Calendar App

Written by Rust programming language

Rust Calendar App

Documentation

Version 1.0

08/19/2024

Table of Contents

1. Introduction

The **Rust Calendar App** is a simple application built in Rust that generates and displays a calendar for a given month and year. The application uses the `chrono` crate for date and time management, and the `slint` GUI toolkit to render the calendar UI. The app is designed to show the current month, along with days from the previous and next months to fill the grid.

Key Features:

- **Display Current Month:** Shows all the days of the current month.
- **Previous/Next Month Days:** Fills in days from the previous and next months to complete the calendar grid.
- **Dynamic Calendar:** Navigate to the next month dynamically with updated day calculations.
- **Weekday Labels:** Displays labels for the days of the week (Sunday to Saturday).

2. Structure

The project is organized as follows:

2.1 Module & Imports

- `chrono::prelude::*`: Provides date and time functionality.
- `datetimeutils`: Custom module for handling month-related utilities like days in a month and converting month indexes.
- `slint::{SharedString, VecModel}`: Used for the GUI components, where `SharedString` is used for strings in the UI, and `VecModel` is used to manage dynamic lists in the UI.
- `std::rc::Rc`: Rust's reference counting pointer, allowing multiple ownership of data.

2.2 Functions

2.2.1 Date and Time Functions

- `get_last_days_of_prev_month`: Determines how many days from the previous month should be shown in the current month's calendar grid.
- 'get_first_days_of_next_month`: Determines how many days from the next month should be shown after the current month
- `get_week_day`: Returns the day of the week for a given date.

2.2.2 Calendar logic

- `generate_month`: Calculates the number of days in a given month and year.
- `insert_days`: Inserts days into the calendar model

2.2.3 Main Calendar Workflow

- `load_calendar`: Manages the entire process of calculating and inserting days for the previous month, current month, and next month into the calendar model
- `run_calendar`: Updates the UI with the generated calendar for the selected month and year.

2.2.4 UI interaction

- `get_week_days`: Prepares and returns a list of weekday labels (Sunday to Saturday)
- Event Handlers: Handles user interactions, such as moving to the next month

3. Main Logic

The main logic revolves around generating and displaying the calendar for a given month and year. Below are the key steps.

3.1 Get the current Year, Month

- The app starts by obtaining the current date using the `chrono` crate.
- Get current year and month using `current_year()` and `current_month()`

3.2 Get Last Days of Previous Month to Be Shown in the Current UI

- Using `get_week_day` function, determine first weekday (1) of current month
  Expected: Sun ~ Sat
- Using `get_last_days_of_prev_month`, calculate how many days from the previous month should be shown. expected: 0 ~ 6
- Using `generate_month` function, get the number of the days of the previous month
  This is used to calculate the start day to be shown first in the UI (expected: 28, 29, 30, 31)
  e.g. 31 – 4 = 27. Finally, 27 will be showed first in the UI.
- Using `insert_days` function, insert the calculated days from the previous month into the calendar UI

3.3 Align the Days of Current Month with the day of the week

Using `insert_days` function with gotten number of days of the current month, fill the days to the current UI.

3.4 Get First Days of Next Month to Be Shown in the Current UI

- Using `generate_month` function, get total number of the days of current month.
- Using `get_week_day` function, determine last weekday (e,g, 31) of current month
  Expected: Sun ~ Sat
- Using `get_first_days_of_next_month`, calculate how many days from the next month should be shown. expected: 0 ~ 6
- Using `insert_days` function, insert the calculated days from the next month into the calendar UI