Otava 216

zápočtový program, Eduard Hopfer

Stručný popis

Program tvoria dve časti: **webová aplikácia** a **server**, na ktorom beží. Webová aplikácia slúži ako Chatovacia miestnosť v prehliadači. Používateľ sa zaregistruje a potom sa vie pripojiť do existujúcich chatovacích miestností, alebo aj vytvoriť svoje vlastné.

Detailný popis

Server

Server jednoducho čaká na **HTTP requesty** a snaží sa im vyhovieť. Ak na ňom nebeží žiadna aplikácia, predpokladá, že sú to len požiadavky typu **GET** a ak žiadaný súbor je na serveri, dá nám ho. Na prázdne požiadavky odpovedá klasicky súborom **Index.html**, ak existuje. V prípade, že súbor sa na serveri nenachádza dostaneme príslušnú chybovú hlášku, v tomto prípade 404 – Not found. Aj bez konfigurácie sa teda dá použiť pre statické webstránky.

Vlastné správanie sa dá pridať jednoducho zavolaním funkcie ***server.AddRoute(Route route)***. Objekty typu Route sú popísané nižšie, v skratke spájajú url s funkciou, ktorá sa má zavolať, keď na ňu prijde požiadavka. Programátor webovej aplikácie sa teda nemusí skoro vôbec starať o to, čo sa deje na serveri.

Server podporuje „relácie“, teda ak naňho dá používateľ požiadavku, server ho istý čas pozná a pamätá si nejaké informácie, napríklad či je používateľ prihlásený atd. To, čo si relácia pamätá už si však rieši webová aplikácia sama.

Webová aplikácia

Akcie na webstránke v prehliadači posielajú asynchrónne požiadavky na server a ten odpovedá naprogramovaným správaním. Teda napríklad keď sa používateľ chce prihlásiť, server dostane požiadavku typu **POST** na url **/api/login** s prihlasovacími údajmi ako parametrami, čo zavolá na serveri funkciu **LoginHandler**, ktorá kontaktuje databázu a zistí, či sú údaje správne a server potom adekvátne odpovie prehliadaču.

Súbory webstránky, teda html, css, javascript a iné, sú uložené v priečinku **wwwroot**

Navigácia webstránky

Na chatovanie je potrebné si zaregistrovať účet a prihlásiť sa. Pri prvom prihlásení si používateľ vyberá ikonku, tá je len kozmetická a bude ju vidno, keď pošle správu.

Predtým, ako môže používateľ chatovať, musí byť v nejakej chatovacej miestnosti. Tú môže sám vyrobiť, alebo sa pridať do už existujúcej. Toto vie urobiť kliknutím na tlačítko **plus**. Tlačítko **Ozubené koleso** otvorí okno, kde sa prepína medzi chatovacími miestnosťami, v ktorých je užívateľ členom. Na vstúpenie do chatovacej miestnosti potrebujete jej ID. To vám môže dať hocikto, kto je už členom tak, že klikne na názov miestnosti a ID sa mu automaticky skopíruje do schránky.

Odhlásiť sa je možné kliknutím na vaše uživateľské meno v pravom hornom rohu.

Toto sú len stručné inštrukcie, dizajn webstránky je vcelku intuitívny, takže s navigáciou by nemal byť problém.

Technická špecifikácia

Server je založený na objekte **HttpListener** z namespaceu **System.Net** štandardnej knižnice. Hlavná trieda je **WebServer** a tá používa objekty tried *SessionManager, Router* a mnoho malých pomocných tried. Popis atrribútov a metód je nižšie:

WebServer

public void AddRoute(Route route)

- pridá route do známych ciest na serveri
- existuje aj preťaženie, ktoré pridá celý array routov

public void Start()

- začne čakanie na požiadavky, server je online až po zavolaní tejto funkcie

private async Task HandleRequestsAsync()

- získa požiadavky od httplistenera, zavolá Router na ich spracovanie a odošle odpoveď naspäť

private Dictionary<string, string> GetParams(HttpListenerRequest request)

- získa parametre GET alebo POST requestu a vráti ich ako slovník stringov

Session

Dátová štruktúra na uchovávanie informácií o pripojenom používateľovi

SessionManager

Mapuje IP adresy na Sessiony

public Session GetSession(IPEndPoint endPoint)

- vráti Session prislúchajúci k danej IP adrese, ak záznam neexistuje, najprv ho vytvorí

public void RemoveInvalidSessions()

- odstráni Sessiony, ktoré už niesu aktuálne

Route

Dátová štruktúra na prácu so špeciálnymi cestami. Každý route má cestu na serveri (napr. /api/login), typ požiadavky a Controller – čo sa má stať, ked niekto podá takúto požiadavku

ResponseData

Dátová štruktúra, predstavuje odpoveď servera na požiadavku. Obsahuje informácie pre browser(status kód, typ posielaných dát) ako aj dáta samotné vo forme byte arrayu.

ExtensionInfo

Dátová štruktúra, predstavuje typ súboru, ktorý server podporuje s informáciami, ako má daný typ pripraviť na odoslanie

Router

Rozhoduje, čo sa stane s danou požiadavkou. Odošle požiadavku ďalej špeciálnym Controllerom alebo len načíta lokálny súbor servera a pripraví ho na odoslanie.

Funkcie typu Loader(string filename, string ext, ExtensionInfo extInfo)

- *FileLoader, PageLoader a ImageLoader*
- prečítajú daný súbor a zmenia ho do binárnej podoby
- všeobecne FileLoader pre textové súbory, ImageLoader pre binárne súbory
- PageLoader pracuje rovnako ako FileLoader, ale je flexibilnejší s názvami súborov,

  uzná napr názvy bez prípony (*Index → Index.html*)

ResponseData Route(Session session, string verb, string dest, Dictionary<string, string> kwargs)

- Zavolá správny Loader, alebo posunie dáta špeciálnemu Controlleru

Handler

Je funkcia typu

*public static ResponseData ***(Session session, Dictionary<string, string> kwargs)*

Controller

Triedy typu Controller dedia z triedy **BaseController**. Controllery slúžia ako obal pre jednotlivé Handler funkcie. Používajú sa na obmedzenie prístupu k niektorým zdrojom na webstránke. Existujú 3 Controllery: **Anonymous**, **Authorized** a **AuthorizedExpirable**. Ako ich mená napovedajú *Anonymous* nič nerieši a rovno volá Handler funkciu, *Authorized* predtým skontroluje, či má aktuálny používateľ práva na zobrazenie daného obsahu a *AuthorizedExpirable* ešte navyše vyžaduje aby práva neboli príliš staré.

Komunikácia s “databázou”

Ako databáza slúžia **json** súbory. Prácu s nimi má na starosti statická trieda **JSONFileService**. Obsahuje funkcie na čítanie a zapisovanie do súborov, ktoré zahŕňajú serializáciu zo C# tried do json objektov a naopak deserializáciu do C# objektov.

public static List<T> GetAll<T>()

- podľa typu T prečíta daný súbor (buď *Users*, alebo *ChatRooms*) a deserializuje ho do príslušnej dátovej štruktúry (User, alebo ChatRoom)

public static void Add<T>(T newObj)

- pridá do databázy nový objekt

public static void Update<T>(Guid objID, Guid toAdd)

public static void Update(Guid objID, string icon)

public static void Update(Guid objID, Message message)

- aktualizuje objekt v databáze s novými dátami

Autorizácia

Heslá v databáze niesu len tak pohodené, ale sú zahashované algoritmom **AES**. To má na starosti trieda **AesEncryptor**. Tá je založená na tomto príklade https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.aes?view=netcore-3.1 .

Handler funkcie a komunikácia s browserom

V browseri je použitá knižnica jquery, pomocou ktorej tlačítka asynchrónne posielajú požiadavky na server, a tak volajú jednotlivé Handler funkcie. Zoznam špeciálnych požiadavok s ich Handler funkciami:

- POST api/login | prihlasovacie údaje ako form data → LoginHandler
  - porovná dáta s databázou a adekvátne odpovie
  - spracúva registráciu aj prihlásenie
- GET api/chatinit → InitializeChatroom
  - pošle browseru zoznam všetkých používateľových chatovacích miestností
- POST api/createRoom | názov novej miestnosti ako form data → CreateRoom
  - vytvorí novú miestnosť a pridá ju do databázy
  - tvorca je na začiatku jej jediným členom
- GET api/messages | *id* ako parameter → GetMessages
  - pošle browseru všetky správy z chatovacej miestnosti s daným *id*
- POST api/messages | správa ako form data → AddMessage
  - aktualizuje správy aktívnej miestnosti s novou správou
- GET api/seticon | nazov ikonky ako parameter → SetIcon
  - aktualizuje ikonku aktívneho používateľa

- GET api/join | id ako parameter → JoinRoom
  - pridá aktívneho používateľa do miestnosti s daným *id*
- GET api/getuser → GetCurrentUserData
  - získa z databázy detailné informácie o aktívnom používateľovi
- GET api/logout → Logout
  - vymaže reláciu aktívneho používateľa a presmeruje ho na domovskú stránku

Nedostatky

Asi každý uzná, že Chat, ktorý nepracuje v reálnom čase veľmi zaujímavý nieje. Bola tu snaha používania technológií ako *SignalR*, ktorý však nevie bežať na takomto podomácky vyrobenom serveri, alebo web sockety, ktorých implementácia by zabrala príliš veľa času. Ďalej databáza vo forme textových súborov je v podstate nepoužiteľná pre takýto projekt ako sa bude počet používteľov a správ v chatovacích miestnostiach zvyšuje. Aby som našiel daného používateľa, musím celý súbor načítať do pamäti a prejsť ho, čo nieje veľmi efektívne a robiť to pre viac používateľov naraz priam nemožné. Taktiež sa nemôžem tváriť, že moje ukladanie hesiel je bezpečné, ale ako sa ukázalo, bezpečné ukladanie hesiel je veľmi komplikovaný odbor. Moje riešenie je však stále lepšie ako nič. Použitie reálnej databázy a web socketov by projektu veľa pridali, ale to už je úloha na inokedy.

Záver

Pri práci na tomto programe som zistil, že naprogramovať jednoduchý webový server nie je také ťažké, ako by človek povedal. Tiež som sa utvrdil v tom, že nikdy nechcem programovať webstránky a že perfecionizmus nepomáha produktivite. V C# sa po tejto skúsenosti už cítim ako doma takže celkovo to hodnotím ako plus.

Systémové požiadavky a spustenie

Server beží na platforme .NET Core verzia >=3 a bol testovaný ako na Linuxe(Fedora 31), tak aj na Windowse(Windows 10). Webstránka sa zobrazí správne v akomkoľvek modernom prehlidači, s výminkou **firefox**u, ktorý nepodporuje dialógové okná.

Pred spustením je potrebné v súbore **Program.cs** prepísať premennú **ProgramDir** na cestu ku priečinku, kde sa projekt nachádza.