Project Documentation  
Objectively CRUDdy  
(The Sequel!)

Author: Mike LeBlanc

Date: March 15, 2024

Table of Contents

[Project Requirements 3](#_Toc161435998)

[Design Plans 4](#_Toc161435999)

[Command Design Pattern 4](#_Toc161436000)

[Principals 4](#_Toc161436001)

[Implementation 5](#_Toc161436002)

[ICommand 5](#_Toc161436003)

[Database Access 5](#_Toc161436004)

[Database Setup 6](#_Toc161436005)

[Class Diagram 6](#_Toc161436006)

[Testing 7](#_Toc161436007)

# Project Requirements

1. Make a tabbed or windowed Person-Data-Entry CRUD Application
   1. First name, Last name, and address
   2. Occupation and Hobbies Combo Boxes populated by our database.
   3. Face-Maker from previous Assignment
   4. Navigation between tabs
   5. Clear navigation of records
   6. Search by last name
2. CRUD
   1. Create new records in the DB.
   2. Read records from the DB.
   3. Update Records in the DB
   4. Delete records from the DB.

# Design Plans

## Command Design Pattern

The command pattern is a design pattern that enables developers to encapsulate requests or operations into separate objects, making it possible to modify the requests dynamically and execute them without the sender or receiver knowing anything about each other.

We execute this by connecting our Command Handler class to the function we expect to happen, when this command is called, the function can fire.

We take this a step further by adding a key binding that when pressed, calls the command handler.

This facilitates the steps needed to allow the same function to be called by both the menu selection and the hotkey.

## Principals

Through the course of this application, we have implemented various coding principals including Single Responsibility and Open-closed principals. Apart from the transparency needed to show differences between internal assignments, the code-behind from the XAML has a Single Responsibility, as well as the Faces class and Command Handler class.

These classes are Open-Closed because they are open for extension while being closed to modification. This forces other programmers to extend the functionality of what it does rather than change the core methods.

# Implementation

## ICommand

A screen shot of a computer program

Description automatically generated

With this ICommand class, we can accept a hotkey command and map it correctly to the desired outcome. We are then able to provide the user with clear and concise instructions on what that hotkey is, and what the hotkey does.

## Database Access

A screen shot of a computer

Description automatically generated

We’ve added an additional class for database access. This separates our business logic from our users and only shows the users what it is they request. In this case, we serve every use in the database to the user so they can see the list of people.

## Database Setup

A screenshot of a computer

Description automatically generated

We’ve gone for a very simple Database setup here, with having each table handle its own data, we’re able to achieve a high Normalization Factor (NF). This also makes it very easy to populate our Combo Boxes with the right data.

No need to build something complex when there is no need for it!

## Class Diagram

A screenshot of a computer

Description automatically generated

Our class diagram shows the simplicity of our class structure and overall application. This aids with ease of maintenance and bug fixing as there are not many other classes to trace errors back through.

# Testing

A yellow circle with white border

Description automatically generated

Figure 1: Application state upon start.

A yellow circle with black text

Description automatically generated

Figure 2: Menu icons with associated hotkeys shown.

A cartoon face with red hair

Description automatically generated

Figure 3: Random Face hotkey pressed. CTRL+R

A screenshot of a computer

Description automatically generated

Figure 4: Help menu option brings up Help Window

A screenshot of a computer

Description automatically generated

Figure 5: Starting point for table.

A screenshot of a computer

Description automatically generated

Figure 6: Selecting a row updates the table.

A screenshot of a computer

Description automatically generated

Figure 7: Which also updates our 4th tab, and ready to output to a file.

A screenshot of a computer

Description automatically generated

Figure 8: Proof that Occupations being pulled from database

A screenshot of a computer

Description automatically generated

Figure 9: Proof that hobbies are being pulled from Database Object

A screenshot of a computer

Description automatically generated

Figure 10: Final output of our program, ready to be put into a file!