Síťové aplikace a správa sítí

HTTP nástěnka

Martin Bartoš

17.11.2019

Obsah

1. Úvod

Tento projekt byl vytvořen v rámci předmětu ISA (Síťové aplikace a správa sítí), kde byla vybrána varianta projektu ***HTTP nástěnka***. Aplikace umožňuje klientům spravovat nástěnky na serveru pomocí **HTTP API**. API jim umožňuje prohlížet, přidávat, upravovat a mazat příspěvky na nástěnkách jako i nástěnky samotné.

Nástěnka obsauje seřazený seznam textových příspěvků. V této aplikaci figuruje jeden server a poté několik klientů, kteří posílají požadavky na server a tím upravují dané nástěnky.

2. Termíny a použité technologie

2.1 API

API (zkratka pro Application Programming Interface) označuje v informatice rozhraní pro programování aplikací. Tento termín používá softwarové inženýrství. Jde o sbírku procedur, funkcí, tříd či protokolů nějaké knihovny (ale třeba i jiného programu nebo jádra operačního systému), které může programátor využívat.

API určuje, jakým způsobem jsou funkce knihovny volány ze zdrojového kódu programu. Rozhraní, které se vytváří při kompilaci a je využíváno při běhu programu, se nazývá ABI. [1]

2.2 HTTP

HTTP (Hypertext Transfer Protocol) je internetový protokol určený pro komunikaci s WWW servery. Slouží pro přenos hypertextových dokumentů ve formátu **HTML**, **XML**, i jiných typů souborů. Používá obvykle port TCP/80, verze 1.1 protokolu je definována v RFC 2616. Společně s elektronickou poštou je HTTP nejvíce používaným protokolem, který se zasloužil o **obrovský rozmach internetu** v posledních letech.

V současné době je používán i pro přenos dalších informací. Pomocí rozšíření **MIME** umí přenášet jakýkoli soubor (podobně jako email), používá se společně s formátem XML pro tzv. webové služby (spouštění vzdálených aplikací) a pomocí aplikačních bran zpřístupňuje i další protokoly, jako je např. FTP nebo SMTP.[2]

2.3 Socket

Síťový **socket** (anglicky network socket) je v informatice koncový bod připojení přes počítačovou síť. S rozvojem Internetu většina komunikace mezi počítači používá rodinu protokolů **TCP**/**IP**. Vlastní přenos zajišťuje IP protokol, a proto je používáno i označení internetový socket.

Vlastní socket je handle (abstraktní odkaz), který může program použít při volání síťového rozhraní pro programování aplikací (**API**), například ve funkci „odeslat tato data na tento socket“. Sockety jsou vnitřně často jen celá čísla, která odkazují do tabulky aktivních spojení.[3]

2.4 C++

C++ je multiparadigmatický **programovací jazyk**, který vyvinul Bjarne Stroustrup a další v Bellových laboratořích AT&T **rozšířením jazyka C**. C++ podporuje několik programovacích stylů (paradigmat) jako je procedurální programování, objektově orientované programování a generické programování, není tedy jazykem čistě objektovým. V současné době patří C++ mezi **nejrozšířenější programovací jazyky**.[4]

3. Návrh a implementace

Základním stavebním kamenem tohoto projektu byl programovací jazyk C++. Tato aplikace má danou strukturu, která bude popsána níže. Budou vysvětleny všechny moduly, které jsou dostupné v tomto projektu a popsány jejich funkcionality.

3.1 Moduly

Tato aplikace obsahuje moduly, které jsou potřebné pro správný chod aplikace.

3.1.1 Common

Zde jsou obsaženy třídy, které jsou využity jak u serveru, tak i u klienta. Jsou to především pomocné třídy, které například vytvářejí **HTTP Requests**, nebo **HTTP Responses**. Také je zde obsažena třída *Parser,* která je rodičovskou třídou pro ***ClientParser*** a ***ServerParser***. Tyto třídy mají na starosti parsování systémových argumentů z CLI a následnou validací vstupů. Také za zmínku stojí třída ***Errors***, která obsahuje statické metody *error*, kde je zaručeno správné ukončení programu při chybě a následné prezentaci chyby.

3.1.2 Client-core

Tento modul pouze obsahuje dvě třídy a to ***Client*** a ***ClientParser***. Jak již bylo řečeno u ***Common*** modulu, ***ClientParser*** je potomek třídy ***Parser*** a zpracovává a validuje vstupy z příkazové řádky. Správná syntaxe zápisu parametrů u ***ClientParseru*** bude popsána později. Nejdůležitější třídou je ***Client***. Tato třída obsahuje hlavní funkcionalitu aplikace z klientské strany. Tato třída bude také popsána později.

3.1.3 Server-core

Jako ***Client-core*** tento modul obsahuje dvě třídy ***Server*** a ***ServerParser***. ***ServerParser*** dědí všechny metody z rodičovské třídy ***Parser*** a pouze bere jiné vstupní argumenty než dříve zmiňovaný ***ClientParser***. Třída ***Server*** se stará o hlavní běh serveru na daném portu pomocí argumentu v CLI.

3.2 Implementační detaily

Zde budou popsány všechny důležité třídy této aplikace pro správně použití a porozumění. Mezi nejdůležitější patří ***ClientParser***, ***ServerParser***, ***Client*** *a* ***Server***.

3.2.1 Třída *ClientParser*

Tato třída se stará o parsování argumentu pro spuštění klienta. Je přesně zadaná syntaxe, která musí být splněna pro každé správně spuštění klienta. Vždy tento klient vezme jeden příkaz, který vykoná na serveru a skončí. Syntax spuštění:

Vypsání pomocné zprávy

./isaclient -h

Spuštění klienta

./isaclient -H <HOST> -p <PORT> <COMMAND>

Kde:

- ***<HOST>*** *hostname, nebo IP adresa, kde server naslouchá*
- *<****PORT****> port, na kterém server naslouchá*
- *<****COMMAND****> příkazy, které budou vyvolány na serveru:*
  - **boards** - *vypíše všechny nástěnky*
  - **board add <name>** - *přidá nástěnku, kde* ***name*** *je název nástěnky*
  - **board delete <name>** - *smaže nástěnku, kde* ***name*** *je název nástěnky*
  - **board list <name>** - *vypíše všechny příspěvky v nástěnce* ***name***
  - **item add <name> <content>** - *přidá příspěvek do nástěnky* ***name*** *s obsahem* ***content***
  - **item delete <name> <id>** - *smaže příspěvek v nástěnce* ***name*** *s* ***id***
  - **item update <name> <id> <content>** - *upraví příspěvek v nástěnce* ***name*** *s* ***id*** *a obsahem* ***content***

Např.= ./isaclient -H localhost -p 8282 board add nastenka1

3.2.2 Třída ServerParser

Tato třída se stará o parsování argumentu pro spuštění serveru. Je přesně zadaná syntaxe, která musí být splněna pro každé správně spuštění serveru. Tato třída bere pouze jeden argument programu a to definování portu, nebo vypsání ***help***. Syntax spuštění:

Vypsání pomocné zprávy

`./isaserver -h`

Spuštění klienta

`./isaserver -p <PORT>`

Kde:

- <**PORT**> port, na kterém server bude naslouchat

3.2.3 Třída Client

Tato třída, jak již bylo dříve zmíněno, je nedůležitejší pro client-side aplikace. Konstruktor této třídy obsahuje host, port a vygenerovaný request z *ClientParser.* Nejdůležitější je především metoda ***run()***, která spustí klienta. Co se vlastně děje v metodě **run**() ? Na začátku se inicializují informační proměnné, kde jsou uloženy všechny potřebné informace. Především jde o address info, nebo socket info.

Nastaví se v dané proměnné port, IP verze 4, TCP,... Potom pomocí funkce ***getaddrinfo(...)*** zjistíme IP adresu daného hosta. Vytvoříme socket. GetAddrInfo funkce nám může vrátit více IP adres a proto se zkusíme připojit k nějaké z nich.

Poté zkontrolujeme velikost requestu a pošleme request na server a čekáme na odpověď ze serveru, kterou poté prezentujeme na STDOUT(obsah) a STDERR(Response hlavičku). Po odpovědi zavíráme socket a tímto je tato komunikace dokončena.

3.2.4 Třída Server

Třída *Server*, stejně jako v třídě *Client*, obsahuje metodu **run**(), která slouží pro spuštění serveru. Konstruktor pro server bere jako argument pouze *port*, na kterém bude server spuštěn. V metodě **run**() se vytvoří také socket se všemi potřebnými parametry. Provede se ***bind*** socketu.

Poté server poslouchá na události, které se stanou u vytvořeného socketu. Dále v nekonečné smyčce čeká na to, až nějaký klient pošle zprávu na server a mohl tuto zprávu zpracovat. Zde jsou další pomocné metody, jako např. ***ManageRequest***, která se postará o vykonání requestu od klienta a poté vygeneruje Response ze serveru a pošle zpět klientovi. A takto je zmapovaná celá komunikace klient-server.

4. Návod k použití

Pro správné spuštění aplikace následujte tyto kroky.

1. Kompilace zdrojových souborů. V adresáři obsažen i Makefile, díky kterému můžeme jednoduše zkompilovat.

- ***make*** nebo ***make all***
- ***make isaclient*** (volitelné; pouze zkompiluje clienta)
- ***make isaserver*** (volitelné; pouze zkompiluje server)

2. (Volitelné) Pro vypsání pomocné hlášky na serveru.

- ***./isaclient -h***
- ***./isaserver -h***

3. Spuštění serveru na libovolném volném portu.

- ***./isaserver -p 8282***

4. Generování requestů pomocí ***isaclient*** targetu. Port musí být shodný jako u serveru a hostname serveru(nebo Ipv4 adresa) např.

- ***./isaclient -H localhost -p 8282 board add nastenka1***
  - Vytvoří nástěnku s názvem **nastenka1**
- ***./isaclient -H localhost -p 8282 boards***
  - Vypíše všechny nástěnky – v tomto případě pouze **nastenka1**
- ***./isaclient -H localhost -p 8282 item add prispevek1 „Toto je muj prvni prispevek \n pro vsechny“***
  - Vytvoří příspěvek **prispevek1** v nástěnce **nastenka1** s obsahem **content**

Příkazy popsány v sekci ClientParser

5. Závěr

Cílem tohoto projektu bylo vyzkoušení si komunikace client-server s HTTP API. Aplikace funguje korektně k zadání projektu a je 100% funkční a přenositelná na ostatní platformy. Aplikace při spuštění serveru vytváří novou instanci serveru a tudíž neobsahuje žádná data.

Data nejsou perzistentní, což by mohl být další cíl pokračování tohoto projektu. Z mého pohledu, tento projekt byl pro mě velkým přínosem, kde jsem si mohl prohloubit znalosti ze sítí a C++. Myslím si, že tento projekt je vytvořen tak, aby byl velice lehce rozšířitelný a dobře udržovatelný a to se, podle mě, povedlo.

6. Literatura

[1] API. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-11-18]. Dostupné z: https://cs.wikipedia.org/wiki/API

[2] HTTP. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-11-18]. Dostupné z: https://cs.wikipedia.org/wiki/Hypertext_Transfer_Protocol

[3] Socket. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-11-18]. Dostupné z: https://cs.wikipedia.org/wiki/Síťový_socket

[4] C++. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-11-18]. Dostupné z: https://cs.wikipedia.org/wiki/C%2B%2B