//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";

//this React component is to display the user's (or other user's in the future) data and information

//kinda like a facebook profile page

export class Profile extends React.Component<RouteComponentProps<{}>, {}> {

data: ProfileDataResult | null = null;

constructor() {

super();

if (User.current == null) {

window.location.href = "/";

return;

}

Http.post<ProfileDataResult>("api/UserGateway/ProfileData", {

username: User.current!.username,

key: User.current!.key

}).then(x => {

this.data = x;

this.setState({});

}).catch(x => console.log(x));

}

communityList: HTMLUListElement | null = null;

public render() {

if (this.data == null) {

return <p>loading</p>;

}

else {

return <div>

<TypeStyle css={

[

{

identifier: ".column",

style: {

float: "left",

width: "50%",

}

},

{

identifier: ".row:after",

style: {

display: "table",

clear: "both",

}

}

]

} />

<div className="row">

<div className="column">

<h1>{this.data.username}</h1>

<p style={{ cursor: "pointer" }}><a style={{ color: "black" }} onClick={(e) => {

if (this.communityList != null) {

if (this.communityList.style.display == "none") {

this.communityList.style.display = "block";

}

else {

this.communityList.style.display = "none";

}

}

}}>{(() => {

if (this.data.myself) {

return <span>communities...</span>

}

else {

return <span>common communities...</span>

}

})()}</a></p>

<ul ref={(input) => this.communityList = input} style={{ display: "none" }}>

{(() => {

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

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

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

{this.data.communitiesName![i]}

<span style={{ fontSize: "11px", color: "#c2c2a3" }}> {this.data.communitiesUniqueName![i]}</span>

</li>

}

return re;

})()}

</ul>

</div>

<div className="column">

{(() => {

if (this.data.myself) {

return <div>

<h3>NFC</h3>

{(() => {

if (this.data.nfcId == null) {

return <p>no nfc initialized (<a href="/initNfc">initalize a new nfc</a>)</p>;

}

else {

return <p>nfc id: {this.data.nfcId} [<a onClick={async () => {

let result: DeleteNfcResult = await Http.post<DeleteNfcResult>("api/UserGateway/DeleteNfc", {

key: User.current!.key

});

if (result.success) {

this.data!.nfcId = null;

this.setState({});

}

else {

alert("error");

console.log(result);

}

}} style={{ cursor: "pointer" }}>delete</a>]</p>;

}

})()}

<br />

<br />

<h3>Boxes</h3>

<ul>

{(() => {

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

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

re[re.length] = <li id={"box\_" + i} key={i}>

{this.data!.boxes![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.data!.boxesKey![i],

newName: newName

});

if (renameResult.success) {

this.data!.boxes![i] = newName;

this.setState({});

}

else {

alert("error");

console.log(renameResult);

}

}}>rename</a>]

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

const classname = "community\_selector\_for\_box";

const lastnode = (e.currentTarget.parentNode!.lastChild! as HTMLElement);

if (lastnode.className == classname) {

lastnode.remove();

return;

}

let node: HTMLDivElement = document.createElement("div");

node.className = classname;

node.setAttribute("style", TypeStyle.CSSPropertiesToString({

width: "200px",

backgroundColor: "#e5d7ea",

height: "300px",

borderRadius: "25px",

overflow: "auto"

}));

let ulNode = document.createElement("ul");

for (let j = 0; j < this.data!.communitiesName!.length; j++) {

let liNode = document.createElement("li");

liNode.setAttribute("style", TypeStyle.CSSPropertiesToString({

cursor: "pointer",

}));

liNode.innerHTML = this.data!.communitiesName![i];

let spanNode = document.createElement("span");

spanNode.innerHTML = " " + this.data!.communitiesUniqueName![i];

spanNode.setAttribute("style", TypeStyle.CSSPropertiesToString({

fontSize: "11px",

color: "#c2c2a3"

}));

liNode.addEventListener("click", async () => {

let result = await Http.post<MoveBoxResult>("api/BoxGateway/MoveBox", {

userKey: User.current!.key,

boxKey: this.data!.boxesKey![i],

toType: "community",

to: this.data!.communitiesUniqueName![j]

});

if (result.success) {

window.location.reload(true);

}

else {

alert("error");

console.log(result);

}

})

liNode.appendChild(spanNode);

ulNode.appendChild(liNode);

}

node.appendChild(ulNode);

e.currentTarget.parentNode!.appendChild(node);

}}>give to community</a>]

</li>

}

return re;

})()}

</ul>

</div>

}

return null;

})()}

</div>

</div>

</div>;

}

}

}

interface ProfileDataResult {

type: string;

authorized: boolean;

message: string;

success: boolean;

username: string | null;

communitiesName: string[] | null;

communitiesUniqueName: string[] | null;

myself: boolean | null;

nfcId: string | null;

boxes: string[] | null;

boxesKey: string[] | null;

}

interface DeleteNfcResult {

type: string;

authorized: boolean;

message: string;

success: boolean;

}

interface RenameBoxResult {

type: string;

authorized: boolean;

message: string;

success: boolean;

}

interface MoveBoxResult {

type: string;

authorized: boolean;

message: string;

success: boolean;

}