neko_todo ソースリスト

美都

2024 年 12 月 3 日

目次

1　Rust ソース

1.1　メインモジュール　main.rs

```rust
// Prevents additional console window on Windows in release, DO NOT REMOVE!!
#![cfg_attr(not(debug_assertions), windows_subsystem = "windows")]

fn main() {
    neko_todo_lib::run()
}
```

1.2 ライブラリメインモジュール lib.rs

```rust
//! tauri メインプロセス
mod app_status;
mod config;
mod todo;

use app_status::AppStatus;
use tauri::State;
use todo::TodoItem;

#[tauri::command]
fn greet(name: &str) -> String {
    format!("Hello, {}! You've been greeted from Rust!", name)
}

#[cfg_attr(mobile, tauri::mobile_entry_point)]
pub fn run() {
    let app_status = AppStatus::new();
    tauri::Builder::default()
        .plugin(tauri_plugin_shell::init())
        .manage(app_status)
        .invoke_handler(tauri::generate_handler![greet, get_todo_list])
        .run(tauri::generate_context!())
        .expect("error while running tauri application");
}

#[tauri::command]
fn get_todo_list(app_status: State<'_, AppStatus>) -> Result<Vec<TodoItem>, String> {
    println!("todo 取得");

    Ok(app_status.todo().get_todo_list().unwrap())
}
```

1.3 アプリケーションステータス app_status.rs

```rust
//! アプリケーション全体のステータスを保持する。

use crate::{config::NekoTodoConfig, todo::Todo};
use std::sync::{Arc, Mutex};

pub struct AppStatus {
    config: Arc<Mutex<NekoTodoConfig>>,
    todo: Todo,
}

impl AppStatus {
    pub fn new() -> Self {
        let config = match NekoTodoConfig::new() {
            Ok(conf) => Arc::new(Mutex::from(conf)),
            Err(e) => {
                eprintln!("致命的エラー。config の取得に失敗。\n {}", e);
                std::process::exit(1)
            }
        };
        let todo = Todo::new();
        Self { config, todo }
    }

    pub fn config(&self) -> &Mutex<NekoTodoConfig> {
        &self.config
    }

    pub fn todo(&self) -> &Todo {
        &self.todo
    }
}
```

1.4 コンフィグ設定処理 config.rs

```rust
//! アプリケーション設定の取得関係

use directories::BaseDirs;
use std::{
    fs::OpenOptions,
    io::{BufWriter, ErrorKind, Result, Write},
    path::PathBuf,
};

const CONF_FILE_NAME: &str = "neko_todo.conf";
const CONF_DIR_NAME: &str = "neko_todo";
const DB_HOST: &str = "NEKO_DB_DB_HOST";
const DB_USER: &str = "NEKO_DB_DB_USER";
const DB_PASS: &str = "NEKO_DB_DB_PASS";

#[derive(Debug)]
pub struct NekoTodoConfig {
    db_host: String,
    db_user: String,
    db_pass: String,
    dirty: bool,
}

impl NekoTodoConfig {
    pub fn new() -> dotenvy::Result<Self> {
        let file = Self::get_config_file_path().map_err(dotenvy::Error::Io)?;
        dotenvy::from_path(file)?;
        Ok(Self {
            db_host: std::env::var(DB_HOST).unwrap_or_default(),
            db_user: std::env::var(DB_USER).unwrap_or_default(),
            db_pass: std::env::var(DB_PASS).unwrap_or_default(),
            dirty: false,
        })
    }

    pub fn get_db_host(&self) -> &str {
        &self.db_host
    }

    pub fn get_db_user(&self) -> &str {
        &self.db_user
    }

```

```rust
    pub fn get_db_pass(&self) -> &str {
        &self.db_pass
    }

    pub fn set_db_host(&mut self, val: &str) {
        self.db_host = val.to_string();
        self.dirty = true;
    }

    pub fn set_db_user(&mut self, val: &str) {
        self.db_user = val.to_string();
        self.dirty = true;
    }

    pub fn set_db_pass(&mut self, val: &str) {
        self.db_pass = val.to_string();
        self.dirty = true;
    }

    fn save(&self) -> Result<()> {
        let path = Self::get_config_file_path()?;
        let file = OpenOptions::new().write(true).truncate(true).open(&path)?;
        let mut buffer = BufWriter::new(file);
        writeln!(buffer, "{}={}", DB_HOST, self.get_db_host())?;
        writeln!(buffer, "{}={}", DB_USER, self.get_db_user())?;
        writeln!(buffer, "{}={}", DB_PASS, self.get_db_pass())?;
        Ok(())
    }

    /// コンフィグファイルのファイル名を生成する
    /// 必要に応じて、コンフィグファイル用のディレクトリ ("neko_todo") を生成し
    /// さらに、存在しなければ、空のコンフィグファイル ("neko_todo.conf") を生成する。
    fn get_config_file_path() -> Result<PathBuf> {
        // 環境依存コンフィグ用ディレクトリの取得
        let mut path: PathBuf = BaseDirs::new().unwrap().config_dir().into();
        // 必要であれば、自分用のディレクトリを生成する。
        // ここでエラーになるのは、OS システムに問題がある。
        path.push(CONF_DIR_NAME);
        if let Err(e) = std::fs::create_dir(&path) {
            if e.kind() != ErrorKind::AlreadyExists {
                return Err(e);
            }
        }

        // コンフィグファイルがなければ、空のファイルを生成する。
```

```
89          path.push(CONF_FILE_NAME);
90          if let Err(e) = std::fs::File::create_new(&path) {
91              if e.kind() != ErrorKind::AlreadyExists {
92                  return Err(e);
93              }
94          }
95          Ok(path)
96      }
97  }
98
99  impl Drop for NekoTodoConfig {
100     fn drop(&mut self) {
101         if self.dirty {
102             self.save().unwrap();
103         }
104     }
105 }
106
107 #[cfg(test)]
108 mod tests {
109     use super::*;
110
111     /// 環境設定の挙動テスト
112     #[test]
113     #[ignore]
114     fn test_env_val() {
115         let val_db_host = "test_host";
116         let val_db_user = "test_user";
117         let val_db_pass = "test_pass";
118         save_curr_conf_file();
119         {
120             let mut conf = NekoTodoConfig::new().unwrap();
121             // 初期状態では空文字列が返るはず
122             assert_eq!(conf.get_db_host(), "");
123             assert_eq!(conf.get_db_user(), "");
124             assert_eq!(conf.get_db_pass(), "");
125             // test_host をセットしてセットされているか確認。
126             conf.set_db_host(val_db_host);
127             conf.set_db_user(val_db_user);
128             conf.set_db_pass(val_db_pass);
129             assert_eq!(conf.get_db_host(), val_db_host);
130             assert_eq!(conf.get_db_user(), val_db_user);
131             assert_eq!(conf.get_db_pass(), val_db_pass);
132         } // この時点で一旦環境ファイルを保存してみる。
133           // 環境ファイルをもう一度ロードして、環境を確認
```

```
134         delete_env_val();
135         let conf = NekoTodoConfig::new().unwrap();
136         assert_eq!(conf.get_db_host(), val_db_host);
137         assert_eq!(conf.get_db_user(), val_db_user);
138         assert_eq!(conf.get_db_pass(), val_db_pass);
139         restore_curr_conf_file();
140     }
141
142     /// テスト環境のため、元の conf ファイルを退避
143     fn save_curr_conf_file() {
144         let file = NekoTodoConfig::get_config_file_path().unwrap();
145         let mut save_file = file.clone();
146         save_file.set_extension("save");
147         if file.exists() {
148             println!(
149                 "現在の環境ファイル [{:?}] を [{:?}] に退避します。",
150                 &file, &save_file
151             );
152             std::fs::rename(file, save_file).unwrap();
153         }
154     }
155
156     /// テスト環境のための一時ファイルを抹消し、元のファイルを復旧
157     fn restore_curr_conf_file() {
158         let file = NekoTodoConfig::get_config_file_path().unwrap();
159         let mut save_file = file.clone();
160         save_file.set_extension("save");
161         if save_file.exists() {
162             if file.exists() {
163                 println!("テスト用環境ファイル{:?}を削除します。", &file);
164                 std::fs::remove_file(&file).unwrap();
165             }
166             println!(
167                 "元の環境ファイル [{:?}] を [{:?}] に復元します。",
168                 &save_file, &file
169             );
170             std::fs::rename(save_file, file).unwrap();
171         }
172     }
173
174     /// テスト環境のため、環境変数をすべて消去する。
175     fn delete_env_val() {
176         std::env::remove_var(DB_HOST);
177         std::env::remove_var(DB_USER);
178         std::env::remove_var(DB_USER);
```

```
179         }
180     }
```

1.5 todo モデル処理 todo.rs

```rust
use chrono::{Days, Local, NaiveDate};
use serde::Serialize;

/// todo 一件の内容
#[derive(Serialize)]
pub struct TodoItem {
    title: String,
    work: String,
    update: NaiveDate,
    start: NaiveDate,
    end: NaiveDate,
    done: bool,
}

/// todo リストの処理全般
pub struct Todo {}

impl Todo {
    pub fn new() -> Self {
        Self {}
    }

    pub fn get_todo_list(&self) -> Result<Vec<TodoItem>, String> {
        let now_date = Local::now().naive_local().date();
        let todo = TodoItem {
            title: "テスト 1".to_string(),
            work: "なにしようかな".to_string(),
            update: now_date,
            start: now_date + Days::new(1),
            end: now_date + Days::new(5),
            done: false,
        };
        let todo2 = TodoItem {
            title: "テスト 2".to_string(),
            work: "こんどはなにしよう。".to_string(),
            update: now_date + Days::new(1),
            start: now_date + Days::new(5),
            end: now_date + Days::new(20),
            done: true,
        };
        let ret = vec![todo, todo2];
        Ok(ret)
    }
```

```
44 }
```

2　フロントエンド　React 関係

2.1　index.html

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Tauri + React</title>
  </head>

  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

2.2 メイン CSS ファイル

```
.logo.vite:hover {
  filter: drop-shadow(0 0 2em #747bff);
}

.logo.react:hover {
  filter: drop-shadow(0 0 2em #61dafb);
}
:root {
  font-family: Inter, Avenir, Helvetica, Arial, sans-serif;
  font-size: 16px;
  line-height: 24px;
  font-weight: 400;

  color: #0f0f0f;
  background-color: #f6f6f6;

  font-synthesis: none;
  text-rendering: optimizeLegibility;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  -webkit-text-size-adjust: 100%;
}

.container {
  margin: 0;
  padding-top: 10vh;
  display: flex;
  flex-direction: column;
  justify-content: center;
  text-align: center;
}

.logo {
  height: 6em;
  padding: 1.5em;
  will-change: filter;
  transition: 0.75s;
}

.logo.tauri:hover {
  filter: drop-shadow(0 0 2em #24c8db);
}

```

```css
.row {
  display: flex;
  justify-content: center;
}

a {
  font-weight: 500;
  color: #646cff;
  text-decoration: inherit;
}

a:hover {
  color: #535bf2;
}

h1 {
  text-align: center;
}

input,
button {
  border-radius: 8px;
  border: 1px solid transparent;
  padding: 0.6em 1.2em;
  font-size: 1em;
  font-weight: 500;
  font-family: inherit;
  color: #0f0f0f;
  background-color: #ffffff;
  transition: border-color 0.25s;
  box-shadow: 0 2px 2px rgba(0, 0, 0, 0.2);
}

button {
  cursor: pointer;
}

button:hover {
  border-color: #396cd8;
}
button:active {
  border-color: #396cd8;
  background-color: #e8e8e8;
}

```

```css
input,
button {
  outline: none;
}

#greet-input {
  margin-right: 5px;
}

@media (prefers-color-scheme: dark) {
  :root {
    color: #f6f6f6;
    background-color: #2f2f2f;
  }

  a:hover {
    color: #24c8db;
  }

  input,
  button {
    color: #ffffff;
    background-color: #0f0f0f98;
  }
  button:active {
    background-color: #0f0f0f69;
  }
}
```

2.3 main.jsx

```jsx
import React from "react";
import ReactDOM from "react-dom/client";
import { UIProvider, extendTheme } from "@yamada-ui/react";
import App from "./App";
import { QueryClient, QueryClientProvider } from "@tanstack/react-query";

const semantics = {
    colors: {
        primary: "red.500",
    },
    colorSchemes: {
        primary: "blue",
    },
};

const globalStyle = {
    body: {
        bg: "#DCB879",
    },
}

const customTheme = extendTheme({ semantics, styles: {globalStyle} })();

const query_client = new QueryClient();

ReactDOM.createRoot(document.getElementById("root")).render(
    <React.StrictMode>
        <UIProvider theme={customTheme}>
            <QueryClientProvider client={query_client}>
                <App />
            </QueryClientProvider>
        </UIProvider>
    </React.StrictMode>,
);
```

2.4 アプリケーションメイン App.jsx

```jsx
//import reactLogo from "./assets/react.svg";
import "./App.css";
import { createBrowserRouter ,createRoutesFromElements, Route, RouterProvider, } from
↪  "react-router-dom";

import BasePage from "./BasePage.jsx";
import TodoList from "./TodoList.jsx";

export const routes = createBrowserRouter(
    createRoutesFromElements(
        <>
            <Route element={ <BasePage/> }>
                <Route path="/" element={<TodoList/>}/>
            </Route>
        </>
    ));

function App() {
    return (
        <RouterProvider router={routes}/>
    );
}
export default App;
```

2.5 全体のベースページ BasePage.jsx

```jsx
import { Outlet } from "react-router-dom";
import "./App.css";


function BasePage() {

    return (
        <>
            <h1> 猫 todo </h1>
            <Outlet/>
        </>
    );
}


export default BasePage;
```

2.6 todo リストの表示 TodoList.jsx

```jsx
import { useEffect, useState } from "react";
import { useQuery } from "@tanstack/react-query";
import { Grid, GridItem} from "@yamada-ui/react";
import { invoke } from "@tauri-apps/api/core";
import "./App.css";
import TodoItem from "./todoitem";

const get_todo_list = async () => invoke('get_todo_list') ;

function TodoList() {

    const { data: todos, isLoading: isTodoListLoading , isError, error} = useQuery({
        queryKey: ['get_todo_list'],
        queryFn: get_todo_list,
    });

    if (isTodoListLoading) {
        return ( <p> loading... </p>);
    }

    if (isError) {
        return ( <p> エラーだよ。{error}</p> );
    }

    console.log(todos);
    return (
        <>
            <h1>テスト</h1>
            <Grid templateColumns="repeat(4, 1fr)" gap="md">
                {todos?.map( todo_item => {
                    return (
                        <GridItem key={todo_item.title} w="full" rounded="md" bg="primary">
                            <TodoItem item={todo_item}/>
                        </GridItem>
                    )}
                )}
            </Grid>
        </>
    );
}


export default TodoList;
```

2.7 todo アイテム表示 todoitem.jsx

```jsx
// todo リストの各アイテム
import { SimpleGrid, GridItem } from "@yamada-ui/react";

export default function TodoItem({item}) {
    return (
        <>
            <SimpleGrid w="full" columns={{base: 2, md: 1}} gap="md">
                <GridItem> <p> {item.done? '済': '未'} </p></GridItem>

                <GridItem>
                    <div style={{textAlign:'right', fontSize:'0.7em'}}>
                        {item.update}
                    </div>
                </GridItem>
            </SimpleGrid>
            <p style={{textAlign:'center', fontSize:'1.1em'}}><strong>{item.title}</strong></p>
            <p>{item.work}</p>
            <div style={{fontSize:'0.9em'}}>
                <p>{item.start} ～ {item.end}</p>
            </div>
        </>
    );
}

```

3　データベース構成

3.1　テーブル生成スクリプト create_table.sql

```
# 猫todo 関係のすべての mariadb オブジェクトの生成

create database if not exists neko_todo;

use neko_todo;

create table if not exists users (
    name varchar(128) primary key,
    password varchar(61)
    );

create table if not exists todo (
    id int unsigned auto_increment primary key,
    user_name varchar(128) not null references users(name),
    title varchar(128) not null,
    work varchar(2048),
    update_date date not null,
    start_date date,
    end_date date,
    done bool
    ) ;

create table if not exists tag (
    name varchar(128) primary key
    );

create table if not exists todo_tag (
    todo_id int unsigned references todo(id),
    tag_name varchar(128) references tag(name),
    primary key(todo_id, tag_name)
    );

create table if not exists sessions (
    id varchar(40) primary key,
    user_name varchar(128) references users(name),
    expired datetime default date_add(current_timestamp, interval 48 hour)
    );

```