Documentation Camp Sleepaway

Authors: Samuel Lööf, Simon Sörqvist, Adam Kumlin

Overview:

The **Program.cs** file in the Camp Sleepaway project acts as the central hub for the program. It orchestrates functionalities related to camper data, counselors, next of kin, and cabins. The code is structured to manage operations such as adding, editing, searching, and deleting records from the database. Below is a breakdown of the key components and operations within the program:

Initialization (IsFirstRun):

The **IsFirstRun()** method checks for a **"first_run.txt"** file in the current directory to determine if it's the program's initial execution. If it's the first run, example data is added (from our example data .csv files) to the database using **AddExampleDataToDb**.**AddAllData()**. The "**first_run.txt**" file is then updated to prevent the repeated addition of example data. If it's not the first run, the method displays "First time data already exists," ensuring the avoidance of redundant data in subsequent executions.

Main Menu (ShowMainMenu):

The **ShowMainMenu()** method presents a user-friendly console menu using the Spectre.Console library. Options include adding new objects, editing records, searching campers, viewing campers and next of kin, deleting objects, and exiting the program. The menu features dynamic prompts and clear console screens for an enhanced user interface.

Adding Objects:

Users can add new objects, including campers, counselors, next of kin, and cabins.
Object-specific input methods are called based on the user's selection, facilitating a structured and user-friendly approach to data entry. The code utilizes conditional statements to manage each object type's input process.

Editing Records:

Users can edit records for campers, counselors, next of kin, and cabins. The program guides users through selection and editing processes using dedicated methods for each object type.

Searching Campers:

The program allows users to search for specific campers based on user-defined criteria. This functionality aids in quickly retrieving camper information from the database.

Displaying Campers and NextOfKins:

Users can view a list of campers along with their corresponding next of kin. This feature provides a comprehensive overview of camper relationships.

Deleting Objects:

Users can delete records for campers, counselors, next of kin, and cabins. The code includes checks and balances, such as ensuring counselors can only be deleted if their campers have left.

Moving on, we have the **Person** class which is designed as a foundational parent class that other Entity Framework Core table classes within the Camp Sleepaway project inherit from. This class serves to abstract common properties shared by entities representing individuals.

The properties of the **Person** class include:

- **FirstName**: A string property with a maximum length of 50 characters. It is required, and an error message is displayed for an invalid first name.
- **LastName**: Another string property with a maximum length of 50 characters. Similar to the first name, it is required, and an error message is provided for an invalid last name.
- **PhoneNumber**: A string property representing a phone number. It is required, supports a maximum length of 16 characters, and accommodates a Swedish phone number format with white spaces. An error message is shown for an invalid phone number.

The **Person** class is not intended to represent a standalone table in Entity Framework Core; instead, it serves as a foundation for other table classes. These other EF table classes inherit from Person to leverage and share the common properties encapsulated within.

The **NextOfKin** class manages associations with campers, handling data, selection, and database interactions. Methods like **InputNextOfKinData** create new instances, and **SearchNextOfKin** finds based on cabin or counselor. Database operations are handled by **SaveToDb**, U**pdateRecordInDb**, and **DeleteFromDb**.

The Counselor class represents camp counselors, providing functionalities for data and database interactions. Methods like **InputCounselorData** and **ChooseCounselorMenu** handle creation and selection. Relationships with cabins and campers are managed, and operations like **GetAllFromDb** and **DeleteFromDb** ensure database integrity.

The Camper class encapsulates camper information, inheriting from the **Person** class. Methods like **InputCamperData** facilitate camper creation, and **ChooseCamperMenu** allows selection. **SearchCamper** finds campers based on **cabin** or **counselor**. **DisplayCampersAndNextOfKins** enhances visibility of camper-next of kin relationships.

Finally, the **Cabin** class represents physical cabins. Methods like **GenerateRandomCabinName** and **InputCabinData** handle name generation and creation. **ChooseCabinMenu** allows cabin selection, and **EditCabinMenu** permits modification, including camper association.

Together, these classes and methods form a concise system for streamlined camp data management.

Methods like **GetCounselorFromCabinId** and **GetCampersFromCabinId** facilitate the retrieval of associated counselor and camper information, respectively. Database interactions are managed through methods such as **SaveToDb**, **UpdateRecordInDb**, and **DeleteFromDb**, ensuring the consistent storage and manipulation of cabin records.

The **Helper** class provides a set of utility methods to facilitate common tasks within the camp management system. It includes the **IsLettersOnly**, **CalculateAge**, **IsPhoneNumberValid**, and the **FormatDate**

These utility methods help maintain code clarity and encapsulate common functionalities, promoting reusability and readability across different parts of the camp management system.

The **CampContext** class serves as the Entity Framework DbContext, facilitating interaction with the underlying database. It includes DbSet properties for the various entities representing tables in the database, namely Campers, Counselors, NextOfKins, and Cabins.

DbContext Configuration:

The **OnConfiguring** method is responsible for configuring the **DbContext** options. It reads the database connection options from the appsettings.json file using the **ConfigurationBuilder**. The connection string is built based on the configuration and used to connect to the SQL Server database. Note that the code related to logging is present but commented out for the final version.

Entity Relationships

The **OnModelCreating** method uses Fluent API to define the relationships between entities. It establishes:

A one-to-one relationship between Counselors and Cabins, where each counselor is associated with one cabin.

A one-to-many relationship between Campers and Cabins, where each cabin can have multiple campers.

Another one-to-many relationship between NextOfKins and Campers, indicating that each camper can have multiple next of kin.

These relationships are crucial for maintaining data integrity and organizing the database schema for our program.