//æøå

import \* as React from 'react';

import { RouteComponentProps } from 'react-router';

import { Cookie } from '../utilities/cookie'

import { User } from "../utilities/user";

import { Http } from "../utilities/http";

import { TypeStyle } from "./TypeStyle"

//the React component is for the user to browser a specific community

export class Community extends React.Component<RouteComponentProps<

{

communityId: string

}>, {}> {

community: GetCommunityResultCommunity | null = null;

chatContainer: HTMLDivElement | null = null;

chatMessages: GetMessageResultMessage[] | null = null;

constructor(\_props: RouteComponentProps<{ communityId: string }> | undefined) {

super(\_props);

let props = \_props!;

if (User.current != null) {

this.initRESTMessage(props);

Http.post<GetCommunityResult>('api/CommunityGateway/GetCommunity', {

userKey: User.current!.key,

uniqueName: props.match.params.communityId

}).then(data => {

if (data.authorized && data.success && data.community != null) {

this.community = data.community;

this.setState({});

}

else {

console.log(data);

//window.location.href = ("/Home");

}

})

}

}

public async initRESTMessage(props: RouteComponentProps<{ communityId: string }>): Promise<void> {

console.log("rest message call");

let data: RESTMessageResult;

try {

data = await Http.post<RESTMessageResult>('api/CommunityGateway/RESTMessage', {

userKey: User.current!.key,

communityId: props.match.params.communityId

})

}

catch (exception) {

setTimeout(() => {

if (this.props) {

this.initRESTMessage(this.props)

}

else {

this.initRESTMessage(props);

}

}, 1000);

return;

}

if (this.chatMessages) {

if (this.chatMessages.find(x => x.objectId == data.restMessage.objectId) == undefined) {

this.chatMessages.unshift(data.restMessage);

}

}

else {

this.chatMessages = [data.restMessage];

}

this.setState({});

if (this.props) {

this.initRESTMessage(this.props)

}

else {

this.initRESTMessage(props);

}

}

public loadMessages(): void {

Http.post<GetMessageResult>('api/CommunityGateway/GetMessage', {

userKey: User.current!.key,

communityId: this.props.match.params.communityId,

skip: this.chatMessages ? this.chatMessages.length : 0,

limit: (this.chatMessages ? this.chatMessages.length : 0) + 10

}).then(data => {

if (data.authorized && data.success) {

if (this.chatMessages == null) {

this.chatMessages = [];

}

for (let i: number = 0; i < data.messages.length; i++) {

this.chatMessages[this.chatMessages.length] = data.messages[i];

}

if (data.messages.length == 0) {

return;

}

this.setState({});

this.chatContainer!.scrollTop = this.chatContainer!.scrollHeight - this.chatContainer!.clientHeight;

if (this.chatContainer!.scrollTop == 0) {

this.loadMessages();

}

}

else {

console.log(data);

}

});

}

public render() {

if (User.current == null) {

this.props.history.push("/Home");

return <div></div>;

}

if (this.community == null) {

return <div>loading</div>

}

else {

return <div>

<TypeStyle css={

[

{

identifier: ".column",

style: {

width: "50%",

float: "left"

}

},

{

identifier: ".row:after",

style: {

display: "table",

clear: "both",

}

}

]

} />

<div className="row">

<div className="column">

<h2>{this.community.name}</h2>

<h4>users</h4>

<ul>

<li>

<input onKeyUp={(e) => {

if (e.keyCode == 13) {

Http.post<InviteResult>('api/CommunityGateway/Invite', {

userKey: User.current!.key,

communityId: this.props.match.params.communityId,

userToInvite: e.currentTarget.value

}).then(data => {

if (data.success && data.authorized) {

this.community!.users[this.community!.users.length] == e.currentTarget.value;

this.setState({});

}

else {

console.log(data);

alert(data.message);

}

})

}

}} placeholder="invite" />

</li>

{(() => {

let re: React.ReactElement<HTMLLIElement>[] = [];

for (let i: number = 0; i < this.community!.users.length; i++) {

re[re.length] = <li key={i}>

{this.community!.users[i]}

</li>

}

return re;

})()}

</ul>

<h4>Boxes</h4>

<ul>

{(() => {

let re: JSX.Element[] = [];

for (let i: number = 0; i < this.community.boxKey.length; i++) {

re[re.length] = <li key={i}>

{this.community!.boxName[i]}

<span style={{

fontSize: "11px",

color: "#c2c2a3"

}}> in </span>

{this.community!.boxLocation[i]}

[<a style={{ cursor: "pointer" }} onClick={async () => {

let newName = prompt("new name");

if (newName == null) {

return;

}

let renameResult = await Http.post<RenameBoxResult>("api/BoxGateway/RenameBox", {

userKey: User.current!.key,

boxKey: this.community!.boxKey![i],

newName: newName

});

if (renameResult.success) {

this.community!.boxKey![i] = newName;

this.setState({});

}

else {

alert("error");

console.log(renameResult);

}

}}>rename</a>]

[<a style={{ cursor: "pointer" }} onClick={() => {

alert("comming soon™");

}}>give to user</a>]

</li>

}

return re;

})()}

</ul>

</div>

<div className="column" style={

{

left: "0px",

float: "right",

height: "100vh",

width: "40%",

backgroundColor: "#444547"

}

}>

<div ref={(input) => { this.chatContainer = input }} style={{

overflowY: "auto",

overflowX: "hidden",

height: "95vh",

wordWrap: "break-word",

whiteSpace: "pre-wrap",

}} onScroll={(e: React.UIEvent<HTMLDivElement>) => {

if (e.currentTarget.scrollTop == 0) {

this.loadMessages();

}

}}>

<div>

{(() => {

if (this.chatMessages == null) {

this.loadMessages();

return <div>loading</div>;

}

let re: JSX.Element[] = [];

for (let i: number = this.chatMessages.length - 1; i > -1; i--) {

let side: string = this.chatMessages![i].author != User.current.username ? "left" : "right";

re[re.length] = <div className="chatMessage" key={i}>

<p style={

{

float: side,

clear: "both",

position: "relative",

backgroundColor: "#d9d9d9",

borderRadius: "1vh",

maxWidth: "50%",

padding: "3px",

}

} title={(new Date(this.chatMessages![i].timeOfSending \* 1000)).toLocaleDateString()}>

<span className="chatMessageAuthor" style={

{

float: side,

fontSize: "11px",

color: "#737373"

}

}>{this.chatMessages![i].author}</span>

<br />

{this.chatMessages![i].content}

</p>

</div>

}

return re;

})()}

</div>

</div>

<input type="text" style={{ height: "5vh", float: "left", bottom: 0, position: "absolute", width: "40%" }}

onKeyUp={(e: React.KeyboardEvent<HTMLInputElement>) => {

if (e.keyCode == 13) {

Http.post<GetCommunityResult>('api/CommunityGateway/SendMessage', {

userKey: User.current!.key,

message: e.currentTarget.value,

communityId: this.community!.uniqueName

}).then(data => {

if (!(data.authorized && data.success)) {

alert("error");

console.log("data");

}

});

e.currentTarget.value = "";

}

}}

/>

</div>

</div>

</div>

}

}

}

interface GetCommunityResult {

type: string;

authorized: boolean;

message: string;

success: boolean;

community: GetCommunityResultCommunity | null;

}

interface GetCommunityResultCommunity {

name: string;

uniqueName: string;

users: string[];

owner: string;

admins: string[];

description: string;

boxKey: string[];

boxLocation: string[];

boxName: string[];

}

interface GetMessageResult {

type: string;

authorized: boolean;

message: string;

success: boolean;

messages: GetMessageResultMessage[];

}

interface GetMessageResultMessage {

author: string;

timeOfSending: number;

content: string;

objectId: string;

}

interface RESTMessageResult {

type: string;

authorized: boolean;

message: string;

success: boolean;

restMessage: GetMessageResultMessage;

}

interface InviteResult {

type: string;

authorized: boolean;

message: string;

success: boolean;

}

interface RenameBoxResult {

type: string;

authorized: boolean;

message: string;

success: boolean;

}