Sieci Komputerowe II, laboratoria, semestr V, grupa C
Mateusz Bednarski 117194
Artur Konieczny 119519

Prosty serwer protokołu HTTP zgodny ze specyfikacją RFC 2616 co najmniej w zakresie żądań: GET, HEAD, PUT, DELETE

Opis protokołu

http jest protokołem bezstanowym typu request-reply. Każda wiadomość zawiera linię z wersją, nazwą żądanego zasobu oraz metodą do wykonania, zakończona CRLF. Potem w kolejnych liniach znajdują się nagłówki w formacie „nazwa: wartość<CRLF>”. Koniec nagłówków sygnalizuje pusta linia. Po niej następuje opcjonalny payload.
Kompilacja i uruchomienie:

Serwer:

```
./build.sh
cd bin
./skserver [-p port] [-ps rozmiar_puli_wątków]
```

Klient:
Projekt buduje się z poziomu Visual Studio

Krótki opis plików źródłowych:

Serwer:

ContentStream – własna implementacja strumienia bajtów zgodna z RAII
HttpException – własne klasy wyjątków
HttpRequest – reprezentacja żądania http
HttpResponse – reprezentacja oraz tworzenie odpowiedzi HTTP
HttpServerTask – zadanie obsługi klienta, skleja całość oraz parsuje żądanie
HttpUtil – pomniejsze helpery
Logger – mechanizm loggera
main – inicjalizacja, rozpoczęcie nasłuchu i spawnowanie obsługi klientów
MtLoader – wczytywanie skojarzeń MIME
Settings – parsowanie argumentów wywołania
ThreadPool – mechanizm puli wątków
Util – pomniejsze funkcje niezwiązane bezpośrednio z protokołem HTTP

Klient:

(pominięto szczegóły implementacyjne)
Model/SkClient.cs – klasa wysyłająca żądanie i przetwarzająca odpowiedź
Model/SkRequest.cs – model żądania http
Model/SkResponse – model odpowiedzi http

ViewModel/(Head/Get/Put/Delete)ViewModel.cs – bindowanie z interfejsem użytownika,
View/(Head/Get/Put/Delete)View.xaml – interfejs użytkownika

Opis implementacji

Serwer:

Tworzona jest pula wątków, każde nadchodzące połączenie spawnuje obiekt HttpServerTask zajmujący się jego obsługą. Odczytywane jest MAX_HEADER_SIZE bajtów z żądania, jeśli nie znajdzie się tam nagłówek Content-Length zwracany jest błąd 411 Length Required. W przeciwnym wypadku odczytywane jest content-length bajtów. Dla get sprawdzanie jest czy zasób istnieje na serwerze (obsługa domyślnego index.html) (404 Not Found jeśli nie istnieje) jest on wczytywany do pamięci i odsyłany (200 OK) wraz z nagłówkami Content-Length oraz Content-Type (dla znanych rozszerzeń, domyślnie application/octet-stream). Dla HEAD to samo tylko, że nie jest wysyłana treść. Dla DELETE następuje próba usunięcia zasobu (204 No Content w przypadku sukcesu). 404 Not Found jeśli nie znaleziono, lub 500 Internal Server Error w przeciwnym przypadku (np. brak uprawnień lub jakikolwiek inny błąd). Dla PUT następuje próba zapisania payloadu na dysk. Jeśli się nie uda 400 Bad Request (szerszy komentarz dlaczego tak w pliku źródłowym). Jeśli zasób nie istniał 201 Created, jeśli już istniał i został nadpisany – 204 No Content. Tak czy inaczej serwer za każdym razem dodaje nagłówki: Connection: Close, Server: SKHTTP, Cache-control: no-cache.

Klient

Dla przygotowanego żądania jest tworzony socket, po czym następuje jego wysłanie(żądania). Odpowiedź jest zapisywane do strumienia (na dysku lub do pliku) co pozwala obsłużyć bardzo duże odpowiedzi. W zależności czy typ MIME jest określony jako nadający się do wyświetlenia, odpowiedź jest wyświetlana w oknie lub zapisywana na dysk. Dla metod HEAD/PUT/DELETE wyświetlane są nagłówki. Dla każdej Status Code i Reason Phase. Budowa żądań wygląda wszędzie bardzo podobnie. Każde zawiera nagłówki: Cache-control: no-cache, Connection: close, User-Agent: SK HTTP Client. PUT dodatkowo dodaje content-type oraz Content-Length.