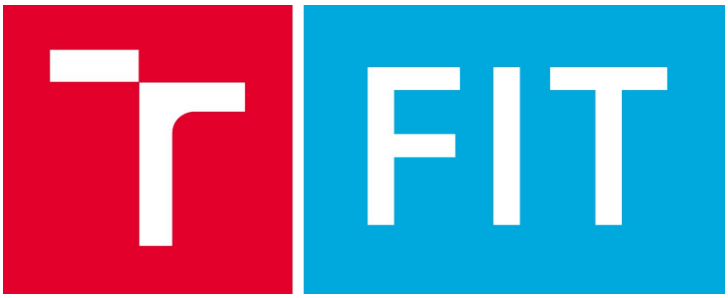


Vysoké učení technické v Brně

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Síťové aplikace a správa sítí

2019/2020

HTTP nástenka

Adam Abrahám (xabrah04)

18.11.2019

Obsah

1. Úvod....2
2. Client-Server komunikácia....2
2.1. Protokol HTTP....2
2.2 HTTP/1.1....3
2.3 Druhy žiadostí HTTP....3
2.4 Príklad HTTP komunikácie....3
3. Implementácia....4
3.1. Riešenie Client-Server komunikácie....4
3.2. Implementácia *isaclient* aplikácie....5
3.3. Implementácia *isaserver* aplikácie....6
3.4. Návod na použitie....6
4. Záver....7
5. Použitá literatúra....7

1. Úvod

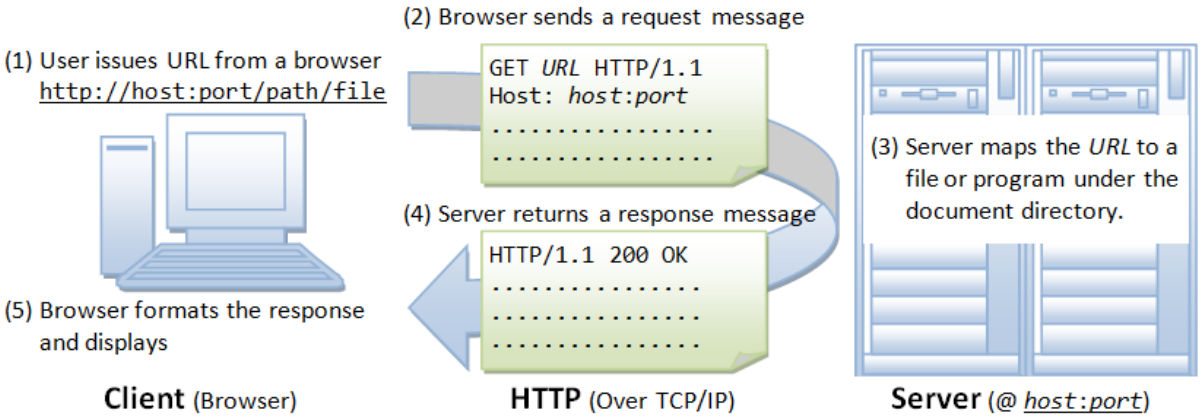
Aplikácia umožňuje klientom spravovať nástenky na serveri pomocou HTTP API. API im umožňuje prezerať, pridávať, upravovať a mazať príspevky na nástenkách ako aj nástenky samotné.

Nástenkou sa rozumie usporiadaný zoznam textových príspevkov. Každý príspevok ma identifikačné číslo a textový obsah.

2. Client-Server komunikácia

2.1. Protokol HTTP

HTTP je protokol definujúci požiadavky a odpovede medzi klientmi a servermi. HTTP klient zvyčajne začne požiadavku nadviazaním TCP spojenia na určenom porte vzdialeného stroja, štandardne na porte 80.
HTTP server počúvajúci na danom porte čaká, kým klient pošle reťazec s požiadavkou ako napr. *"GET / HTTP/1.1"* nasledovaný sériou hlavičiek. Niektoré hlavičky sú nepovinné, zatiaľ čo verzia HTTP/1.1 niektoré vyžaduje, ako názov stroja. V našom prípade sa používajú hlavičky Content-Length a Content-Type, ale aplikácia sa vysporiada aj s neznámymi hlavičkami. Po hlavičkách môže nasledovať aj teleso s ľubovoľnými údajmi. Po prijatí požiadavky server pošle reťazec s odpoveďou ako napr. *"200 OK"* nasledovanou hlavičkami spolu so samotnou správou, ktorej telo tvorí obsah požadovaného údaja, chybové hlásenie alebo iná informácia.



Obrázok 1: Proces http komunikácie
Zdroj: https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html

2.2 HTTP/1.1

V aplikáciách je použitá verzia HTTP/1.1. Verzia 1.1 je definovaná v RFC 2616 z roku 1999, dnes je najčastejšie používaná. Dopĺňa možnosti HTTP/1.0 ako napr. hierarchické proxy, cashovanie, trvalé spojenia, virtuálne servery a je spätne kompatibilná. Má takmer dvojnásobný počet hlavičiek oproti verzii 1.0.

Protokol 1.1 vytvára tzv. perzistentné spojenie a preto servery podporujúce verziu 1.1 spojenie hneď neuzavrú, ale chvíľu čakajú na ďalšie príkazy. Klient tak môže pokračovať v otázkach na ostatné prvky HTML stánky a spojenie ukončiť sám.

2.3 Druhy žiadostí HTTP

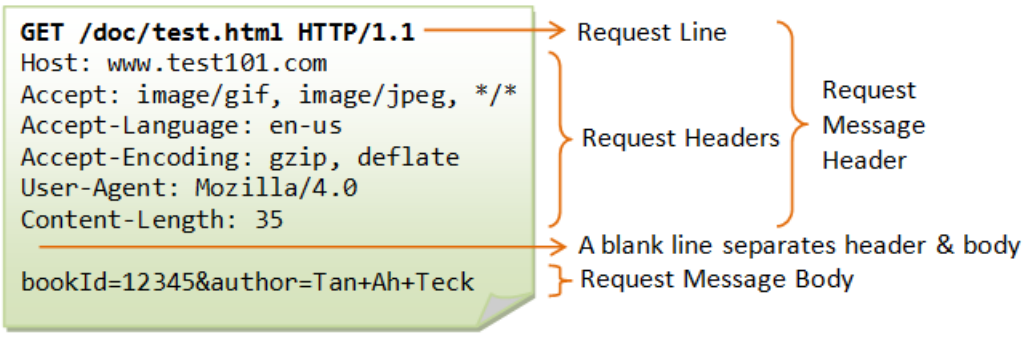
Zvyčajne sa nazývajú metódy, ako *GET, POST, PUT, DELETE, HEAD, TRACE, OPTIONS* a *CONNECT.*

V aplikácii sa využívajú nasledovné:

- **GET** - Žiada o zdroj uvedením jeho URL
- **POST** - Podobne ako GET, okrem toho, že je pridané telo správy zvyčajne obsahujúce dvojice kľúč-hodnota z HTML formulára a taktiež na upload súborov.
- **PUT** - Používa sa na úpravu, resp. editáciu údajov špecifikovaného objektu, podobne ako POST, pričom POST vytvára nový objekt a PUT upravuje tieto dáta.
- **DELETE** – Používa sa na zmazanie údajov.

2.4 Príklad HTTP komunikácie

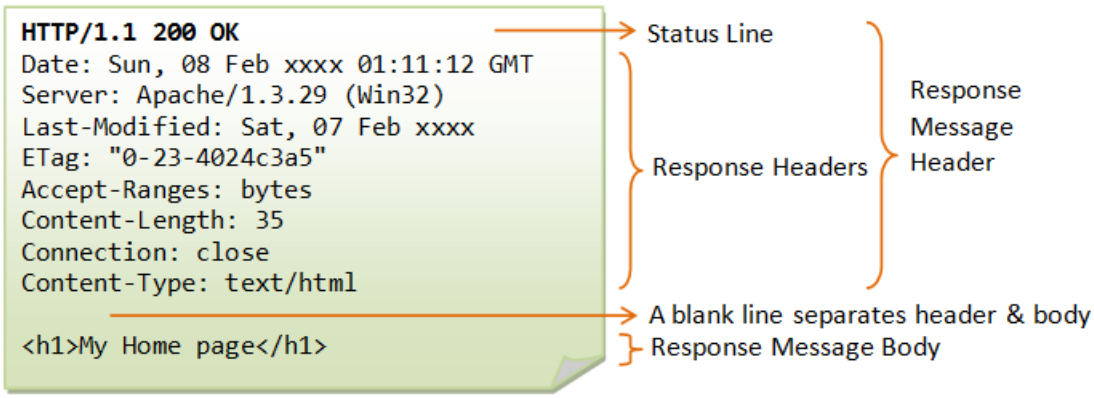
HTTP žiadosť od klienta:



Obrázok 2: HTTP žiadosť od klienta

Zdroj: https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html

HTTP odpoveď od servera:



Obrázok 3: HTTP žiadosť od klienta

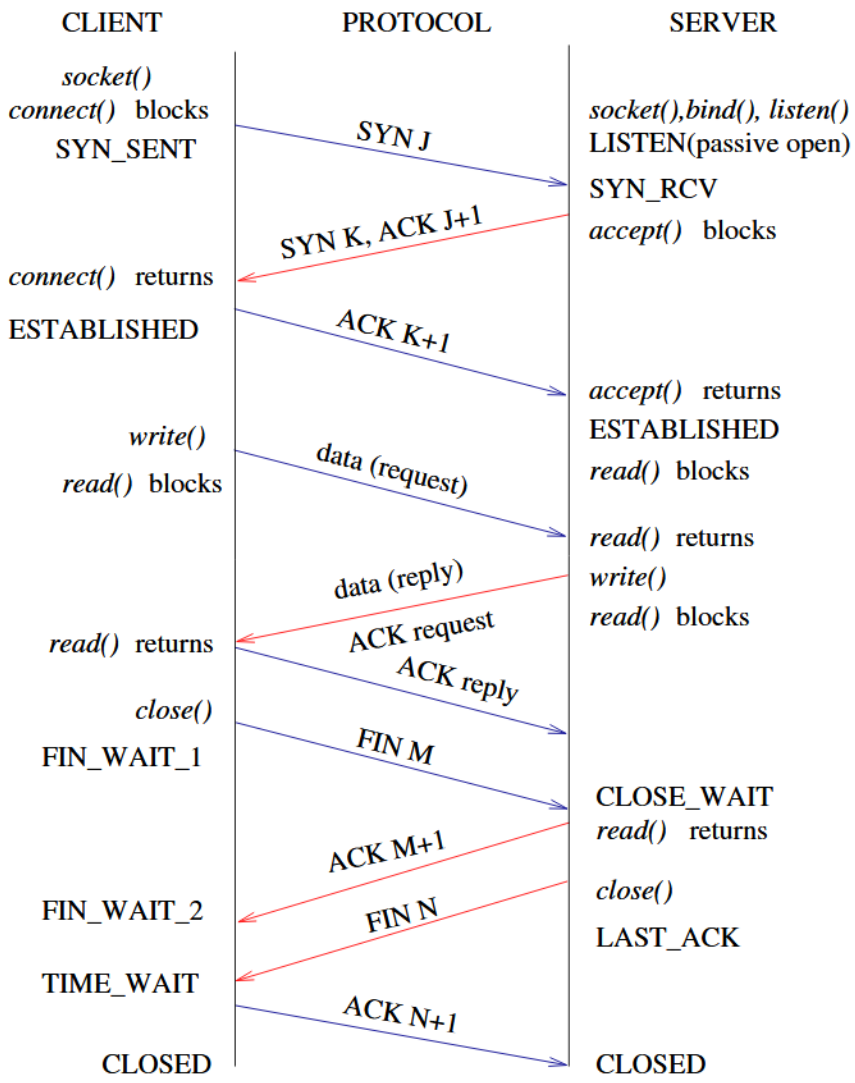
Zdroj: https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html

3. Implementácia

3.1. Riešenie Client-Server komunikácie

Pri implementácii komunikácie medzi klientom a serverom som sa inšpiroval tcp demo aplikáciou od prof. Ing. Petra Matouška, ktorú máme k dispozícii v informačnom systéme pod adresárom isa/príklady. Komunikácia prebieha rovnako ako je zmienené vyššie. Server počúva na porte, ktorý je zadaný ako parameter *isaserver* aplikácie a čaká, kým klient pošle reťazec s požiadavkou. Klient začne požiadavku nadviazaním TCP spojenia na porte, ktorý je zadaný ako parameter *isaclient* aplikácie.

Pre prácu s reťazcami som si naimplementoval štruktúru *string* a pomocné funkcie k nej. Požiadavka a odpoveď sa posielajú pomocou poľa znakov. Keďže nie je v zadaní špecifikovaná veľkosť bufferu, zvolil som si pevnú veľkosť 1024. Táto veľkosť je používaná aj v demo aplikácii a mala by byť postačujúca pre znázornenie funkčnosti aplikácie. Presiahnutie veľkosti bufferu končí ukončením aplikácie s chybovým hlásením.



Obrázok 4: TCP komunikácia

Zdroj: https://wis.fit.vutbr.cz/FIT/st/cfs.php.cs?file=%2Fcourse%2FISA-IT%2Flectures%2Fisa-sockets.pdf&cid=13349

3.2. Implementácia *isaclient* aplikácie

Na začiatku aplikácie sa zaháji kontrola zadaných parametrov. Pri nesprávnom parametre sa aplikácia ukončí s chybovým hlásením.
Po kontrole sa vykoná funkcia *handleArguments(),* ktorá zistí typ HTTP žiadosti podľa príkazu zadaného ako parameter aplikácie. Typ žiadosti a ostatné položky príkazu, ako napr. meno nástenky, identifikačné číslo príspevku, alebo obsah tela(content) sú predané ako parametre do funkcie *createRequest()*. Funkcia slúži na vytvorenie HTTP správy, pridáva potrebné hlavičky do správy a pri potrebe aj obsah tela.

Po vytvorení žiadosti sa zistí port a host servera, obe sú zadané ako parametre aplikácie. Preklad doménového mena sa rieši pomocou funkcie *gethostbyname()*. Klient vytvorí socket pomocou funckie *socket()* a pripojí sa na server pomocou funkcie *connect()*. Pomocou funkcie *getsockname()* sa zistí lokálna IP adresa a port klienta. Tieto údaje sú potrebné pri posielaní žiadosti na server. Pri hore uvedených funkciách sú implementované aj chybové hlásenia.
Klient pošle žiadosť na server a čaká na odpoveď. Hlavička získaná z odpovede sa vypíše na stderr a obsah tela na stdout.

3.3. Implementácia *isaserver* aplikácie

Pre prácu s nástenkami a príspevkami som si naimplementoval dvojsmerne viazané lineárne zoznamy. Použil som implementáciu pomocných funkcii z môjho predošlého projektu pre predmet Algoritmy. Štruktúra *zoznam* obsahuje ukazovateľ na prvú nástenku. Štruktúra *nástenka* obsahuje meno, ukazovateľ na predošlú a nasledujúcu nástenku a ukazovateľ na prvý a posledný príspevok.

Na začiatku aplikácie sa taktiež vykoná kontrola parametrov. Po kontrole sa vytvorí socket cez funkciu *socket()*. Port serveru sa zistí z parametrov aplikácie na ktorý sa pripojí pomocou funkcie *bind()*. Po úspešnom vykonaní predošlých funkcií server čaká na klienta a schvaľuje pripojenie cez funkciu *accept()*. Po úspešnom spojení server príjme správu od klienta pomocou funkcie *read()*. Správa sa predá ako parameter do funkcie *processRequest()*, ktorá rozdelí správu na jednotlivé riadky, ktoré sú ďalej spracované vo funkcii *processLine()*. Podľa riadkov sa zistí typ požiadavky a jednotlivé hlavičky, ako Content-Length a Content-Type. Ostatné hlavičky aplikácia ignoruje. Tieto informácie sa ukladajú do štruktúry *rqst*. Po získaní potrebných informácií sa vykoná funkcia *createResponse()*, ktorá slúži na vykonanie jednotlivých úloh nad nástenkami, alebo príspevkami. Podľa úspešnosti sa vytvorí odpoveď na požiadavku a server vracia požadované údaje s príslušným kódom, hlavičkami alebo aj telom. Odpoveď sa pošle späť ku klientovi. Spojenie s klientom sa zruší a server čaká na nové spojenie.

3.4. Návod na použitie

Oba programy po spustení s parametrom -h vypíšu na stdout nápovedu o spôsobe spustenia.

Server akceptuje iba jeden povinný parameter, **-p**, ktorý určuje port na ktorom bude server očakávať spojenia.
Príklad použitia: **./isaserver -p 4242**

Klient akceptuje povinné parametre, **-H** pre zadanie hosťovskej adresy, **-p** pre zadanie portu servera a príkazy **<command>.**

Podporované príkazy:

- **boards** – vypíše zoznam násteniek
- **board add <name>** - pridá nástanku <name>
- **board delete <name>** - vymaže nástenku <name>
- **board list <name>** - vypíše zoznam príspevkov na nástenke <name>
- **item add <name> <content>** - pridá príspevok do nástenky <name>
- **item delete <name> <id>** - zmaže príspevok z nástenky <name>
- **item update <name> <id> <content>** - prepíše príspevok na nástenke <name>

Príklad použitia: **./isaclient -H localhost -p 4242 boards**

4. Záver

Aplikácie som vypracoval samostatne. Inšpiroval som sa demo tcp aplikáciou, ktorú máme k dispozícii v informačnom systéme. Na preklad aplikácií som použil súbor Makefile z demo aplikácie. Pomocné štruktúry ako string a lineárne zoznamy som implementoval podľa predošlých školských projektov. Aplikácie boli testované na školskom serveri merlin.

5. Použitá literatúra

ISA sockets prednáška: https://wis.fit.vutbr.cz/FIT/st/cfs.php.cs?file=%2Fcourse%2FISA-IT%2Flectures%2Fisa-sockets.pdf&cid=13349

Hypertextový prenosový protokol: https://sk.wikipedia.org/wiki/Hypertextov%C3%BD_prenosov%C3%BD_protokol

https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html

Demo TCP aplikácia: https://wis.fit.vutbr.cz/FIT/st/cfs.php.cs?file=%2Fcourse%2FISA-IT%2Fexamples&cid=13349