوم، ایجاد اتصال (ساختن connection) به آدرس درخواستی کاربر با پروتکل HTTP به صورت پیش‌فرض و یا ایجاد اتصال امن با پروتکل HTTPS با استفاده از سوکت ساخته شده در مرحله اول (به ازای هر آدرس ورودی)
4. ساختن بدنه درخواست HTTP/HTTPS با اجرای متود (create_get_request) و ارسال این درخواست با استفاده از سوکت
5. در حلقه while بی نهایت تا واقتی که سرور داده‌ای برای ارسال داشته باشد و اندازه تکه (chunk) دریافتی از مقادیر دریافتی قبلی کمتر نباشد، به دریافت از طریق بافر ادامه می‌دهد.
   اما وقتی نشانه رسمی پایان فایل در chunk یافت شود به این حلقه خاتمه داده می‌شود و عمل دریافت پایان می‌یابد. همچنین اگر میزان timeout درنظر گرفته شده سپری شود و دریافتی وجود نداشته باشد و سایز chunk نیز کوچکتر از مقادیر دریافتی قبلی باشد، حلقه while شکسته می‌شود.
6. تکه‌های دریافتی chunk به هم چسبانده شده و با اجرای متود tokenize_response جداسازی می‌شود.
7. عملیات دیکود روی پاسخ دریافتی اجرا می‌شود تا پاسخ از فرمت باینری به رشته تبدیل شود.
8. پاسخ دریافتی به عنوان نتیجه متود send_request برگردانده می‌شود.