HTTP nástenka

Oliver Chmelický

xchmel28

Brno 4.11.2019

OBSAH:

Technologický úvod

HTTP protokol je spojovaná služba, aplikačnej vrstvy v počítačovej komunikácii. Z toho vyplíva, že pracuje nad protokolom, ktorý zabezpečuje na transportnej vrstve spoľahlivú komunikáciu. Takáto komunikácie je o čosi obsiahlejšia a koncové zariadenia majú zvýšenú výpočetnú réžiu. Vo výsledku máme pomalšiu komunikáciu, no vďaka stále zlepšujúcim sa technológiám je tento fakt pre bežného uživateľa nepodstatný.

Z programátorského hľadiska sa programátor nemusí zaoberať vytváraním handshakov a potvrdzovaním správ no jediné, čo musí špecifikovať je port na ktorom aplikácia pracuje. Zo sieťovej vrstvy je potrebné určiť, ip adresu kam sa dáta posielajú, prípadne na strane servera na ktorom rozhraní načúva. Zvyšok programovania sa už venuje len aplikačnej vrstve. HTTP potokol je ľudsky čitateľný protokol (získané dáta sú dekódované do čitateľnej podoby).

Klient vytvára HTTP Request a server na tento request odpovedá. Zloženie requestu má tri hlavné časti. Je to request line, headder lines, kde sa špecifikujú informácie requeste, spojení, prípadne o klientovi a samotné telo správy, ktoré je nepovinné a závisí od metódy.

Server spracuje takýto request a následne odpovie správou HTTP Response kde klient nájde status line, headder lines a samotné telo odpovede.

Návod na použitie

Preklad serveru a clienta v domvskom adresári projektu:

```
make
```

Pred prekladom odporúčam použiť príkaz

```
make clean
```

Štart serveru

```
./isaserver -p <port>
```

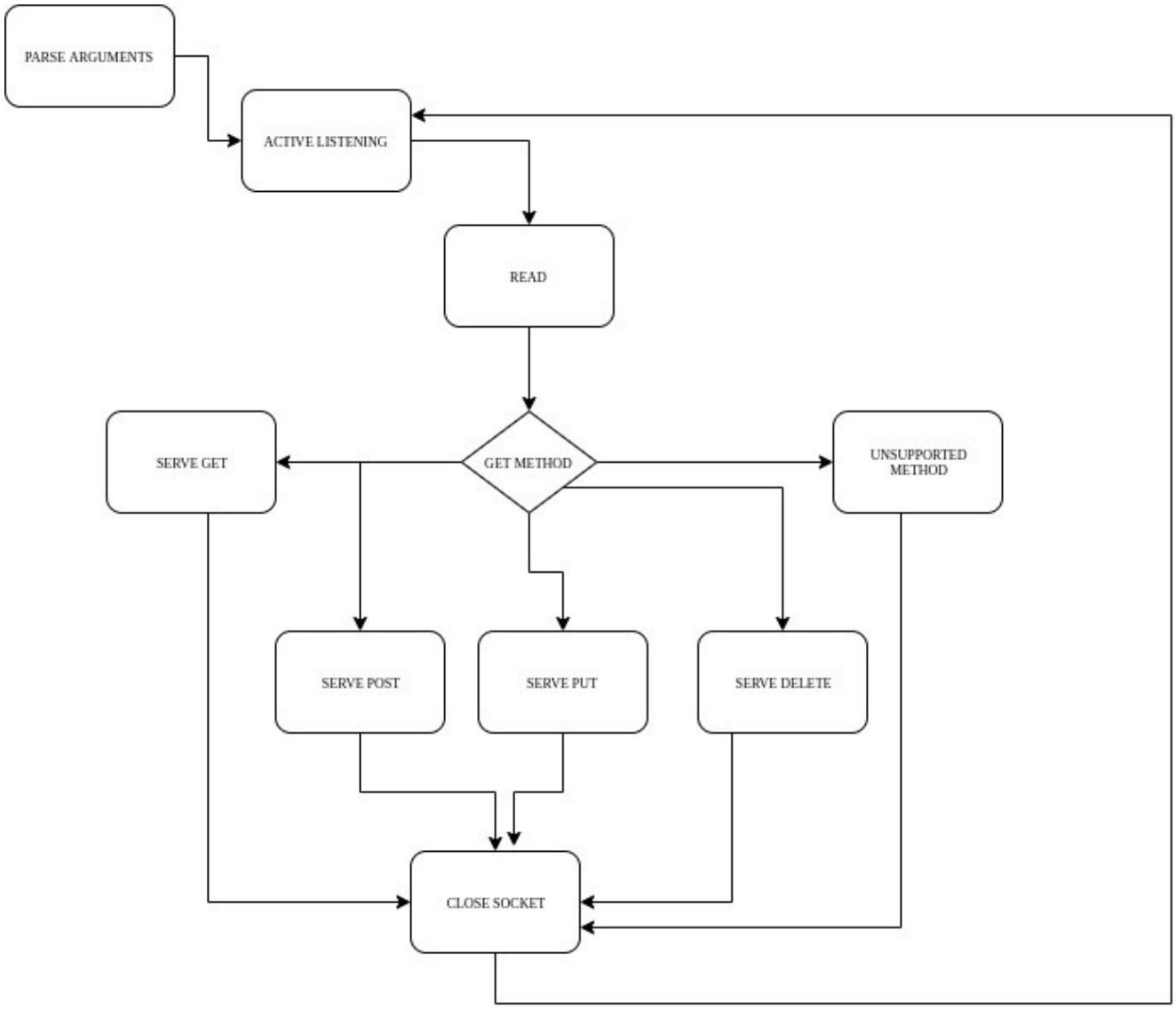
Štart clienta

```
./isaclient -H <host> -p <port> <command>
```

Client po získaní odpovede vypíše status code, status message a body odpovede.

Server

Server prijíma ako argument parameter -p <port number> číslo na ktoré sa binduje server. Server naslúcha na všetkých rozhraniach na danom porte.



Základné chovanie serveru na pri obdržaní dotazu

Server aktívne čaká na zahájenie spojenia od clienta z čoho vzíde nový socket na novom porte pridelenom operačným systémom.

V requeste si server najprv skontroluje, či klient komunikuje protokolom HTTP 1.1, následne zistí metódu a potom zavolá metódu

serve(const string& request, int socket, map<string, vector<string> >& db) príslušnej triedy. Ak sa jedná o metódu, ktorej server nerozumie, tak vracia error 404 Bad Request.

GET

Get rozparsuje URI a zistí akým spôsobom zareagovať. Túto vlastnosť majú všetky handlery podporovaných metód, tak v ďalších metódach to už spomínané nebude.

V prípade aj sa jedná o boards, tak trieda získa volaním databázovej vrstvy všetky existujúce nástenky a zašle ich späť klientovi.
Ak sa jedná o board, tak v závislosti od názvu sa buď vráti obsah nástenky, alebo chybná odpoveď 404 Not Found

POST

U tejto metódy je potrebné skontrolovať hlavičky v requeste. Kontrolujem Content-Type, kde podporovaným formátom je text/plain a Content-Length pre získanie dĺžky tela správy. Body **musí** byť dlhšie ako 0.

Ak je všetko v poriadku, tak podľa príslušnej URI reagujem na boards tak, že regulárnym výrazom skontrolujem, či názov, ktorý sa nachádza v body requestu spĺňa podmienky názvu `[a-zA-Z0-9]` nástenky. Následne skontrolujem, či táto nástenka existuje. Ak áno vraciam error 409 Conflict inak vraciam 201 Created.

V prípade boards/[name] pozerám rovnako hlavičky ako v predchádzajúcom prípade a skontrolujem, či nástenka existuje. Ak áno vytváram nový záznam do nástenky, Ak nie vraciam odpoveď 404 Not Found.

PUT

PUT je posledná z dvoch metód, ktorá **musí** mať body v requeste na server. Server zistí, či existuje nástenka, nad ktorou klient chce robiť update a následne si server skontroluje, či id zadanej nástenky je v rozmädzí, v ktorom existujú nejaké príspevky. V prípade neúspechu oboch kontrol, sa vracia 404 Not Found. V prípade úspechu 200 OK.

DELETE

Delete je opäť jednoduchšia metóda, pretože ak sa nachádza body v requeste, tak je jednoducho odignorované.

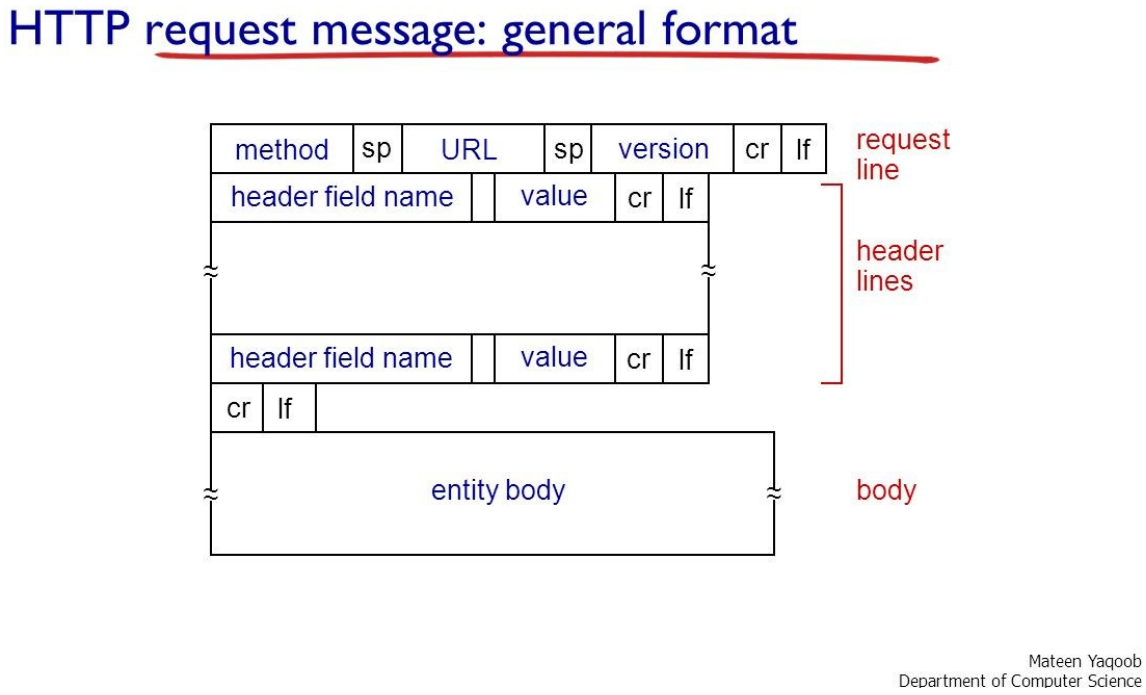
Ak sa jedná o delete nástenky, tak sa server pozrie či nástenka existuje. Ak áno, tak ju vymaže, ak nie vracia sa 404 Not Found.

Pri delete príspevku z nástenky sa skontroluje existencia nástenky a následne existencia id. Ak áno, tak sa príspevok vymaže a vracia sa 200 OK, v opačnom prípade 404 Not Found.

Klient

Implementácia HTTP komunikácie v tejto časti projektu je jednoduchšia ako na serverovej časti.

Po rozparsovaní argumentov sa vytvorí HTTP request, kde sa vyplní hlavička Content-Type a Content-Length.



Na konci klient čaká na odpoveď. Z odpovede vypíše Status code a telo správy.

Klient vracia 0 ak všetko prebehlo v poriadku a získal akúkoľvek odpoveď od serveru. Inú ako nulovú hodnotu vracia ak sa nepodarí vyparsovať argumenty, naviazať spojenie, alebo prečítať telo.

Pred dokončením behu programu sa zavolá systémové volanie close() na socket a program sa ukončí.

Obmädzenia a vlastnosti serveru

Server je schopný prijímať requesty s maximálnou dĺžkou BUFSIZ.

Server nepodporuje Transfer-Encoding: chunked.

Server po obslúžení socket uzavrie (Connection : keep-alive nie je podporované).

Metódy POST a PUT vyžadujú Content-Length väčšie ako 0, inak server vráti error.

ZDROJE :

https://docs.citrix.com/en-us/netscaler/12/appexpert/http-callout/http-request-response-notes-format.html

https://medium.com/from-the-scratch/http-server-what-do-you-need-to-know-to-build-a-simple-http-server-from-scratch-d1ef8945e4fa