Smart Stay Backend

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 SmartStay Namespace Reference

Namespaces

- namespace Common
- namespace Core
- namespace IO
- namespace Validation

    The `SmartStay.Validation` namespace contains classes and methods for validating data and enforcing business rules within the SmartStay application. These validations help ensure data integrity and compliance with application requirements.

5.2 SmartStay.Common Namespace Reference

Namespaces

- namespace Enums

    This namespace contains enumerations used within the SmartStay application.
- namespace Exceptions

    This namespace contains custom exceptions used within the SmartStay application.
- namespace Models

    This namespace contains common models used within the SmartStay application.

5.3 SmartStay.Common.Enums Namespace Reference

This namespace contains enumerations used within the SmartStay application.

Enumerations

- enum AccommodationType {
  None , Hotel , House , Apartment ,
  Villa , BedAndBreakfast , Hostel , Resort ,
  Cottage , Cabin , Guesthouse , Chalet ,
  Lodge }

    Enumeration representing different types of accommodations available for booking.
- enum CancellationResult {
  Success , ReservationNotFound , AccommodationNotFound , RoomNotFound ,
  Error }

    Enumeration representing the possible outcomes of a reservation cancellation attempt.
- enum PaymentMethod {
  Unchanged , None , PayPal , MultiBanco ,
  BankTransfer }

    Enumeration representing the possible payment methods available for transactions.
- enum PaymentResult {
  Success , InvalidAmount , AlreadyFullyPaid , AmountExceedsTotal ,
  InvalidPaymentMethod , Error }

    Enumeration representing the possible outcomes of a payment attempt.
- enum PaymentStatus {
  Unpaid , Pending , Completed , PartiallyPaid ,
  Rejected , Refunded , Cancelled }

    Enumerator representing payment status.
- enum RemoveAccommodationResult {
  Success , AccommodationNotFound , OwnerNotFound , AccommodationRemovalFailed ,
  AccommodationDisassociationFailed , Error }

    Enumeration representing the results of the accommodation removal process. This enum is used to indicate the outcome of the removal operation for an accommodation.
- enum ReservationStatus {
  Pending , CheckedIn , CheckedOut , Cancelled ,
  NoShow , Confirmed , Declined }

    Enumeration representing the current status of a reservation.
- enum RoomType {
  None , Single , Double , Twin ,
  Suite , Family , Studio , Deluxe ,
  Penthouse , Dormitory , Accessible , PresidentialSuite }

    Enumeration representing different types of rooms available within accommodations.
- enum UpdateAccommodationResult {
  Success , AccommodationNotFound , InvalidType , InvalidName ,
  InvalidAddress , Error }

    Enumeration representing the results of the accommodation update process. This enum is used to indicate the outcome of the update operation for an accommodation.
- enum UpdateClientResult {
  Success , ClientNotFound , InvalidFirstName , InvalidLastName ,
  InvalidEmail , InvalidPhoneNumber , InvalidAddress , InvalidPaymentMethod ,
  Error }

    Enumeration representing the results of the client update process. This enum is used to indicate the outcome of the update operation for a client.
- enum UpdateOwnerResult {
  Success , OwnerNotFound , InvalidFirstName , InvalidLastName ,
  InvalidEmail , InvalidPhoneNumber , InvalidAddress }

    Enum representing the result of an owner update operation.
- enum UpdateReservationResult {
  Success , ReservationNotFound , AccommodationNotFound , RoomNotFound ,
  RoomIsNull , DatesUnavailable , InvalidDates , Error }

Enumeration representing the results of the reservation update process. This enum is used to indicate the outcome of the update operation for a reservation.

5.3.1 Detailed Description

This namespace contains enumerations used within the SmartStay application.

<copyright file="AccommodationType.cs"> Copyright (c) 2024 Enrique Rodrigues. All Rights Reserved. </copyright> <file> This file contains the definition of the AccommodationType enumeration used in the SmartStay application, representing different accommodation types available for booking. </file> <author>Enrique Rodrigues</author> <date>07/10/2024</date>

<copyright file="CancellationResult.cs"> Copyright (c) 2024 Enrique Rodrigues. All Rights Reserved. </copyright> <file> This file contains the definition of the CancellationResult enumeration used in the SmartStay application, representing the different results of a reservation cancellation attempt. </file> <author>Enrique Rodrigues</author> <date>29/11/2024</date>

<copyright file="PaymentMethod.cs"> Copyright (c) 2024 Enrique Rodrigues. All Rights Reserved. </copyright> <file> This file contains the definition of the PaymentMethod enumeration used in the SmartStay application, representing different payment methods available for bookings and transactions. </file> <author>Enrique Rodrigues</author> <date>07/10/2024</date>

<copyright file="PaymentResult.cs"> Copyright (c) 2024 Enrique Rodrigues. All Rights Reserved. </copyright> <file> This file contains the definition of the PaymentResult enumeration used in the SmartStay application, representing the possible outcomes of a payment attempt. </file> <author>Enrique Rodrigues</author> <date>01/12/2024</date>

<copyright file="PaymentStatus.cs"> Copyright (c) 2024 Enrique Rodrigues. All Rights Reserved. </copyright> <file> This file contains the definition of the PaymentStatus enumeration used in the SmartStay application representing various payment status. </file> <author>Enrique Rodrigues</author> <date>07/10/2024</date>

<copyright file="RemoveAccommodationResult.cs"> Copyright (c) 2024 Enrique Rodrigues. All Rights Reserved. </copyright> <file> This file contains the definition of the RemoveAccommodationResult enumeration used in the SmartStay application, representing the different results of an accommodation removal attempt. </file> <author>Enrique Rodrigues</author> <date>29/11/2024</date>

<copyright file="ReservationStatus.cs"> Copyright (c) 2024 Enrique Rodrigues. All Rights Reserved. </copyright> <file> This file contains the definition of the ReservationStatus enumeration used in the SmartStay application, representing the different statuses a reservation can have. </file> <author>Enrique Rodrigues</author> <date>07/10/2024</date>

<copyright file="RoomType.cs"> Copyright (c) 2024 Enrique Rodrigues. All Rights Reserved. </copyright> <file> This file contains the definition of the RoomType enumeration used in the SmartStay application, representing different room types available within accommodations. </file> <author>Enrique Rodrigues</author> <date>27/11/2024</date>

<copyright file="UpdateAccommodationResult.cs"> Copyright (c) 2024 Enrique Rodrigues. All Rights Reserved. </copyright> <file> This file contains the definition of the UpdateAccommodationResult enumeration used in the SmartStay application, representing the different results of an accommodation update attempt. </file> <author>Enrique Rodrigues</author> <date>29/11/2024</date>

<copyright file="UpdateClientResult.cs"> Copyright (c) 2024 Enrique Rodrigues. All Rights Reserved. </copyright> <file> This file contains the definition of the UpdateClientResult enumeration used in the SmartStay application, representing the different results of a client update attempt. </file> <author>Enrique Rodrigues</author> <date>29/11/2024</date>

<copyright file="UpdateOwnerResult.cs"> Copyright (c) 2024 Enrique Rodrigues. All Rights Reserved. </copyright> <file> This file contains the definition of the UpdateOwnerResult enumeration used in the SmartStay application, representing the different results of an owner update attempt. </file> <author>Enrique Rodrigues</author> <date>29/11/2024</date>

<copyright file="UpdateReservationResult.cs"> Copyright (c) 2024 Enrique Rodrigues. All Rights Reserved. </copyright> <file> This file contains the definition of the UpdateReservationResult enumeration used in the SmartStay application, representing the different results of a reservation update attempt. </file> <author>Enrique Rodrigues</author> <date>29/11/2024</date>

5.3.2 Enumeration Type Documentation

5.3.2.1 AccommodationType

enum SmartStay.Common.Enums.AccommodationType

Enumeration representing different types of accommodations available for booking.

Enumerator

None	Indicates that the accommodation type is not defined. This is used when the accommodation type is not chosen.
Hotel	Represents a traditional hotel accommodation, typically offering private rooms and common amenities.
House	Represents a standalone house accommodation, ideal for private stays and larger groups.
Apartment	Represents an apartment accommodation, typically part of a larger building, offering self-contained living space.
Villa	Represents a villa accommodation, usually a larger, luxury residence often with a private pool and garden.
BedAndBreakfast	Represents a bed and breakfast accommodation, providing a private room with breakfast included, often in a home setting.
Hostel	Represents a hostel accommodation, often offering dormitory-style rooms and shared facilities, popular among budget travelers.
Resort	Represents a resort accommodation, typically offering all-inclusive services and multiple leisure amenities on-site.
Cottage	Represents a cottage accommodation, usually a small, cozy house in a rural or nature setting.
Cabin	Represents a cabin accommodation, typically a small, rustic structure often located in remote or forested areas.
Guesthouse	Represents a guesthouse accommodation, which offers a private room within a larger property, usually with shared amenities.
Chalet	Represents a chalet accommodation, usually a wooden house located in mountain regions, popular for ski vacations.
Lodge	Represents a lodge accommodation, typically found in nature destinations, offering basic to luxurious amenities.

Definition at line 19 of file AccommodationType.cs.

5.3.2.2 CancellationResult

enum SmartStay.Common.Enums.CancellationResult

Enumeration representing the possible outcomes of a reservation cancellation attempt.

Enumerator

Success	Indicates that the reservation cancellation was successful.
ReservationNotFound	Indicates that the reservation could not be cancelled because the reservation with the specified ID could not be found.
AccommodationNotFound	Indicates that the reservation could not be cancelled because the associated accommodation could not be found.
RoomNotFound	Indicates that the reservation could not be cancelled because the associated room could not be found.
Error	Indicates an unspecified error occurred during the cancellation process.

Definition at line 19 of file CancellationResult.cs.

5.3.2.3 PaymentMethod

enum SmartStay.Common.Enums.PaymentMethod

Enumeration representing the possible payment methods available for transactions.

Enumerator

Unchanged	Indicates that the payment method should not be changed. This is used when the payment method should remain the same.
None	No specific payment method selected; used as a default or placeholder value.
PayPal	Payment method through PayPal, allowing secure online payments.
MultiBanco	Payment method using MultiBanco, a popular Portuguese banking payment system.
BankTransfer	Payment method via bank transfer, where funds are transferred directly between bank accounts.

Definition at line 19 of file PaymentMethod.cs.

5.3.2.4 PaymentResult

enum SmartStay.Common.Enums.PaymentResult

Enumeration representing the possible outcomes of a payment attempt.

Enumerator

Success	Indicates that the payment was successful.
InvalidAmount	Indicates that the payment amount provided was invalid (e.g., less than or equal to zero).
AlreadyFullyPaid	Indicates that the reservation is already fully paid.
AmountExceedsTotal	Indicates that the payment amount exceeds the total cost of the reservation.
InvalidPaymentMethod	Indicates that the provided payment method is invalid.
Error	Indicates an unspecified error occurred during the payment process.

Definition at line 19 of file PaymentResult.cs.

5.3.2.5 PaymentStatus

enum SmartStay.Common.Enums.PaymentStatus

Enumerator representing payment status.

Enumerator

Unpaid	Payment has not been made yet.
Pending	Payment has been initiated but not yet completed (e.g., pending in processing).

Enumerator

Completed	Payment has been completed successfully.
PartiallyPaid	Payment was partially completed; more payments are expected.
Rejected	Payment was rejected, usually by the payment processor.
Refunded	Payment was refunded to the client.
Cancelled	Payment has been cancelled, typically by the client or system.

Definition at line 19 of file PaymentStatus.cs.

5.3.2.6 RemoveAccommodationResult

enum SmartStay.Common.Enums.RemoveAccommodationResult

Enumeration representing the results of the accommodation removal process. This enum is used to indicate the outcome of the removal operation for an accommodation.

Enumerator

Success	Indicates that the accommodation was successfully removed.
AccommodationNotFound	Indicates that the accommodation with the specified ID could not be found.
OwnerNotFound	Indicates that the owner associated with the accommodation could not be found.
AccommodationRemovalFailed	Indicates that the accommodation could not be removed from the system.
AccommodationDisassociationFailed	Indicates that the accommodation could not be disassociated from the owner.
Error	Indicates that an unknown error occurred during the removal process.

Definition at line 20 of file RemoveAccommodationResult.cs.

5.3.2.7 ReservationStatus

enum SmartStay.Common.Enums.ReservationStatus

Enumeration representing the current status of a reservation.

Enumerator

Pending	Reservation has been made but the client has not yet checked in.
CheckedIn	Client has checked in to the accommodation.
CheckedOut	Client has checked out from the accommodation.
Cancelled	Reservation was cancelled before the client checked in.
NoShow	Client did not show up for the reservation.
Confirmed	Reservation has been confirmed, but the client has not yet checked in.
Declined	Reservation was declined or denied due to some issue (e.g., payment failure, overbooked, etc.).

Definition at line 19 of file ReservationStatus.cs.

5.3.2.8 RoomType

enum SmartStay.Common.Enums.RoomType

Enumeration representing different types of rooms available within accommodations.

Enumerator

None	Indicates that the room type is not defined. This is used when the room type is not chosen.
Single	Represents a single room, typically designed for one occupant with a single bed.
Double	Represents a double room, typically designed for two occupants with a double bed or two single beds.
Twin	Represents a twin room, featuring two separate single beds for two occupants.
Suite	Represents a suite, offering a more spacious and luxurious setup, often with separate living and sleeping areas.
Family	Represents a family room, designed to accommodate larger groups or families, often with multiple beds.
Studio	Represents a studio room, typically featuring an open-plan design with combined sleeping, living, and kitchenette areas.
Deluxe	Represents a deluxe room, offering premium amenities and a more luxurious experience compared to standard rooms.
Penthouse	Represents a penthouse room, usually located on the top floor with luxurious features and expansive views.
Dormitory	Represents a dormitory-style room, typically featuring multiple beds in a shared space, common in hostels.
Accessible	Represents an accessible room, specifically designed for guests with disabilities, ensuring barrier-free access and amenities.
PresidentialSuite	Represents a presidential suite, offering the highest level of luxury and space within an accommodation, often with exclusive services.

Definition at line 19 of file RoomType.cs.

5.3.2.9 UpdateAccommodationResult

enum SmartStay.Common.Enums.UpdateAccommodationResult

Enumeration representing the results of the accommodation update process. This enum is used to indicate the outcome of the update operation for an accommodation.

Enumerator

Success	Indicates that the accommodation was successfully updated.
AccommodationNotFound	Indicates that the accommodation with the specified ID could not be found.
InvalidType	Indicates that the provided accommodation type is invalid.
InvalidName	Indicates that the provided accommodation name is invalid.
InvalidAddress	Indicates that the provided accommodation address is invalid.
Error	Indicates that an unknown error occurred during the accommodation update process.

Definition at line 20 of file UpdateAccommodationResult.cs.

5.3.2.10 UpdateClientResult

enum SmartStay.Common.Enums.UpdateClientResult

Enumeration representing the results of the client update process. This enum is used to indicate the outcome of the update operation for a client.

Enumerator

Success	Indicates that the client was successfully updated.
ClientNotFound	Indicates that the client with the specified ID could not be found.
InvalidFirstName	Indicates that the provided first name is invalid.
InvalidLastName	Indicates that the provided last name is invalid.
InvalidEmail	Indicates that the provided email address is invalid.
InvalidPhoneNumber	Indicates that the provided phone number is invalid.
InvalidAddress	Indicates that the provided address is invalid.
InvalidPaymentMethod	Indicates that the provided payment method is invalid.
Error	Indicates that an unknown error occurred during the update process.

Definition at line 20 of file UpdateClientResult.cs.

5.3.2.11 UpdateOwnerResult

enum SmartStay.Common.Enums.UpdateOwnerResult

Enum representing the result of an owner update operation.

Enumerator

Success	The operation was successful.
OwnerNotFound	The owner with the specified ID was not found.
InvalidFirstName	The first name provided is invalid.
InvalidLastName	The last name provided is invalid.
InvalidEmail	The email provided is invalid.
InvalidPhoneNumber	The phone number provided is invalid.
InvalidAddress	The address provided is invalid.

Definition at line 19 of file UpdateOwnerResult.cs.

5.3.2.12 UpdateReservationResult

enum SmartStay.Common.Enums.UpdateReservationResult

Enumeration representing the results of the reservation update process. This enum is used to indicate the outcome of the update operation for a reservation.

Enumerator

Success	Indicates that the reservation was successfully updated.
ReservationNotFound	Indicates that the reservation with the specified ID could not be found.
AccommodationNotFound	Indicates that the accommodation with the specified ID could not be found.
RoomNotFound	Indicates that the room associated with the reservation could not be found.
RoomIsNull	Indicates that the room found was null.
DatesUnavailable	Indicates that the new dates for the reservation are unavailable.
InvalidDates	Indicates that the given dates are not valid.
Error	Indicates that an unknown error occurred during the reservation update process.

Definition at line 20 of file UpdateReservationResult.cs.

5.4 SmartStay.Common.Exceptions Namespace Reference

This namespace contains custom exceptions used within the SmartStay application.

Data Structures

- class AccommodationCreationException

  Represents an error that occurs during the accommodation creation process in the SmartStay application. This exception is thrown when there is an issue with validating or processing the accommodation's data.

- class AddAccommodationSystemException

  Represents an error that occurs when an accommodation cannot be added to the system. This exception is thrown when there is an issue during the addition process, such as conflicts, system-level errors, or other reasons preventing the accommodation from being added.

- class ClientCreationException

  Represents an error that occurs during the client creation process in the SmartStay application. This exception is thrown when there is an issue with validating or processing the client's data.

- class EntityNotFoundException

  Represents an error that occurs when an entity is not found in the system. This exception is thrown when an operation cannot proceed because the specified entity (e.g., Reservation, Room) does not exist in the system.

- class OwnerAddAccommodationException

  Represents an error that occurs when an accommodation cannot be added to an owner's list of accommodations. This exception is thrown when there is an issue with the adding process, such as validation failures, conflicts, or other reasons why the accommodation cannot be associated with the owner.

- class OwnerCreationException

  Represents an error that occurs during the owner creation process in the SmartStay application. This exception is thrown when there is an issue with validating or processing the owner's data.

- class ReservationCreationException

  Represents an error that occurs during the reservation creation process in the SmartStay application. This exception is thrown when there is an issue with validating or processing the reservation's data, such as invalid dates, cost calculations, or availability.

- class TotalCostException

  Represents an error that occurs during the calculation or validation of the total cost in the SmartStay application. This exception is thrown when there is an issue with calculating the total cost of a reservation, such as invalid dates or incorrect cost calculations.

5.4.1 Detailed Description

This namespace contains custom exceptions used within the SmartStay application.

<copyright file="AccommodationCreationException.cs"> Copyright (c) 2024 Enrique Rodrigues. All Rights Reserved. </copyright> <file> This file contains the definition of the AccommodationCreationException class used in the SmartStay application to handle errors related to accommodation creation. </file> <author>Enrique Rodrigues</author> <date>01/12/2024</date>

<copyright file="AddAccommodationSystemException.cs"> Copyright (c) 2024 Enrique Rodrigues. All Rights Reserved. </copyright> <file> This file contains the definition of the AddAccommodationSystemException class used in the SmartStay application to handle errors related to adding an accommodation to the system. </file> <author>Enrique Rodrigues</author> <date>01/12/2024</date>

<copyright file="ClientCreationException.cs"> Copyright (c) 2024 Enrique Rodrigues. All Rights Reserved. </copyright> <file> This file contains the definition of the ClientCreationException class used in the SmartStay application to handle errors related to client creation. </file> <author>Enrique Rodrigues</author> <date>01/12/2024</date>

<copyright file="EntityNotFoundException.cs"> Copyright (c) 2024 Enrique Rodrigues. All Rights Reserved. </copyright> <file> This file contains the definition of the EntityNotFoundException class used in the SmartStay application to handle errors related to missing entities. </file> <author>Enrique Rodrigues</author> <date>01/12/2024</date>

<copyright file="OwnerAddAccommodationException.cs"> Copyright (c) 2024 Enrique Rodrigues. All Rights Reserved. </copyright> <file> This file contains the definition of the OwnerAddAccommodationException class used in the SmartStay application to handle errors related to adding an accommodation to an owner's list of accommodations. </file> <author>Enrique Rodrigues</author> <date>01/12/2024</date>

<copyright file="OwnerCreationException.cs"> Copyright (c) 2024 Enrique Rodrigues. All Rights Reserved. </copyright> <file> This file contains the definition of the OwnerCreationException class used in the SmartStay application to handle errors related to owner creation. </file> <author>Enrique Rodrigues</author> <date>01/12/2024</date>

<copyright file="ReservationCreationException.cs"> Copyright (c) 2024 Enrique Rodrigues. All Rights Reserved. </copyright> <file> This file contains the definition of the ReservationCreationException class used in the SmartStay application to handle errors related to reservation creation. </file> <author>Enrique Rodrigues</author> <date>01/12/2024</date>

<copyright file="TotalCostException.cs"> Copyright (c) 2024 Enrique Rodrigues. All Rights Reserved. </copyright> <file> This file contains the definition of the TotalCostException class used in the SmartStay application to handle errors related to the calculation or validation of the total cost. </file> <author>Enrique Rodrigues</author> <date>01/12/2024</date>

5.5 SmartStay.Common.Models Namespace Reference

This namespace contains common models used within the SmartStay application.

Data Structures

- class ImportResult

  Represents the result of an accommodation import operation, summarizing the outcome of the process.

5.5.1 Detailed Description

This namespace contains common models used within the SmartStay application.

<copyright file="ImportResult.cs"> Copyright (c) 2024 Enrique Rodrigues. All Rights Reserved. </copyright> <file> This file contains the definition of the ImportResult class used in the SmartStay application to summarize the outcome of an import operation for accommodations. </file> <author>Enrique Rodrigues</author> <date>01/12/2024</date>

5.6 SmartStay.Core Namespace Reference

Namespaces

- namespace Models

  The `SmartStay.Core.Models` *namespace contains the primary data models used within the* SmartStay *application. These models represent core entities and structures essential for managing application data.*
- namespace Repositories

  The `SmartStay.Repositories` *namespace provides data access layers for retrieving and storing application data. It contains repositories that manage database interactions for various entities within the* SmartStay *application.*
- namespace Services

  The `Core.Services` *namespace contains service classes that implement business logic for the* SmartStay *application. These services coordinate actions between repositories and models to fulfill application requirements.*
- namespace Utilities

  The `SmartStay.Utilities` *namespace provides helper functions and utility classes used throughout the* SmartStay *application. These utilities support common operations and enhance reusability across different components of the application.*

5.7 SmartStay.Core.Models Namespace Reference

The `SmartStay.Core.Models` namespace contains the primary data models used within the SmartStay application. These models represent core entities and structures essential for managing application data.

Namespaces

- namespace Interfaces

  This namespace contains interfaces used within the SmartStay *application.*

Data Structures

- class Accommodation

  Defines the Accommodation class, which encapsulates the details of an accommodation, such as its type, name, address, nightly price, and availability status. This class provides methods to update availability and calculate total cost.
- class Client

  Defines the Client class, which encapsulates the details of a client including personal information such as first name, last name, email address, phone number, residential address, and preferred payment method. This class validates the provided data upon creation or when modifying specific properties, ensuring that all data is consistent and correct.
- class Owner

Defines the Owner class, which encapsulates the details of an accommodation owner, including personal information such as first name, last name, email address, phone number, and a list of owned accommodations. This class validates the provided data upon creation or when modifying specific properties, ensuring that all data is consistent and correct.

- class Payment

  Represents a payment made in the SmartStay system, with details such as amount, date, method, and status.

- class Reservation

  Defines the Reservation class, which encapsulates reservation details such as client ID, accommodation type, dates, and payment information. This class ensures data consistency by validating input parameters upon creation or when modifying specific properties.

- class Room

  Defines the Room class, which encapsulates the details of a room within an accommodation, such as its type, price per night, and reservation details. This class provides methods for updating availability and calculating the total cost for a stay.

5.7.1 Detailed Description

The `SmartStay.Core.Models` namespace contains the primary data models used within the SmartStay application. These models represent core entities and structures essential for managing application data.

5.8 SmartStay.Core.Models.Interfaces Namespace Reference

This namespace contains interfaces used within the SmartStay application.

Data Structures

- interface IManageableEntity

  Defines the IManageableEntity$<T>$ interface for managing a collection of entities of type T . This interface standardizes methods for adding, removing, importing, and exporting entities.

5.8.1 Detailed Description

This namespace contains interfaces used within the SmartStay application.

$<$copyright file="ManageableEntity.cs"$>$ Copyright (c) 2024 All Rights Reserved $</$copyright$>$ $<$file$>$ This file contains the definition of the IManageableEntity interface, which provides a standard structure for managing collections of entities within the SmartStay application.

This interface can be implemented by any collection class to provide a consistent API for managing entities, facilitating code reuse and standardization across different types of entity collections (e.g., Clients, Reservations, Accommodations). $</$file$>$ $<$author$>$Enrique Rodrigues$</$author$>$ $<$date$>$11/11/2024$</$date$>$

5.9 SmartStay.Core.Repositories Namespace Reference

The `SmartStay.Repositories` namespace provides data access layers for retrieving and storing application data. It contains repositories that manage database interactions for various entities within the SmartStay application.

Data Structures

- class Accommodations

    Represents a collection of Accommodation objects, managed in a dictionary for fast lookup by accommodation ID.
- class Clients

    Represents a collection of Client objects, managed in a dictionary for fast lookup by client ID. Implements the IManageableEntity<Client> interface for standardized management.
- class Owners

    Represents a collection of Owner objects, managed in a dictionary for fast lookup by owner ID. Implements the IManageableEntity<Owner> interface for standardized management.
- class Reservations

    Represents a collection of Reservation objects, managed in a dictionary for fast lookup by reservation ID.

5.9.1 Detailed Description

The `SmartStay.Repositories` namespace provides data access layers for retrieving and storing application data. It contains repositories that manage database interactions for various entities within the SmartStay application.

5.10 SmartStay.Core.Services Namespace Reference

The `Core.Services` namespace contains service classes that implement business logic for the SmartStay application. These services coordinate actions between repositories and models to fulfill application requirements.

Data Structures

- class BookingManager

    Provides a facade for managing clients, reservations, and accommodations in the booking system. This class centralizes all operations for adding, removing, importing, and exporting data for these entities. It interacts with internal repositories to simplify the main API and ensure a standardized approach.

5.10.1 Detailed Description

The `Core.Services` namespace contains service classes that implement business logic for the SmartStay application. These services coordinate actions between repositories and models to fulfill application requirements.

5.11 SmartStay.Core.Utilities Namespace Reference

The `SmartStay.Utilities` namespace provides helper functions and utility classes used throughout the SmartStay application. These utilities support common operations and enhance reusability across different components of the application.

Data Structures

- class DateRange

    Represents a range of dates with a start and end date. Implements IComparable<DateRange> to allow sorting and comparisons.
- class **JsonHelper**

    Provides static methods to serialize and deserialize objects to and from JSON format.

5.11.1 Detailed Description

The `SmartStay.Utilities` namespace provides helper functions and utility classes used throughout the SmartStay application. These utilities support common operations and enhance reusability across different components of the application.

<copyright file="DateRange.cs"> Copyright (c) 2024 All Rights Reserved </copyright> <file> This file contains the definition of the DateRange class, which represents a range of dates. </file> <author>Enrique Rodrigues</author> <date>21/11/2024</date>

5.12 SmartStay.IO Namespace Reference

Namespaces

- namespace Extensions

    This namespace contains File Extension functions, such as ensuring a directory exists, used within the SmartStay application.
- namespace FileOperations

    Provides file handling operations such as reading from and writing to files.

5.13 SmartStay.IO.Extensions Namespace Reference

This namespace contains File Extension functions, such as ensuring a directory exists, used within the SmartStay application.

Data Structures

- class **FileExtensions**

    Provides extension methods for file-related operations.

5.13.1 Detailed Description

This namespace contains File Extension functions, such as ensuring a directory exists, used within the SmartStay application.

<copyright file="FileExtensions.cs"> Copyright (c) 2024 All Rights Reserved </copyright> <file> This file contains the implementation of extension methods for file-related operations. </file> <author>Enrique Rodrigues</author> <date>20/11/2024</date>

5.14 SmartStay.IO.FileOperations Namespace Reference

Provides file handling operations such as reading from and writing to files.

Data Structures

- class **FileHandler**

  Provides static methods for file operations such as reading from and writing to files.

- class **PathValidator**

  Provides utility methods for validating file paths and extensions.

5.14.1 Detailed Description

Provides file handling operations such as reading from and writing to files.

This namespace contains utility methods for file operations used within the SmartStay application.

<copyright file="FileHandler.cs"> Copyright (c) 2024 All Rights Reserved </copyright> <file> This file contains utility methods for reading from and writing to files, including directory management for non-existing paths. </file>

<author>Enrique Rodrigues</author> <date>20/11/2024</date>

This namespace contains utility methods for file operations used within the SmartStay application.

<copyright file="PathValidator.cs"> Copyright (c) 2024 All Rights Reserved </copyright> <file> This file contains utility methods for validating file paths and file extensions. </file> <author>Enrique Rodrigues</author> <date>20/11/2024</date>

5.15 SmartStay.Validation Namespace Reference

The `SmartStay.Validation` namespace contains classes and methods for validating data and enforcing business rules within the SmartStay application. These validations help ensure data integrity and compliance with application requirements.

Namespaces

- namespace Resources
- namespace Validators

  The `SmartStay.Validation.Validators` namespace contains classes and methods for validating various types of input data in the SmartStay application. These validations enforce data integrity and compliance with application-specific requirements.

Data Structures

- class **ValidationErrorMessages**

  Provides a mechanism to retrieve localized validation error messages based on the given ValidationErrorCode. Messages are retrieved from resource files depending on the current culture of the application.

- class ValidationException

  Represents an exception that is thrown when a validation error occurs. This exception contains an error code that corresponds to a specific validation failure. The error message is retrieved from the resource files based on the error code and the current culture.

Enumerations

- enum ValidationErrorCode {
  InvalidName = 1001 , InvalidEmail = 1002 , InvalidPhoneNumber = 1003 , InvalidAddress = 1004 ,
  InvalidPaymentMethod = 1005 , InvalidAccommodationType = 1006 , InvalidId = 1007 , InvalidDateRange =
  1008 ,
  InvalidDate = 1009 , InvalidTotalCost = 1010 , InvalidPaymentValue = 1011 , InvalidReservationStatus = 1012 ,
  InvalidAccommodationName = 1013 , InvalidPrice = 1014 , InvalidPaymentStatus = 1015 , InvalidAvailabilityStatus
  = 1016 ,
  InvalidRoomType = 1017 }

  Defines error codes for validation failures within the SmartStay application.

5.15.1   Detailed Description

The `SmartStay.Validation` namespace contains classes and methods for validating data and enforcing business rules within the SmartStay application. These validations help ensure data integrity and compliance with application requirements.

<copyright file="ValidationErrorCodes.cs"> Copyright (c) 2024 All Rights Reserved </copyright> <file> This file contains the definition of the ValidationErrorCode enum, which represents specific error codes related to validation failures within the SmartStay application. </file> <author>Enrique Rodrigues</author> <date>09/11/2024</date>

<copyright file="ValidationException.cs"> Copyright (c) 2024 All Rights Reserved </copyright> <file> This file defines the ValidationException class, which is a custom exception used to represent validation errors in the SmartStay application. This exception includes an error code and a localized error message based on the validation failure. </file> <author>Enrique Rodrigues</author> <date>19/11/2024</date>

5.15.2   Enumeration Type Documentation

5.15.2.1   ValidationErrorCode

enum `SmartStay.Validation.ValidationErrorCode`

Defines error codes for validation failures within the SmartStay application.

Enumerator

InvalidName	Error code indicating that the provided name is invalid.
InvalidEmail	Error code indicating that the provided email address is invalid.
InvalidPhoneNumber	Error code indicating that the provided phone number is invalid.
InvalidAddress	Error code indicating that the provided address is invalid.
InvalidPaymentMethod	Error code indicating that the provided payment method is invalid.
InvalidAccommodationType	Error code indicating that the provided accommodation type is invalid.
InvalidId	Error code indicating that the provided ID is invalid.
InvalidDateRange	Error code indicating that the provided date range is invalid, typically when the check-in date is later than or equal to the check-out date.
InvalidDate	Error code indicating that the provided date is invalid, typically when the date is in the past or does not meet the expected criteria.
InvalidTotalCost	Error code indicating that the total cost provided is invalid, usually if it is a negative value.

Enumerator

InvalidPaymentValue	Error code indicating that the provided payment value is invalid, such as when it is negative or exceeds the total cost.
InvalidReservationStatus	Error code indicating that the provided reservation status is invalid, typically if it does not match any defined status in the ReservationStatus enumeration.
InvalidAccommodationName	Error code indicating that the provided accommodation name is invalid.
InvalidPrice	Error code indicating that the provided price is invalid.
InvalidPaymentStatus	Error code indicating that the provided payment status is invalid.
InvalidAvailabilityStatus	Error code indicating that the provided availability status is invalid.
InvalidRoomType	Error code indicating that the provided room type is invalid.

Definition at line 21 of file ValidationErrorCodes.cs.

5.16 SmartStay.Validation.Resources Namespace Reference

Data Structures

- class **ValidationMessages**

  A strongly-typed resource class, for looking up localized strings, etc.

5.17 SmartStay.Validation.Validators Namespace Reference

The `SmartStay.Validation.Validators` namespace contains classes and methods for validating various types of input data in the SmartStay application. These validations enforce data integrity and compliance with application-specific requirements.

Data Structures

- class **AccommodationValidator**

  Defines the AccommodationValidator class, which provides functionality for validating accommodation types in the SmartStay application.

- class **AddressValidator**

  Defines the AddressValidator class, which provides functionality for validating addresses used in the SmartStay application.

- class **ClientValidator**

  Defines the ClientValidator class, which provides functionality for validating client-related data in the SmartStay application.

- class **DateValidator**

  Defines the DateValidator class, which provides functionality for validating dates related to reservations, ensuring they adhere to application-specific rules.

- class **EmailValidator**

  Defines the EmailValidator class, which provides functionality for validating email addresses within the SmartStay application.

- class **NameValidator**

  Defines the NameValidator class, which provides functionality for validating various types of names within the SmartStay application.

- class **OwnerValidator**

Defines the OwnerValidator class, which provides functionality for validating owner-related data in the SmartStay application.

- class **PaymentValidator**

  The `SmartStay.Validation.Validators` namespace contains classes and methods for validating various types of input data in the SmartStay application. These validations enforce data integrity and compliance with application-specific requirements.

- class **PhoneNumberValidator**

  The `SmartStay.Validation.Validators` namespace contains classes and methods for validating various types of input data in the SmartStay application. These validations enforce data integrity and compliance with application-specific requirements.

- class **ReservationValidator**

  Provides validation methods for reservation-related data.

- class **RoomValidator**

  The `RoomValidator` class provides methods for validating room-related data within the SmartStay application. It ensures integrity and compliance with business rules.

5.17.1 Detailed Description

The `SmartStay.Validation.Validators` namespace contains classes and methods for validating various types of input data in the SmartStay application. These validations enforce data integrity and compliance with application-specific requirements.

The `SmartStay.Validation.Validators` namespace provides classes and methods dedicated to validating various aspects of the SmartStay application. These validations ensure that input data adheres to business requirements and standards.

<copyright file="AddressValidator.cs"> Copyright (c) 2024 All Rights Reserved </copyright> <file> This file contains the implementation of the AddressValidator class, which provides methods for validating address-related data in the SmartStay application. </file> <author>Enrique Rodrigues</author> <date>19/11/2024</date>

<copyright file="ClientValidator.cs"> Copyright (c) 2024 All Rights Reserved </copyright> <file> This file contains the implementation of the ClientValidator class, which provides validation methods for client-related data in the SmartStay application. </file> <author>Enrique Rodrigues</author> <date>19/11/2024</date>

<copyright file="DateValidator.cs"> Copyright (c) 2024 All Rights Reserved </copyright> <file> This file contains the implementation of the DateValidator class, which provides methods for validating dates used in the SmartStay application, such as check-in and check-out dates. </file> <author>Enrique Rodrigues</author> <date>19/11/2024</date>

<copyright file="NameValidator.cs"> Copyright (c) 2024 All Rights Reserved </copyright> <file> This file contains the implementation of the NameValidator class, which provides methods for validating names, including user names and accommodation names, ensuring they meet the application's requirements. </file> <author>Enrique Rodrigues</author> <date>19/11/2024</date>

<copyright file="OwnerValidator.cs"> Copyright (c) 2024 All Rights Reserved </copyright> <file> This file contains the implementation of the OwnerValidator class, which provides validation methods for owner-related data in the SmartStay application. </file> <author>Enrique Rodrigues</author> <date>27/11/2024</date>

Chapter 6

Data Structure Documentation

6.1 SmartStay.Core.Models.Accommodation Class Reference

Defines the Accommodation class, which encapsulates the details of an accommodation, such as its type, name, address, nightly price, and availability status. This class provides methods to update availability and calculate total cost.

Public Member Functions

- Accommodation ()

    Initializes a new instance of the Accommodation class.
- Accommodation (int ownerId, AccommodationType type, string name, string address)

    Initializes a new instance of the Accommodation class with the specified details: type, name, address, and price per night.
- Accommodation (int id, int ownerId, AccommodationType type, string name, string address, List< Room > rooms)

    Constructor to initialize a new Accommodation with all details, including a manually specified ID, owner ID, type, name, address, and list of rooms. **This constructor should be avoided in normal cases** *as it allows manual assignment of the accommodation ID, which can lead to conflicts and issues with ID uniqueness. The system is designed to automatically handle unique ID assignment, and using other constructors is recommended for creating accommodation objects to ensure proper handling of IDs.*
    This constructor is marked with [JsonConstructor] so it will be used for JSON deserialization purposes, but it should not be used when creating new accommodation objects manually.
- Room? FindRoomById (int roomId)

    Finds and returns a room from the accommodation by its room ID.
- bool AddRoom (Room room)

    Adds a new room to the accommodation.
- bool DeleteRoom (int roomId)

    Deletes a room from the accommodation's room list.
- Accommodation Clone ()

    Creates a deep copy of the current Accommodation instance.
- override string ToString ()

    Overridden ToString method to provide accommodation information in a readable JSON format.

Properties

- static int LastAssignedId [get, set]

    Public getter and setter for the last assigned ID.
- int Id [get]

    Public getter for the accommodation ID.
- int OwnerId [get, set]

    Public getter and setter for the Owner ID.
- AccommodationType Type [get, set]

    Public getter and setter for the Type.
- string Name [get, set]

    Public getter and setter for the Name.
- string Address [get, set]

    Public getter and setter for the Address.
- List< Room > Rooms [get]

    Gets a deep copy of the list of rooms in the accommodation.

6.1.1 Detailed Description

Defines the Accommodation class, which encapsulates the details of an accommodation, such as its type, name, address, nightly price, and availability status. This class provides methods to update availability and calculate total cost.

Definition at line 31 of file Accommodation.cs.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Accommodation() [1/3]

```
SmartStay.Core.Models.Accommodation.Accommodation ( ) [inline]
```

Initializes a new instance of the Accommodation class.

This constructor is required for Protobuf-net serialization/deserialization.

It should **not** be used directly in normal application code. Instead, use the constructor with parameters for creating instances of Accommodation.

Definition at line 98 of file Accommodation.cs.

6.1.2.2 Accommodation() [2/3]

```
SmartStay.Core.Models.Accommodation.Accommodation (
            int ownerId,
            AccommodationType type,
            string name,
            string address ) [inline]
```

Initializes a new instance of the Accommodation class with the specified details: type, name, address, and price per night.

Parameters

ownerId	The ID of the owner of the accommodation.
type	The type of the accommodation (e.g., Hotel, House).
name	The name of the accommodation.
address	The address of the accommodation.
pricePerNight	The nightly price of the accommodation.

Exceptions

ValidationException	Thrown if any of the provided parameters fail validation:
ValidationException	Thrown if the accommodation type is invalid.
ValidationException	Thrown if the accommodation name is invalid.
ValidationException	Thrown if the address is invalid.
ValidationException	Thrown if the price per night is invalid.

The constructor validates the provided parameters using the Validator class before initializing the properties. If any validation fails, a ValidationException is thrown with the appropriate error code.

Definition at line 123 of file Accommodation.cs.

6.1.2.3 Accommodation() [3/3]

```
SmartStay.Core.Models.Accommodation.Accommodation (
            int id,
            int ownerId,
            AccommodationType type,
            string name,
            string address,
            List< Room > rooms )  [inline]
```

Constructor to initialize a new Accommodation with all details, including a manually specified ID, owner ID, type, name, address, and list of rooms. **This constructor should be avoided in normal cases** as it allows manual assignment of the accommodation ID, which can lead to conflicts and issues with ID uniqueness. The system is designed to automatically handle unique ID assignment, and using other constructors is recommended for creating accommodation objects to ensure proper handling of IDs.
This constructor is marked with [JsonConstructor] so it will be used for JSON deserialization purposes, but it should not be used when creating new accommodation objects manually.

Parameters

id	The manually specified ID of the accommodation. This should not be used under normal circumstances as the system handles ID assignment automatically.
owner⟵ Id	The ID of the owner of the accommodation.
type	The type of accommodation (e.g., hotel, apartment, etc.).
name	The name of the accommodation.
address	The residential address of the accommodation.
rooms	The list of rooms available in the accommodation.

Definition at line 157 of file Accommodation.cs.

6.1.3 Member Function Documentation

6.1.3.1 AddRoom()

```
bool SmartStay.Core.Models.Accommodation.AddRoom (
             Room room )  [inline]
```

Adds a new room to the accommodation.

Parameters

room	The Room object to be added to the accommodation's room list.

Returns

true if the room was added successfully; otherwise, false.

Definition at line 249 of file Accommodation.cs.

6.1.3.2 Clone()

```
Accommodation SmartStay.Core.Models.Accommodation.Clone ( )  [inline]
```

Creates a deep copy of the current Accommodation instance.

Returns

A new Accommodation instance with identical data to the current instance.

Definition at line 320 of file Accommodation.cs.

6.1.3.3 DeleteRoom()

```
bool SmartStay.Core.Models.Accommodation.DeleteRoom (
             int roomId )  [inline]
```

Deletes a room from the accommodation's room list.

Parameters

room↩ Id	The ID of the Room to be removed from the accommodation.

Returns

> true if the room was found and removed; otherwise, false.

Definition at line 265 of file Accommodation.cs.

6.1.3.4 FindRoomById()

```
Room? SmartStay.Core.Models.Accommodation.FindRoomById (
            int roomId ) [inline]
```

Finds and returns a room from the accommodation by its room ID.

Parameters

room↩ Id	The ID of the room to find.

Returns

> The room with the specified ID, or null if not found.

Definition at line 239 of file Accommodation.cs.

6.1.3.5 ToString()

```
override string SmartStay.Core.Models.Accommodation.ToString ( ) [inline]
```

Overridden ToString method to provide accommodation information in a readable JSON format.

Returns

> A JSON string representation of the accommodation object.

Definition at line 336 of file Accommodation.cs.

6.1.4 Property Documentation

6.1.4.1 Address

```
string SmartStay.Core.Models.Accommodation.Address [get], [set], [add]
```

Public getter and setter for the Address.

Definition at line 215 of file Accommodation.cs.

6.1.4.2 Id

`int SmartStay.Core.Models.Accommodation.Id [get]`

Public getter for the accommodation ID.

Definition at line 183 of file Accommodation.cs.

6.1.4.3 LastAssignedId

`int SmartStay.Core.Models.Accommodation.LastAssignedId [static], [get], [set]`

Public getter and setter for the last assigned ID.

Definition at line 171 of file Accommodation.cs.

6.1.4.4 Name

`string SmartStay.Core.Models.Accommodation.Name [get], [set]`

Public getter and setter for the Name.

Definition at line 206 of file Accommodation.cs.

6.1.4.5 OwnerId

`int SmartStay.Core.Models.Accommodation.OwnerId [get], [set]`

Public getter and setter for the Owner ID.

Definition at line 188 of file Accommodation.cs.

6.1.4.6 Rooms

`List<Room> SmartStay.Core.Models.Accommodation.Rooms [get]`

Gets a deep copy of the list of rooms in the accommodation.

This property creates and returns a deep copy of the underlying rooms collection. Modifications to the returned list or its elements will not affect the original data.

Performance Note: Creating a deep copy can incur a performance cost, especially for large collections. Use this property sparingly if performance is critical.

Definition at line 232 of file Accommodation.cs.

6.1.4.7 Type

AccommodationType SmartStay.Core.Models.Accommodation.Type [get], [set]

Public getter and setter for the Type.

Definition at line 197 of file Accommodation.cs.
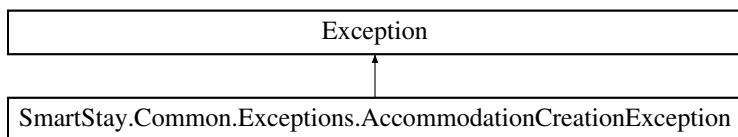
The documentation for this class was generated from the following file:

- Accommodation.cs

6.2 SmartStay.Common.Exceptions.AccommodationCreationException Class Reference

Represents an error that occurs during the accommodation creation process in the SmartStay application. This exception is thrown when there is an issue with validating or processing the accommodation's data.

Inheritance diagram for SmartStay.Common.Exceptions.AccommodationCreationException:

```
┌─────────────────────────────────────────────────────────────────┐
│                          Exception                               │
└─────────────────────────────────────────────────────────────────┘
                               ▲
                               │
┌─────────────────────────────────────────────────────────────────┐
│  SmartStay.Common.Exceptions.AccommodationCreationException       │
└─────────────────────────────────────────────────────────────────┘
```

Public Member Functions

- AccommodationCreationException (string message)

    Initializes a new instance of the AccommodationCreationException class with a specified error message.
- AccommodationCreationException (string message, Exception innerException)

    Initializes a new instance of the AccommodationCreationException class with a specified error message and a reference to the inner exception that is the cause of this exception.
- override string ToString ()

    Returns a string representation of the AccommodationCreationException instance, including the error message and any inner exceptions.

Properties

- override string Message [get]

    Gets the error message that explains the reason for the exception.

6.2.1 Detailed Description

Represents an error that occurs during the accommodation creation process in the SmartStay application. This exception is thrown when there is an issue with validating or processing the accommodation's data.

The AccommodationCreationException class extends the base Exception class, providing more specific context about errors encountered during the creation of an accommodation object. This is typically used when validation or other errors occur while trying to create a new accommodation.

Definition at line 25 of file AccommodationCreationException.cs.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 AccommodationCreationException() [1/2]

```
SmartStay.Common.Exceptions.AccommodationCreationException.AccommodationCreationException (
            string message )  [inline]
```

Initializes a new instance of the AccommodationCreationException class with a specified error message.

Parameters

message	The message that describes the error.

Definition at line 32 of file AccommodationCreationException.cs.

6.2.2.2 AccommodationCreationException() [2/2]

```
SmartStay.Common.Exceptions.AccommodationCreationException.AccommodationCreationException (
            string message,
            Exception innerException )  [inline]
```

Initializes a new instance of the AccommodationCreationException class with a specified error message and a reference to the inner exception that is the cause of this exception.

Parameters

message	The message that describes the error.
innerException	The exception that is the cause of the current exception.

Definition at line 42 of file AccommodationCreationException.cs.

6.2.3 Member Function Documentation

6.2.3.1 ToString()

```
override string SmartStay.Common.Exceptions.AccommodationCreationException.ToString ( )  [inline]
```

Returns a string representation of the AccommodationCreationException instance, including the error message and any inner exceptions.

Returns

A string that represents the current exception, typically including the error message and any inner exceptions.

For example, the string representation could look like: "Accommodation creation failed due to invalid owner ID."

Definition at line 69 of file AccommodationCreationException.cs.

6.2.4 Property Documentation

6.2.4.1 Message

```
override string SmartStay.Common.Exceptions.AccommodationCreationException.Message [get]
```

Gets the error message that explains the reason for the exception.

The error message that describes the reason for the exception.

This property is inherited from the Exception class and can be used to retrieve the message passed when the exception was thrown.

Definition at line 56 of file AccommodationCreationException.cs.
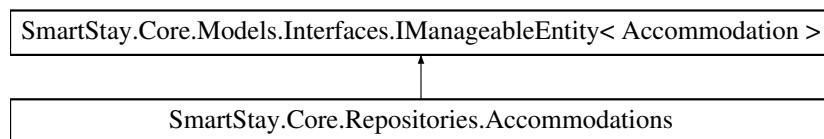
The documentation for this class was generated from the following file:

- AccommodationCreationException.cs

6.3 SmartStay.Core.Repositories.Accommodations Class Reference

Represents a collection of Accommodation objects, managed in a dictionary for fast lookup by accommodation ID.

Inheritance diagram for SmartStay.Core.Repositories.Accommodations:

```
┌─────────────────────────────────────────────────────────────────────────┐
│  SmartStay.Core.Models.Interfaces.IManageableEntity< Accommodation >      │
└─────────────────────────────────────────────────────────────────────────┘
                                    ▲
                                    │
┌─────────────────────────────────────────────────────────────────────────┐
│           SmartStay.Core.Repositories.Accommodations                      │
└─────────────────────────────────────────────────────────────────────────┘
```

Public Member Functions

- bool Add (Accommodation accommodation)

    Attempts to add a new accommodation to the collection.
- bool Remove (Accommodation accommodation)

    Removes an accommodation from the collection.
- ImportResult Import (string data)

    Imports accommodations from a JSON string into the collection. Existing accommodations with the same ID are replaced.
- string Export ()

    Exports the current list of accommodations to a JSON string.
- void Save (string filePath)

    Saves the current state of the accommodations collection to a file by serializing the object into a Protobuf format. If an error occurs during the saving process, it will be caught and logged.
- void Load (string filePath)

    Loads the collection from a binary file and assigns it to the current instance. If an error occurs during the loading process, it will be caught and logged.
- Accommodation? FindAccommodationById (int accommodationId)

    Finds an accommodation by its unique ID.
- int CountAccommodations ()

    Counts the number of accommodations in the collection.

Public Member Functions inherited from
SmartStay.Core.Models.Interfaces.IManageableEntity< **Accommodation** >

- bool Add (T item)

    Adds a single entity of type T to the collection.
- bool Remove (T item)

    Removes a specified entity of type T from the collection.
- ImportResult Import (string data)

    Imports a list of items from a serialized string.
- string Export ()

    Exports the current list of items as a serialized string.
- void Save (string filePath)

    Saves the collection to a binary file.
- void Load (string filePath)

    Loads the collection from a binary file.

6.3.1 Detailed Description

Represents a collection of Accommodation objects, managed in a dictionary for fast lookup by accommodation ID.

Definition at line 32 of file Accommodations.cs.

6.3.2 Member Function Documentation

6.3.2.1 Add()

```
bool SmartStay.Core.Repositories.Accommodations.Add (
            Accommodation accommodation )  [inline]
```

Attempts to add a new accommodation to the collection.

Parameters

accommodation	The Accommodation to add to the collection.

Returns

> `true` if the accommodation was successfully added to the collection; `false` if an accommodation with the same ID already exists in the collection.

Exceptions

ArgumentNullException	Thrown if *accommodation* is `null`.

Definition at line 57 of file Accommodations.cs.

6.3.2.2 CountAccommodations()

```
int SmartStay.Core.Repositories.Accommodations.CountAccommodations ( ) [inline]
```

Counts the number of accommodations in the collection.

Returns

The number of accommodations in the collection.

Definition at line 285 of file Accommodations.cs.

6.3.2.3 Export()

```
string SmartStay.Core.Repositories.Accommodations.Export ( ) [inline]
```

Exports the current list of accommodations to a JSON string.

Returns

A JSON string representation of the accommodations in the collection.</returns>

Definition at line 137 of file Accommodations.cs.

6.3.2.4 FindAccommodationById()

```
Accommodation? SmartStay.Core.Repositories.Accommodations.FindAccommodationById (
            int accommodationId ) [inline]
```

Finds an accommodation by its unique ID.

Parameters

accommodation↩ Id	The unique ID of the accommodation to find.

Returns

Returns the Accommodation object if found; otherwise, `null`.

Definition at line 273 of file Accommodations.cs.

6.3.2.5 Import()

```
ImportResult SmartStay.Core.Repositories.Accommodations.Import (
            string data ) [inline]
```

Imports accommodations from a JSON string into the collection. Existing accommodations with the same ID are replaced.

Parameters

data	The JSON string containing the list of accommodations.

Returns

An ImportResult summarizing the outcome of the import operation.

Exceptions

ArgumentException	Thrown if the data is null or empty.
ArgumentException	Thrown if deserialization of the data fails.

Definition at line 102 of file Accommodations.cs.

6.3.2.6   Load()

```
void SmartStay.Core.Repositories.Accommodations.Load (
            string filePath )  [inline]
```

Loads the collection from a binary file and assigns it to the current instance. If an error occurs during the loading process, it will be caught and logged.

Parameters

filePath	The file path to load the collection from.

Exceptions

IOException	Thrown when an I/O error occurs while loading the data.
SerializationException	Thrown when a deserialization error occurs while loading the data.

Definition at line 237 of file Accommodations.cs.

6.3.2.7   Remove()

```
bool SmartStay.Core.Repositories.Accommodations.Remove (
            Accommodation accommodation )  [inline]
```

Removes an accommodation from the collection.

Parameters

accommodation	The Accommodation object to remove from the collection.

Returns

true if the accommodation was successfully removed from the collection; false if the accommodation was not found.

Exceptions

ArgumentNullException	Thrown if *accommodation* is null.</exception

Definition at line 82 of file Accommodations.cs.

6.3.2.8 Save()

```
void SmartStay.Core.Repositories.Accommodations.Save (
            string filePath ) [inline]
```

Saves the current state of the accommodations collection to a file by serializing the object into a Protobuf format. If an error occurs during the saving process, it will be caught and logged.

Parameters

filePath	The path of the file to save the data.

Exceptions

IOException	Thrown when an I/O error occurs while saving the data.
SerializationException	Thrown when a serialization error occurs while saving the data.

Definition at line 204 of file Accommodations.cs.
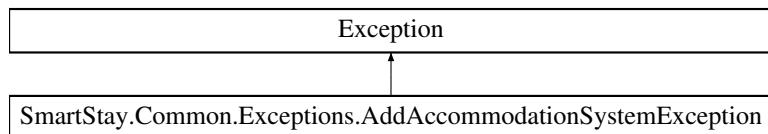
The documentation for this class was generated from the following file:

- Accommodations.cs

6.4 SmartStay.Common.Exceptions.AddAccommodationSystem↩ Exception Class Reference

Represents an error that occurs when an accommodation cannot be added to the system. This exception is thrown when there is an issue during the addition process, such as conflicts, system-level errors, or other reasons preventing the accommodation from being added.

Inheritance diagram for SmartStay.Common.Exceptions.AddAccommodationSystemException:

```
┌─────────────────────────────────────────────────────────────────┐
│                            Exception                             │
└─────────────────────────────────────────────────────────────────┘
                                 ▲
                                 │
┌─────────────────────────────────────────────────────────────────┐
│  SmartStay.Common.Exceptions.AddAccommodationSystemException     │
└─────────────────────────────────────────────────────────────────┘
```

Public Member Functions

- AddAccommodationSystemException (string message)

  Initializes a new instance of the AddAccommodationSystemException class with a specified error message.
- AddAccommodationSystemException (string message, Exception innerException)

  Initializes a new instance of the AddAccommodationSystemException class with a specified error message and a reference to the inner exception that is the cause of this exception.
- override string ToString ()

  Returns a string representation of the AddAccommodationSystemException instance, including the error message and any inner exceptions.

Properties

- override string Message `[get]`

  Gets the error message that explains the reason for the exception.

6.4.1 Detailed Description

Represents an error that occurs when an accommodation cannot be added to the system. This exception is thrown when there is an issue during the addition process, such as conflicts, system-level errors, or other reasons preventing the accommodation from being added.

The AddAccommodationSystemException class extends the base Exception class, providing more specific context about errors encountered while adding an accommodation to the system.

Definition at line 25 of file AddAccommodationSystemException.cs.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 AddAccommodationSystemException() [1/2]

```
SmartStay.Common.Exceptions.AddAccommodationSystemException.AddAccommodationSystemException (
            string message )  [inline]
```

Initializes a new instance of the AddAccommodationSystemException class with a specified error message.

Parameters

message	The message that describes the error.

Definition at line 32 of file AddAccommodationSystemException.cs.

6.4.2.2 AddAccommodationSystemException() [2/2]

```
SmartStay.Common.Exceptions.AddAccommodationSystemException.AddAccommodationSystemException (
            string message,
            Exception innerException )  [inline]
```

Initializes a new instance of the AddAccommodationSystemException class with a specified error message and a reference to the inner exception that is the cause of this exception.

Parameters

message	The message that describes the error.
innerException	The exception that is the cause of the current exception.

Definition at line 42 of file AddAccommodationSystemException.cs.

6.4.3 Member Function Documentation

6.4.3.1 ToString()

```
override string SmartStay.Common.Exceptions.AddAccommodationSystemException.ToString ( )  [inline]
```

Returns a string representation of the AddAccommodationSystemException instance, including the error message and any inner exceptions.

Returns

A string that represents the current exception, typically including the error message and any inner exceptions.

For example, the string representation could look like: "Failed to add accommodation to the system due to a conflict or system-level error."

Definition at line 69 of file AddAccommodationSystemException.cs.

6.4.4 Property Documentation

6.4.4.1 Message

```
override string SmartStay.Common.Exceptions.AddAccommodationSystemException.Message  [get]
```

Gets the error message that explains the reason for the exception.

The error message that describes the reason for the exception.

This property is inherited from the Exception class and can be used to retrieve the message passed when the exception was thrown.

Definition at line 56 of file AddAccommodationSystemException.cs.

The documentation for this class was generated from the following file:

  • AddAccommodationSystemException.cs

6.5 SmartStay.Core.Services.BookingManager Class Reference

Provides a facade for managing clients, reservations, and accommodations in the booking system. This class centralizes all operations for adding, removing, importing, and exporting data for these entities. It interacts with internal repositories to simplify the main API and ensure a standardized approach.

Public Member Functions

- BookingManager (ILogger< BookingManager > logger)

  Constructor to initialize the BookingManager with logger and repository dependencies.
- void SaveAll (string dataFolder)

  Saves all repositories (Clients, Accommodations, Reservations, Owners) to their respective files.
- void LoadAll (string dataFolder)

  Loads all repositories (Clients, Accommodations, Reservations, Owners) from their respective files.
- Client CreateBasicClient (string firstName, string lastName, string email)

  Creates a new client with basic information and adds them to the system. This method validates the input parameters and handles any validation errors. If an exception occurs during the client creation, a ClientCreationException is thrown.
- Client CreateCompleteClient (string firstName, string lastName, string email, string phoneNumber, string address)

  Creates a new client with all information and adds them to the system. This method validates the input parameters and handles any validation errors. If an exception occurs during the client creation, a ClientCreationException is thrown.
- Client FindClientById (int clientId)

  Finds a client in the system by their unique ID.
- UpdateClientResult UpdateClient (int clientId, string? firstName=null, string? lastName=null, string? email=null, string? phoneNumber=null, string? address=null, PaymentMethod paymentMethod=Payment↩Method.Unchanged)

  Updates the details of an existing client.
- bool RemoveClient (int clientId)

  Removes a client from the system.
- Owner CreateBasicOwner (string firstName, string lastName, string email)

  Creates a new owner with basic information and adds them to the system. This method validates the input parameters and handles any validation errors. If an exception occurs during the owner creation, a OwnerCreationException is thrown.
- Owner CreateCompleteOwner (string firstName, string lastName, string email, string phoneNumber, string address)

  Creates a new owner with all information and adds them to the system. This method validates the input parameters and handles any validation errors. If an exception occurs during the owner creation, a OwnerCreationException is thrown.
- Owner FindOwnerById (int ownerId)

  Finds an owner in the system by their unique ID.
- UpdateOwnerResult UpdateOwner (int ownerId, string? firstName=null, string? lastName=null, string? email=null, string? phoneNumber=null, string? address=null)

  Updates the details of an existing owner.
- bool RemoveOwner (int ownerId)

  Removes an owner from the system.
- Reservation CreateReservation (int clientId, int accommodationId, int roomId, DateTime checkIn, DateTime checkOut)

  Creates a new reservation for a specified client, accommodation, and room within a given date range.
- UpdateReservationResult UpdateReservation (int reservationId, DateTime? newCheckIn=null, DateTime? newCheckOut=null)

  Updates the check-in and/or check-out dates of an existing reservation.
- CancellationResult CancelReservation (int reservationId)

  Cancels a reservation by its unique ID, freeing up the associated accommodation for the specified dates.
- Accommodation CreateAccommodation (int ownerId, AccommodationType type, string name, string address)

  Creates and adds a new accommodation to the system. This method validates the input parameters, checks if the owner exists, and handles any validation or system errors. If any issue occurs during the creation of the accommodation or adding it to the system, specific exceptions are thrown to handle the error appropriately.

- UpdateAccommodationResult UpdateAccommodation (int accommodationId, AccommodationType type=AccommodationType.None, string? name=null, string? address=null)

    Updates the details of an existing accommodation.

- RemoveAccommodationResult RemoveAccommodation (int accommodationId)

    Removes an accommodation from the system and disassociates it from its owner. This method will first ensure the accommodation exists in the system, then check if the owner of the accommodation is valid. If both are found, the accommodation is removed from the system and from the owner's list of accommodations.

Properties

- Owners Owners   [get]

    Exposes the `Owners` repository as a read-only property.

- Clients Clients   [get]

    Exposes the `Clients` repository as a read-only property.

- Reservations Reservations   [get]

    Exposes the `Reservations` repository as a read-only property.

- Accommodations Accommodations   [get]

    Exposes the `Accommodations` repository as a read-only property.

6.5.1 Detailed Description

Provides a facade for managing clients, reservations, and accommodations in the booking system. This class centralizes all operations for adding, removing, importing, and exporting data for these entities. It interacts with internal repositories to simplify the main API and ensure a standardized approach.

This class offers a unified interface for handling key booking operations and data entities, facilitating integrations with other system components or external applications.

Definition at line 35 of file BookingManager.cs.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 BookingManager()

```
SmartStay.Core.Services.BookingManager.BookingManager (
            ILogger< BookingManager > logger )  [inline]
```

Constructor to initialize the BookingManager with logger and repository dependencies.

Parameters

logger	The logger to be used for logging activities within the BookingManager.

Definition at line 72 of file BookingManager.cs.

6.5.3 Member Function Documentation

6.5.3.1 CancelReservation()

```
CancellationResult SmartStay.Core.Services.BookingManager.CancelReservation (
            int reservationId ) [inline]
```

Cancels a reservation by its unique ID, freeing up the associated accommodation for the specified dates.

Parameters

reservation← Id	The unique ID of the reservation to cancel.

Returns

> A CancellationResult value indicating the outcome of the cancellation attempt.

This method cancels a reservation, which involves:

- Finding the reservation by its unique ID.

- Finding the associated accommodation and room based on the reservation.

- Removing the reservation from the room's reserved dates.

- Marking the reservation as cancelled and freeing up the accommodation for future bookings.

The following exceptions are handled during the cancellation process:

- If the reservation does not exist, a EntityNotFoundException is thrown.

- If the accommodation or room associated with the reservation cannot be found, a EntityNotFoundException is thrown.

If an error occurs during the cancellation process (e.g., failure to remove the reservation from the room), an appropriate error message is logged and the cancellation attempt is considered a failure.

<note type="warning"> The cancellation will not succeed if the room cannot be found or if there is an error in removing the reservation. </note>

Definition at line 972 of file BookingManager.cs.

6.5.3.2 CreateAccommodation()

```
Accommodation SmartStay.Core.Services.BookingManager.CreateAccommodation (
            int ownerId,
            AccommodationType type,
            string name,
            string address ) [inline]
```

Creates and adds a new accommodation to the system. This method validates the input parameters, checks if the owner exists, and handles any validation or system errors. If any issue occurs during the creation of the accommodation or adding it to the system, specific exceptions are thrown to handle the error appropriately.

Parameters

owner← Id	The unique ID of the accommodation owner.
type	The type of the accommodation (e.g., hotel, apartment, etc.).
name	The name of the accommodation (e.g., "Luxury Hotel").
address	The address of the accommodation (e.g., "123 Main St, City, Country").

Returns

The newly created Accommodation object, representing the accommodation that was successfully created and added.

Exceptions

EntityNotFoundException	Thrown if the owner with the specified ID is not found in the system. This ensures that the accommodation is associated with a valid owner before creation.
OwnerAddAccommodationException	Thrown if the accommodation could not be added to the owner's list of accommodations. This exception is raised if the owner has any issues adding the accommodation to their list, such as a conflict or internal error.
AddAccommodationSystemException	Thrown if an error occurs while adding the accommodation to the system (e.g., system error, storage issue). This ensures that the accommodation is successfully added to the system after being added to the owner's list.
AccommodationCreationException	Thrown if an error occurs during the accommodation creation process, including issues with owner association or system errors. This is a general exception to catch and rethrow more specific errors during the accommodation creation process.

This method attempts to create a new accommodation, ensuring that the accommodation type, name, and address are valid. The owner is validated by ID, and the accommodation is added both to the owner's list and the overall system. If any step fails, appropriate exceptions are thrown to allow for specific error handling.

Definition at line 1220 of file BookingManager.cs.

6.5.3.3 CreateBasicClient()

```
Client SmartStay.Core.Services.BookingManager.CreateBasicClient (
            string firstName,
            string lastName,
            string email )  [inline]
```

Creates a new client with basic information and adds them to the system. This method validates the input parameters and handles any validation errors. If an exception occurs during the client creation, a ClientCreationException is thrown.

Parameters

firstName	The first name of the client.
lastName	The last name of the client.
email	The email address of the client.

Exceptions

ClientCreationException	Thrown when an error occurs during the creation of the client, typically due to invalid input parameters or other issues that prevent the client from being created.

Returns

The Client object created.

Definition at line 207 of file BookingManager.cs.

6.5.3.4 CreateBasicOwner()

```
Owner SmartStay.Core.Services.BookingManager.CreateBasicOwner (
            string firstName,
            string lastName,
            string email ) [inline]
```

Creates a new owner with basic information and adds them to the system. This method validates the input parameters and handles any validation errors. If an exception occurs during the owner creation, a OwnerCreationException is thrown.

Parameters

firstName	The first name of the owner.
lastName	The last name of the owner.
email	The email address of the owner.

Exceptions

OwnerCreationException	Thrown when an error occurs during the creation of the owner, typically due to invalid input parameters or other issues that prevent the owner from being created.

Returns

The Owner object created.

Definition at line 472 of file BookingManager.cs.

6.5.3.5 CreateCompleteClient()

```
Client SmartStay.Core.Services.BookingManager.CreateCompleteClient (
            string firstName,
            string lastName,
            string email,
            string phoneNumber,
            string address ) [inline]
```

Creates a new client with all information and adds them to the system. This method validates the input parameters and handles any validation errors. If an exception occurs during the client creation, a ClientCreationException is thrown.

Parameters

firstName	The first name of the client.
lastName	The last name of the client.
email	The email address of the client.
phoneNumber	The phone number of the client.
address	The residential address of the client.

Exceptions

ClientCreationException	Thrown when an error occurs during the creation of the client, typically due to invalid input parameters or other issues that prevent the client from being created.

Returns

The Client object created.

Definition at line 254 of file BookingManager.cs.

6.5.3.6 CreateCompleteOwner()

```
Owner SmartStay.Core.Services.BookingManager.CreateCompleteOwner (
            string firstName,
            string lastName,
            string email,
            string phoneNumber,
            string address )  [inline]
```

Creates a new owner with all information and adds them to the system. This method validates the input parameters and handles any validation errors. If an exception occurs during the owner creation, a OwnerCreationException is thrown.

Parameters

firstName	The first name of the owner.
lastName	The last name of the owner.
email	The email address of the owner.
phoneNumber	The phone number of the owner.
address	The residential address of the owner.

Exceptions

OwnerCreationException	Thrown when an error occurs during the creation of the owner, typically due to invalid input parameters or other issues that prevent the owner from being created.

Returns

The Owner object created.

Definition at line 519 of file BookingManager.cs.

6.5.3.7 CreateReservation()

```
Reservation SmartStay.Core.Services.BookingManager.CreateReservation (
            int clientId,
            int accommodationId,
            int roomId,
            DateTime checkIn,
            DateTime checkOut )  [inline]
```

Creates a new reservation for a specified client, accommodation, and room within a given date range.

Parameters

clientId	The unique identifier of the client creating the reservation.
accommodation↩Id	The unique identifier of the accommodation where the reservation is being made.
roomId	The unique identifier of the room to be reserved.
checkIn	The check-in date for the reservation.
checkOut	The check-out date for the reservation.

Returns

A Reservation object representing the successfully created reservation.

Exceptions

ArgumentException	Thrown if: • The specified accommodation is not found. • The specified room is not found within the accommodation. • The specified room is unavailable for the given date range. • The reservation could not be added to the reservation list.
TotalCostException	Thrown if there is an error calculating the total cost of the reservation.
ReservationCreationException	Thrown if there is an error validating the reservation details during creation.

This method performs the following steps:

1. Logs the reservation attempt with the provided parameters.

2. Finds the specified accommodation and room.

3. Validates the room's availability using Room.IsAvailable.

4. Calculates the total cost of the reservation using Room.CalculateTotalCost.

5. Creates a new Reservation object after validation.

6. Attempts to add the reservation to the room and the central reservation list.

If any step fails, appropriate exceptions are thrown with detailed logging.

Definition at line 748 of file BookingManager.cs.

6.5.3.8 FindClientById()

```
Client SmartStay.Core.Services.BookingManager.FindClientById (
            int clientId ) [inline]
```

Finds a client in the system by their unique ID.

Parameters

client↩ id	The unique identifier for the client.

Exceptions

ValidationException	Thrown when any of the input parameters are invalid.

Returns

A Client object if found, otherwise throws an exception.

Definition at line 298 of file BookingManager.cs.

6.5.3.9 FindOwnerById()

```
Owner SmartStay.Core.Services.BookingManager.FindOwnerById (
            int ownerId ) [inline]
```

Finds an owner in the system by their unique ID.

Parameters

owner↩ Id	The unique identifier for the owner.

Returns

An Owner object if found, otherwise null.

Exceptions

ValidationException	Thrown when any of the input parameters are invalid.

Definition at line 562 of file BookingManager.cs.

6.5.3.10 LoadAll()

```
void SmartStay.Core.Services.BookingManager.LoadAll (
            string dataFolder ) [inline]
```

Loads all repositories (Clients, Accommodations, Reservations, Owners) from their respective files.

Definition at line 162 of file BookingManager.cs.

6.5.3.11 RemoveAccommodation()

```
RemoveAccommodationResult SmartStay.Core.Services.BookingManager.RemoveAccommodation (
            int accommodationId ) [inline]
```

Removes an accommodation from the system and disassociates it from its owner. This method will first ensure the accommodation exists in the system, then check if the owner of the accommodation is valid. If both are found, the accommodation is removed from the system and from the owner's list of accommodations.

Parameters

accommodation↩ Id	The unique ID of the accommodation to remove from the system.

Returns

Returns a RemoveAccommodationResult indicating the result of the removal operation.

This method ensures that both the accommodation and the associated owner are updated in the system to reflect the removal. The accommodation is removed from the system, and the relationship between the accommodation and the owner is also removed.

Definition at line 1382 of file BookingManager.cs.

6.5.3.12 RemoveClient()

```
bool SmartStay.Core.Services.BookingManager.RemoveClient (
            int clientId ) [inline]
```

Removes a client from the system.

Parameters

client↩ Id	The unique ID of the client to remove.

Returns

True if the client was found and removed, otherwise false.

Definition at line 436 of file BookingManager.cs.

6.5.3.13 RemoveOwner()

```
bool SmartStay.Core.Services.BookingManager.RemoveOwner (
            int ownerId ) [inline]
```

Removes an owner from the system.

Parameters

owner↩ Id	The unique ID of the owner to remove.

Returns

True if the owner was found and removed, otherwise false.

Definition at line 688 of file BookingManager.cs.

6.5.3.14 SaveAll()

```
void SmartStay.Core.Services.BookingManager.SaveAll (
              string dataFolder ) [inline]
```

Saves all repositories (Clients, Accommodations, Reservations, Owners) to their respective files.

Definition at line 134 of file BookingManager.cs.

6.5.3.15 UpdateAccommodation()

```
UpdateAccommodationResult SmartStay.Core.Services.BookingManager.UpdateAccommodation (
              int accommodationId,
              AccommodationType type = AccommodationType::None,
              string? name = null,
              string? address = null ) [inline]
```

Updates the details of an existing accommodation.

Parameters

accommodation↩ Id	The ID of the accommodation to update.
type	The new type of the accommodation (optional).
name	The new name of the accommodation (optional).
address	The new address of the accommodation (optional).

Returns

Returns an UpdateAccommodationResult indicating the result of the update operation.

This method updates the type, name, and address of an accommodation. It performs validation before updating any fields and ensures that only valid information is used to update the accommodation details.

Definition at line 1305 of file BookingManager.cs.

6.5.3.16 UpdateClient()

UpdateClientResult SmartStay.Core.Services.BookingManager.UpdateClient (
           int *clientId,*
           string? *firstName = null,*
           string? *lastName = null,*
           string? *email = null,*
           string? *phoneNumber = null,*
           string? *address = null,*
           PaymentMethod *paymentMethod = PaymentMethod::Unchanged*) [inline]

Updates the details of an existing client.

Parameters

clientId	The unique ID of the client to update.
firstName	The new first name of the client.
lastName	The new last name of the client.
email	The new email address of the client.
phoneNumber	The new phone number of the client.
address	The new address of the client.
paymentMethod	The new preferred payment method of the client.

Returns

Returns an UpdateClientResult indicating the result of the update operation.

This method attempts to update the details of an existing client by validating the provided data. If the client with the specified ID is not found, it returns UpdateClientResult.ClientNotFound. If any of the fields fail validation, the corresponding error code is returned. If all validations pass, the client details are updated, and UpdateClient↩
Result.Success is returned.

Definition at line 333 of file BookingManager.cs.

6.5.3.17 UpdateOwner()

UpdateOwnerResult SmartStay.Core.Services.BookingManager.UpdateOwner (
           int *ownerId,*
           string? *firstName = null,*
           string? *lastName = null,*
           string? *email = null,*
           string? *phoneNumber = null,*
           string? *address = null*) [inline]

Updates the details of an existing owner.

Parameters

ownerId	The unique ID of the owner to update.
firstName	The new first name of the owner.
lastName	The new last name of the owner.
email	The new email address of the owner.
phoneNumber	The new phone number of the owner.
address	The new address of the owner.

Returns

Returns an UpdateOwnerResult indicating the result of the update operation.

This method attempts to update the details of an existing owner by validating the provided data. If the owner with the specified ID is not found, it returns UpdateOwnerResult.OwnerNotFound. If any of the fields fail validation, the corresponding error code is returned. If all validations pass, the owner details are updated, and UpdateOwner↩ Result.Success is returned.

Definition at line 599 of file BookingManager.cs.

6.5.3.18 UpdateReservation()

```
UpdateReservationResult SmartStay.Core.Services.BookingManager.UpdateReservation (
            int reservationId,
            DateTime? newCheckIn = null,
            DateTime? newCheckOut = null ) [inline]
```

Updates the check-in and/or check-out dates of an existing reservation.

Parameters

reservationId	The unique ID of the reservation to update.
newCheckIn	The new check-in date for the reservation, or null if no change is required.
newCheckOut	The new check-out date for the reservation, or null if no change is required.

Returns

Returns an UpdateReservationResult indicating the result of the update operation.

This method updates the check-in and check-out dates of a reservation if necessary. It checks the availability of the associated accommodation and room for the new dates. The method excludes the current reservation's existing date range when verifying availability. If no new dates are specified, the reservation remains unchanged.

The following conditions are checked during the update:

- Whether the reservation exists

- Whether the accommodation and room associated with the reservation exist

- Whether the new dates are valid (check-out must be later than check-in)

- Whether the room is available for the new dates (excluding the current reservation's dates)

Definition at line 867 of file BookingManager.cs.

6.5.4 Property Documentation

6.5.4.1 Accommodations

```
Accommodations SmartStay.Core.Services.BookingManager.Accommodations [get]
```

Exposes the Accommodations repository as a read-only property.

Definition at line 100 of file BookingManager.cs.

6.5.4.2 Clients

`Clients` SmartStay.Core.Services.BookingManager.Clients `[get]`

Exposes the `Clients` repository as a read-only property.

Definition at line 90 of file BookingManager.cs.

6.5.4.3 Owners

`Owners` SmartStay.Core.Services.BookingManager.Owners `[get]`

Exposes the `Owners` repository as a read-only property.

Definition at line 85 of file BookingManager.cs.

6.5.4.4 Reservations

`Reservations` SmartStay.Core.Services.BookingManager.Reservations `[get]`

Exposes the `Reservations` repository as a read-only property.

Definition at line 95 of file BookingManager.cs.

The documentation for this class was generated from the following file:

- BookingManager.cs

6.6 SmartStay.Core.Models.Client Class Reference

Defines the Client class, which encapsulates the details of a client including personal information such as first name, last name, email address, phone number, residential address, and preferred payment method. This class validates the provided data upon creation or when modifying specific properties, ensuring that all data is consistent and correct.

Public Member Functions

- Client ()

  Initializes a new instance of the Client class.
- Client (string firstName, string lastName, string email)

  Constructor to initialize a new client with basic details: first name, last name, and email. Validates the input parameters.
  Throws specific validation exceptions if any input is invalid.
- Client (string firstName, string lastName, string email, string phoneNumber, string address)

  Constructor to initialize a new client with basic details (first name, last name, email) and additional details (phone number and address). Validates the input parameters and throws validation exceptions for any invalid inputs.
- Client (string firstName, string lastName, string email, string phoneNumber, string address, PaymentMethod preferredPaymentMethod)

  Constructor to initialize a new client with all details including the preferred payment method.
- Client (int id, string firstName, string lastName, string email, string phoneNumber, string address, PaymentMethod preferredPaymentMethod)

  Constructor to initialize a new Client with all details, including a manually specified ID, first name, last name, email, phone number, address, and preferred payment method. **This constructor should be avoided in normal cases** *as it allows manual assignment of the client ID, which can lead to conflicts and issues with ID uniqueness. The system is designed to automatically handle unique ID assignment, and using other constructors is recommended for creating client objects to ensure proper handling of IDs.*
  This constructor is marked with [JsonConstructor] so it will be used for JSON deserialization purposes, but it should not be used when creating new client objects manually.
- override string ToString ()

  Overridden ToString method to provide client information in a readable JSON format.

Properties

- static int LastAssignedId `[get, set]`

    Public getter and setter for the last assigned ID.
- int Id `[get]`

    Public getter for the user Id.
- string FirstName `[get, set]`

    Public getter and setter for the FirstName. Sets the value after validating it.
- string LastName `[get, set]`

    Public getter and setter for the LastName. Sets the value after validating it.
- string Email `[get, set]`

    Public getter and setter for the Email. Sets the value after validating it.
- string PhoneNumber `[get, set]`

    Public getter and setter for the PhoneNumber. Sets the value after validating it.
- string Address `[get, set]`

    Public getter and setter for the Address. Sets the value after validating it.
- PaymentMethod PreferredPaymentMethod `[get, set]`

    Public getter and setter for the PreferredPaymentMethod. Sets the value after validating it.

6.6.1 Detailed Description

Defines the Client class, which encapsulates the details of a client including personal information such as first name, last name, email address, phone number, residential address, and preferred payment method. This class validates the provided data upon creation or when modifying specific properties, ensuring that all data is consistent and correct.

Definition at line 33 of file Client.cs.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 Client() [1/5]

```
SmartStay.Core.Models.Client.Client ( )  [inline]
```

Initializes a new instance of the Client class.

This constructor is required for Protobuf-net serialization/deserialization.

It should **not** be used directly in normal application code. Instead, use the constructor with parameters for creating instances of Client.

Definition at line 108 of file Client.cs.

6.6.2.2 Client() [2/5]

```
SmartStay.Core.Models.Client.Client (
            string firstName,
            string lastName,
            string email )  [inline]
```

Constructor to initialize a new client with basic details: first name, last name, and email. Validates the input parameters.
Throws specific validation exceptions if any input is invalid.

Parameters

firstName	The first name of the client.
lastName	The last name of the client.
email	The email address of the client.

Exceptions

ValidationException	Thrown if the first name, last name, or email is invalid. Each validation has a specific error code: **InvalidName:** if the first or last name is invalid. **InvalidEmail:** if the email address is invalid.

Definition at line 127 of file Client.cs.

6.6.2.3 Client() [3/5]

```
SmartStay.Core.Models.Client.Client (
            string firstName,
            string lastName,
            string email,
            string phoneNumber,
            string address )  [inline]
```

Constructor to initialize a new client with basic details (first name, last name, email) and additional details (phone number and address). Validates the input parameters and throws validation exceptions for any invalid inputs.

Parameters

firstName	The first name of the client.
lastName	The last name of the client.
email	The email address of the client.
phoneNumber	The phone number of the client.
address	The residential address of the client.

Exceptions

ValidationException	Thrown when any of the input parameters are invalid. Each validation has a specific error code: **InvalidName:** if the first or last name is invalid (from the basic constructor). **InvalidEmail:** if the email address is invalid (from the basic constructor). **InvalidPhoneNumber:** if the phone number is invalid. **InvalidAddress:** if the address is invalid.

Definition at line 156 of file Client.cs.

6.6.2.4 Client() [4/5]

```
SmartStay.Core.Models.Client.Client (
            string firstName,
```

```
        string lastName,
        string email,
        string phoneNumber,
        string address,
        PaymentMethod preferredPaymentMethod ) [inline]
```

Constructor to initialize a new client with all details including the preferred payment method.

Parameters

firstName	The first name of the client.
lastName	The last name of the client.
email	The email address of the client.
phoneNumber	The phone number of the client.
address	The residential address of the client.
preferredPaymentMethod	The preferred payment method of the client.

Exceptions

ValidationException	Thrown when any of the input parameters are invalid: **InvalidName:** if the first or last name is invalid. **InvalidEmail:** if the email address is invalid. **InvalidPhoneNumber:** if the phone number is invalid. **InvalidAddress:** if the address is invalid. **InvalidPaymentMethod:** if the preferred payment method is invalid.

Returns

The Client object created.

Definition at line 183 of file Client.cs.

6.6.2.5 Client() [5/5]

```
SmartStay.Core.Models.Client.Client (
        int id,
        string firstName,
        string lastName,
        string email,
        string phoneNumber,
        string address,
        PaymentMethod preferredPaymentMethod ) [inline]
```

Constructor to initialize a new Client with all details, including a manually specified ID, first name, last name, email, phone number, address, and preferred payment method. **This constructor should be avoided in normal cases** as it allows manual assignment of the client ID, which can lead to conflicts and issues with ID uniqueness. The system is designed to automatically handle unique ID assignment, and using other constructors is recommended for creating client objects to ensure proper handling of IDs.

This constructor is marked with [JsonConstructor] so it will be used for JSON deserialization purposes, but it should not be used when creating new client objects manually.

Parameters

id	The manually specified ID of the client. This should not be used under normal circumstances as the system handles ID assignment automatically.
firstName	The first name of the client.
lastName	The last name of the client.
email	The email address of the client.
phoneNumber	The phone number of the client.
address	The residential address of the client.
preferredPaymentMethod	The preferred payment method of the client.

Definition at line 210 of file Client.cs.

6.6.3 Member Function Documentation

6.6.3.1 ToString()

```
override string SmartStay.Core.Models.Client.ToString ( )  [inline]
```

Overridden ToString method to provide client information in a readable JSON format.

Returns

A JSON string representation of the client object.

Definition at line 332 of file Client.cs.

6.6.4 Property Documentation

6.6.4.1 Address

```
string SmartStay.Core.Models.Client.Address  [get], [set], [add]
```

Public getter and setter for the Address. Sets the value after validating it.

Definition at line 284 of file Client.cs.

6.6.4.2 Email

```
string SmartStay.Core.Models.Client.Email  [get], [set]
```

Public getter and setter for the Email. Sets the value after validating it.

Definition at line 264 of file Client.cs.

6.6.4.3 FirstName

```
string SmartStay.Core.Models.Client.FirstName  [get], [set]
```

Public getter and setter for the FirstName. Sets the value after validating it.

Definition at line 244 of file Client.cs.

6.6.4.4 Id

```
int SmartStay.Core.Models.Client.Id  [get]
```

Public getter for the user Id.

Definition at line 238 of file Client.cs.

6.6.4.5 LastAssignedId

```
int SmartStay.Core.Models.Client.LastAssignedId  [static], [get], [set]
```

Public getter and setter for the last assigned ID.

Definition at line 226 of file Client.cs.

6.6.4.6 LastName

```
string SmartStay.Core.Models.Client.LastName  [get], [set]
```

Public getter and setter for the LastName. Sets the value after validating it.

Definition at line 254 of file Client.cs.

6.6.4.7 PhoneNumber

```
string SmartStay.Core.Models.Client.PhoneNumber  [get], [set]
```

Public getter and setter for the PhoneNumber. Sets the value after validating it.

Definition at line 274 of file Client.cs.

6.6.4.8 PreferredPaymentMethod

```
PaymentMethod SmartStay.Core.Models.Client.PreferredPaymentMethod [get], [set]
```

Public getter and setter for the PreferredPaymentMethod. Sets the value after validating it.

Definition at line 294 of file Client.cs.
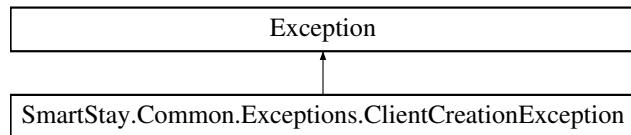
The documentation for this class was generated from the following file:

- Client.cs

6.7 SmartStay.Common.Exceptions.ClientCreationException Class Reference

Represents an error that occurs during the client creation process in the SmartStay application. This exception is thrown when there is an issue with validating or processing the client's data.

Inheritance diagram for SmartStay.Common.Exceptions.ClientCreationException:

```
┌─────────────────────────────────────────────────────┐
│                     Exception                        │
└─────────────────────────────────────────────────────┘
                          ▲
                          │
┌─────────────────────────────────────────────────────┐
│ SmartStay.Common.Exceptions.ClientCreationException  │
└─────────────────────────────────────────────────────┘
```

Public Member Functions

- ClientCreationException (string message)

    Initializes a new instance of the ClientCreationException class with a specified error message.
- ClientCreationException (string message, Exception innerException)

    Initializes a new instance of the ClientCreationException class with a specified error message and a reference to the inner exception that is the cause of this exception.
- override string ToString ()

    Returns a string representation of the ClientCreationException instance, including the error message and any inner exceptions.

Properties

- override string Message `[get]`

    Gets the error message that explains the reason for the exception.

6.7.1 Detailed Description

Represents an error that occurs during the client creation process in the SmartStay application. This exception is thrown when there is an issue with validating or processing the client's data.

The ClientCreationException class extends the base Exception class, providing more specific context about errors encountered during the creation of a client. This is typically used when validation or other errors occur while trying to create a new client object.

Definition at line 25 of file ClientCreationException.cs.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 ClientCreationException() [1/2]

```
SmartStay.Common.Exceptions.ClientCreationException.ClientCreationException (
            string message ) [inline]
```

Initializes a new instance of the ClientCreationException class with a specified error message.

Parameters

message	The message that describes the error.

Definition at line 31 of file ClientCreationException.cs.

6.7.2.2 ClientCreationException() [2/2]

```
SmartStay.Common.Exceptions.ClientCreationException.ClientCreationException (
            string message,
            Exception innerException )  [inline]
```

Initializes a new instance of the ClientCreationException class with a specified error message and a reference to the inner exception that is the cause of this exception.

Parameters

message	The message that describes the error.
innerException	The exception that is the cause of the current exception.

Definition at line 41 of file ClientCreationException.cs.

6.7.3 Member Function Documentation

6.7.3.1 ToString()

```
override string SmartStay.Common.Exceptions.ClientCreationException.ToString ( )  [inline]
```

Returns a string representation of the ClientCreationException instance, including the error message and any inner exceptions.

Returns

A string that represents the current exception, typically including the error message and any inner exceptions.

For example, the string representation could look like: "Client creation failed due to invalid email format."

Definition at line 68 of file ClientCreationException.cs.

6.7.4 Property Documentation

6.7.4.1 Message

```
override string SmartStay.Common.Exceptions.ClientCreationException.Message  [get]
```

Gets the error message that explains the reason for the exception.

The error message that describes the reason for the exception.

This property is inherited from the Exception class and can be used to retrieve the message passed when the exception was thrown.

Definition at line 55 of file ClientCreationException.cs.
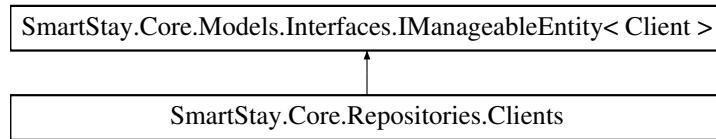
The documentation for this class was generated from the following file:

- ClientCreationException.cs

6.8 SmartStay.Core.Repositories.Clients Class Reference

Represents a collection of Client objects, managed in a dictionary for fast lookup by client ID. Implements the IManageableEntity<Client> interface for standardized management.

Inheritance diagram for SmartStay.Core.Repositories.Clients:

```
┌─────────────────────────────────────────────────────────────┐
│ SmartStay.Core.Models.Interfaces.IManageableEntity< Client > │
└─────────────────────────────────────────────────────────────┘
                              ▲
┌─────────────────────────────────────────────────────────────┐
│          SmartStay.Core.Repositories.Clients                 │
└─────────────────────────────────────────────────────────────┘
```

Public Member Functions

- bool Add (Client client)

  Attempts to add a new client to the collection.
- bool Remove (Client client)

  Removes a client from the collection.
- ImportResult Import (string data)

  Imports a list of clients from a JSON string. Replaces any existing clients with the same ID in the collection.
- string Export ()

  Exports the current list of clients to a JSON string.
- void Save (string filePath)

  Saves the current state of the clients collection to a file by serializing the object into a Protobuf format. If an error occurs during the saving process, it will be caught and logged.
- void Load (string filePath)

  Loads the collection from a binary file and assigns it to the current instance. If an error occurs during the loading process, it will be caught and logged.
- Client? FindClientById (int id)

  Finds a client by their unique ID.
- int CountClients ()

  Counts the number of clients in the collection.

Public Member Functions inherited from SmartStay.Core.Models.Interfaces.IManageableEntity< Client >

- bool Add (T item)

  Adds a single entity of type T to the collection.
- bool Remove (T item)

  Removes a specified entity of type T from the collection.
- ImportResult Import (string data)

  Imports a list of items from a serialized string.
- string Export ()

  Exports the current list of items as a serialized string.
- void Save (string filePath)

  Saves the collection to a binary file.
- void Load (string filePath)

  Loads the collection from a binary file.

6.8.1 Detailed Description

Represents a collection of Client objects, managed in a dictionary for fast lookup by client ID. Implements the IManageableEntity<Client> interface for standardized management.

Definition at line 33 of file Clients.cs.

6.8.2 Member Function Documentation

6.8.2.1 Add()

```
bool SmartStay.Core.Repositories.Clients.Add (
            Client client ) [inline]
```

Attempts to add a new client to the collection.

Parameters

| client | The Client to add to the collection. |

Returns

> `true` if the client was successfully added to the collection; `false` if a client with the same ID already exists in the collection.

Exceptions

| ArgumentNullException | Thrown if *client* is `null`. |

Definition at line 60 of file Clients.cs.

6.8.2.2 CountClients()

```
int SmartStay.Core.Repositories.Clients.CountClients ( ) [inline]
```

Counts the number of clients in the collection.

Returns

> The number of clients in the collection.

Definition at line 274 of file Clients.cs.

6.8.2.3 Export()

```
string SmartStay.Core.Repositories.Clients.Export ( ) [inline]
```

Exports the current list of clients to a JSON string.

Returns

> A JSON string representation of the clients in the collection.

Definition at line 138 of file Clients.cs.

6.8.2.4 FindClientById()

Client? SmartStay.Core.Repositories.Clients.FindClientById (
        int *id*) [inline]

Finds a client by their unique ID.

Parameters

id	The unique ID of the client to find.

Returns

Returns the Client object if found; otherwise, null.

Definition at line 262 of file Clients.cs.

6.8.2.5 Import()

ImportResult SmartStay.Core.Repositories.Clients.Import (
        string *data*) [inline]

Imports a list of clients from a JSON string. Replaces any existing clients with the same ID in the collection.

Parameters

data	The JSON string containing the list of clients.

Returns

An ImportResult summarizing the outcome of the import operation.

Exceptions

ArgumentException	Thrown if the data is null or empty.
ArgumentException	Thrown if deserialization of the data fails.

Definition at line 104 of file Clients.cs.

6.8.2.6 Load()

void SmartStay.Core.Repositories.Clients.Load (
        string *filePath*) [inline]

Loads the collection from a binary file and assigns it to the current instance. If an error occurs during the loading process, it will be caught and logged.

Parameters

filePath	The file path to load the collection from.

Exceptions

IOException	Thrown when an I/O error occurs while loading the data.
SerializationException	Thrown when a deserialization error occurs while loading the data.

Definition at line 226 of file Clients.cs.

6.8.2.7 Remove()

```
bool SmartStay.Core.Repositories.Clients.Remove (
            Client client ) [inline]
```

Removes a client from the collection.

Parameters

client	The Client object to remove from the collection.

Returns

> `true` if the client was successfully removed from the collection; `false` if the client was not found.

Exceptions

ArgumentNullException	Thrown if *client* is `null.`

Definition at line 85 of file Clients.cs.

6.8.2.8 Save()

```
void SmartStay.Core.Repositories.Clients.Save (
            string filePath ) [inline]
```

Saves the current state of the clients collection to a file by serializing the object into a Protobuf format. If an error occurs during the saving process, it will be caught and logged.

Parameters

filePath	The path of the file to save the data.

Exceptions

IOException	Thrown when an I/O error occurs while saving the data.

Exceptions

SerializationException	Thrown when a serialization error occurs while saving the data.

Definition at line 193 of file Clients.cs.
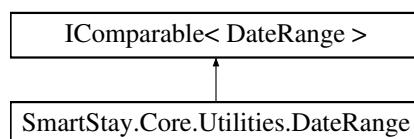
The documentation for this class was generated from the following file:

- Clients.cs

6.9 SmartStay.Core.Utilities.DateRange Class Reference

Represents a range of dates with a start and end date. Implements IComparable<DateRange> to allow sorting and comparisons.

Inheritance diagram for SmartStay.Core.Utilities.DateRange:

Public Member Functions

- DateRange (DateTime start, DateTime end)

    Initializes a new instance of the DateRange class.
- int CompareTo (DateRange? other)

    Compares the current DateRange with another DateRange to determine their relative order.
- override bool Equals (object? obj)

    Determines whether the current DateRange is equal to another DateRange.
- override int GetHashCode ()

    Returns a hash code for this DateRange based on its Start and End dates.
- DateRange Clone ()

    Creates a deep copy of the current DateRange instance.

Static Public Member Functions

- static bool operator== (DateRange left, DateRange right)

    Defines equality comparison for DateRange instances.
- static bool operator!= (DateRange left, DateRange right)

    Defines inequality comparison for DateRange instances.
- static bool operator< (DateRange left, DateRange right)

    Defines the less-than comparison for DateRange instances.
- static bool operator<= (DateRange left, DateRange right)

    Defines the less-than-or-equal comparison for DateRange instances.
- static bool operator> (DateRange left, DateRange right)

    Defines the greater-than comparison for DateRange instances.
- static bool operator>= (DateRange left, DateRange right)

    Defines the greater-than-or-equal comparison for DateRange instances.

Properties

- DateTime Start [get, set]

    Gets or sets the start date of the range.
- DateTime End [get, set]

    Gets or sets the end date of the range.

6.9.1   Detailed Description

Represents a range of dates with a start and end date. Implements IComparable<DateRange> to allow sorting and comparisons.

Definition at line 25 of file DateRange.cs.

6.9.2   Constructor & Destructor Documentation

6.9.2.1   DateRange()

```
SmartStay.Core.Utilities.DateRange.DateRange (
            DateTime start,
            DateTime end )  [inline]
```

Initializes a new instance of the DateRange class.

Parameters

start	The start date of the range.
end	The end date of the range.

Definition at line 46 of file DateRange.cs.

6.9.3   Member Function Documentation

6.9.3.1   Clone()

```
DateRange SmartStay.Core.Utilities.DateRange.Clone ( )  [inline]
```

Creates a deep copy of the current DateRange instance.

Returns

   A new DateRange instance with identical data to the current instance.

Definition at line 175 of file DateRange.cs.

6.9.3.2   CompareTo()

```
int SmartStay.Core.Utilities.DateRange.CompareTo (
            DateRange?  other )  [inline]
```

Compares the current DateRange with another DateRange to determine their relative order.

Parameters

other	The other DateRange to compare with.

Returns

A value less than zero if this instance precedes *other* ; zero if this instance occurs at the same position in the sort order; and a value greater than zero if this instance follows *other* .

Definition at line 62 of file DateRange.cs.

6.9.3.3 Equals()

```
override bool SmartStay.Core.Utilities.DateRange.Equals (
            object?  obj )  [inline]
```

Determines whether the current DateRange is equal to another DateRange.

Parameters

obj	The object to compare with.

Returns

`true` if the current DateRange is equal to the other DateRange; otherwise, `false`.

Definition at line 82 of file DateRange.cs.

6.9.3.4 GetHashCode()

```
override int SmartStay.Core.Utilities.DateRange.GetHashCode ( )  [inline]
```

Returns a hash code for this DateRange based on its Start and End dates.

Returns

A hash code for the current DateRange.

Definition at line 96 of file DateRange.cs.

6.9.3.5 operator"!=()

```
static bool SmartStay.Core.Utilities.DateRange.operator!= (
            DateRange left,
            DateRange right )  [inline], [static]
```

Defines inequality comparison for DateRange instances.

Parameters

left	The first DateRange.
right	The second DateRange.

Returns

> `true` if the DateRange instances are not equal; otherwise, `false`.

Definition at line 119 of file DateRange.cs.

6.9.3.6 operator<()

```
static bool SmartStay.Core.Utilities.DateRange.operator< (
            DateRange left,
            DateRange right )  [inline], [static]
```

Defines the less-than comparison for DateRange instances.

Parameters

left	The first DateRange.
right	The second DateRange.

Returns

> `true` if the first DateRange precedes the second; otherwise, `false`.

Definition at line 131 of file DateRange.cs.

6.9.3.7 operator<=()

```
static bool SmartStay.Core.Utilities.DateRange.operator<= (
            DateRange left,
            DateRange right )  [inline], [static]
```

Defines the less-than-or-equal comparison for DateRange instances.

Parameters

left	The first DateRange.
right	The second DateRange.

Returns

> `true` if the first DateRange precedes or is equal to the second; otherwise, `false`.

Definition at line 143 of file DateRange.cs.

6.9.3.8 operator==()

```
static bool SmartStay.Core.Utilities.DateRange.operator== (
            DateRange left,
            DateRange right ) [inline], [static]
```

Defines equality comparison for DateRange instances.

Parameters

left	The first DateRange.
right	The second DateRange.

Returns

> `true` if the DateRange instances are equal; otherwise, `false`.

Definition at line 108 of file DateRange.cs.

6.9.3.9 operator>()

```
static bool SmartStay.Core.Utilities.DateRange.operator> (
            DateRange left,
            DateRange right ) [inline], [static]
```

Defines the greater-than comparison for DateRange instances.

Parameters

left	The first DateRange.
right	The second DateRange.

Returns

> `true` if the first DateRange follows the second; otherwise, `false`.

Definition at line 154 of file DateRange.cs.

6.9.3.10 operator>=()

```
static bool SmartStay.Core.Utilities.DateRange.operator>= (
            DateRange left,
            DateRange right ) [inline], [static]
```

Defines the greater-than-or-equal comparison for DateRange instances.

Parameters

left	The first DateRange.
right	The second DateRange.

Returns

> `true` if the first DateRange follows or is equal to the second; otherwise, `false`.

Definition at line 166 of file DateRange.cs.

6.9.4 Property Documentation

6.9.4.1 End

`DateTime SmartStay.Core.Utilities.DateRange.End [get], [set]`

Gets or sets the end date of the range.

Definition at line 37 of file DateRange.cs.

6.9.4.2 Start

`DateTime SmartStay.Core.Utilities.DateRange.Start [get], [set]`

Gets or sets the start date of the range.

Definition at line 31 of file DateRange.cs.
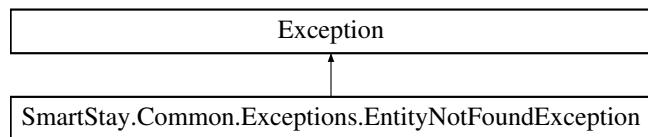
The documentation for this class was generated from the following file:

- DateRange.cs

6.10 SmartStay.Common.Exceptions.EntityNotFoundException Class Reference

Represents an error that occurs when an entity is not found in the system. This exception is thrown when an operation cannot proceed because the specified entity (e.g., Reservation, Room) does not exist in the system.

Inheritance diagram for SmartStay.Common.Exceptions.EntityNotFoundException:

Public Member Functions

- EntityNotFoundException (string entityType, int entityId)

  Initializes a new instance of the EntityNotFoundException class with a specified entity type and ID.
- EntityNotFoundException (string entityType, int entityId, string message)

  Initializes a new instance of the EntityNotFoundException class with a specified entity type, ID, and message.
- EntityNotFoundException (string entityType, int entityId, string message, Exception innerException)

  Initializes a new instance of the EntityNotFoundException class with a specified entity type, ID, message, and a reference to the inner exception that is the cause of this exception.
- override string ToString ()

  Returns a string representation of the EntityNotFoundException instance, including the error message and any inner exceptions.

Properties

- string EntityType `[get]`

  Gets the type of the entity that was not found.
- int EntityId `[get]`

  Gets the ID of the entity that was not found.

6.10.1 Detailed Description

Represents an error that occurs when an entity is not found in the system. This exception is thrown when an operation cannot proceed because the specified entity (e.g., Reservation, Room) does not exist in the system.

The EntityNotFoundException class extends the base Exception class, providing more specific context about errors encountered when an entity is not found in the system. This is typically used in scenarios where an operation cannot continue due to the absence of an expected entity.

Definition at line 26 of file EntityNotFoundException.cs.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 EntityNotFoundException() [1/3]

```
SmartStay.Common.Exceptions.EntityNotFoundException.EntityNotFoundException (
            string entityType,
            int entityId ) [inline]
```

Initializes a new instance of the EntityNotFoundException class with a specified entity type and ID.

Parameters

entityType	The type of the entity that was not found (e.g., "Reservation", "Room").
entityId	The ID of the entity that was not found.

Definition at line 58 of file EntityNotFoundException.cs.

6.10.2.2 EntityNotFoundException() [2/3]

```
SmartStay.Common.Exceptions.EntityNotFoundException.EntityNotFoundException (
          string entityType,
          int entityId,
          string message ) [inline]
```

Initializes a new instance of the EntityNotFoundException class with a specified entity type, ID, and message.

Parameters

entityType	The type of the entity that was not found (e.g., "Reservation", "Room").
entityId	The ID of the entity that was not found.
message	The message that describes the error.

Definition at line 72 of file EntityNotFoundException.cs.

6.10.2.3 EntityNotFoundException() [3/3]

```
SmartStay.Common.Exceptions.EntityNotFoundException.EntityNotFoundException (
          string entityType,
          int entityId,
          string message,
          Exception innerException ) [inline]
```

Initializes a new instance of the EntityNotFoundException class with a specified entity type, ID, message, and a reference to the inner exception that is the cause of this exception.

Parameters

entityType	The type of the entity that was not found (e.g., "Reservation", "Room").
entityId	The ID of the entity that was not found.
message	The message that describes the error.
innerException	The exception that is the cause of the current exception.

Definition at line 87 of file EntityNotFoundException.cs.

6.10.3 Member Function Documentation

6.10.3.1 ToString()

```
override string SmartStay.Common.Exceptions.EntityNotFoundException.ToString ( ) [inline]
```

Returns a string representation of the EntityNotFoundException instance, including the error message and any inner exceptions.

Returns

A string that represents the current exception, typically including the error message and any inner exceptions.

For example, the string representation could look like: "Reservation with ID 123 was not found."

Definition at line 105 of file EntityNotFoundException.cs.

6.10.4 Property Documentation

6.10.4.1 EntityId

`int SmartStay.Common.Exceptions.EntityNotFoundException.EntityId [get]`

Gets the ID of the entity that was not found.

An integer representing the ID of the missing entity.

This property provides the unique identifier of the entity that could not be found, allowing the caller to identify which specific entity was missing.

Definition at line 50 of file EntityNotFoundException.cs.

6.10.4.2 EntityType

`string SmartStay.Common.Exceptions.EntityNotFoundException.EntityType [get]`

Gets the type of the entity that was not found.

A string representing the entity type, e.g., "Reservation", "Room".

This property provides the name of the entity that caused the exception, allowing specific handling based on the entity type in exception filters or logging.

Definition at line 38 of file EntityNotFoundException.cs.

The documentation for this class was generated from the following file:

- EntityNotFoundException.cs

6.11 SmartStay.Core.Models.Interfaces.IManageableEntity$<$ in T $>$ Interface Template Reference

Defines the IManageableEntity$<$T$>$ interface for managing a collection of entities of type *T* . This interface standardizes methods for adding, removing, importing, and exporting entities.

Public Member Functions

- bool Add (T item)

  Adds a single entity of type T to the collection.
- bool Remove (T item)

  Removes a specified entity of type T from the collection.
- ImportResult Import (string data)

  Imports a list of items from a serialized string.
- string Export ()

  Exports the current list of items as a serialized string.
- void Save (string filePath)

  Saves the collection to a binary file.
- void Load (string filePath)

  Loads the collection from a binary file.

6.11.1 Detailed Description

Defines the IManageableEntity$<$T$>$ interface for managing a collection of entities of type *T* . This interface standardizes methods for adding, removing, importing, and exporting entities.

Template Parameters

T	The type of entities managed by the implementing collection class.

Definition at line 28 of file ManageableEntity.cs.

6.11.2 Member Function Documentation

6.11.2.1 Add()

```
bool SmartStay.Core.Models.Interfaces.IManageableEntity< in T >.Add (
            T item )
```

Adds a single entity of type *T* to the collection.

Parameters

item	The entity to add to the collection.

Returns

Returns `true` if the entity was successfully added; otherwise, `false`.

6.11.2.2 Export()

```
string SmartStay.Core.Models.Interfaces.IManageableEntity< in T >.Export ( )
```

Exports the current list of items as a serialized string.

Returns

A serialized string representing the collection of items.

6.11.2.3 Import()

```
ImportResult SmartStay.Core.Models.Interfaces.IManageableEntity< in T >.Import (
            string data )
```

Imports a list of items from a serialized string.

Parameters

data	The serialized string representing a collection of items.

6.11.2.4 Load()

void SmartStay.Core.Models.Interfaces.IManageableEntity< in T >.Load (
                string *filePath*)

Loads the collection from a binary file.

Parameters

filePath	The file path to load the collection from.

6.11.2.5 Remove()

bool SmartStay.Core.Models.Interfaces.IManageableEntity< in T >.Remove (
                T *item*)

Removes a specified entity of type *T* from the collection.

Parameters

item	The entity to remove from the collection.

Returns

Returns true if the entity was successfully removed; otherwise, false.

6.11.2.6 Save()

void SmartStay.Core.Models.Interfaces.IManageableEntity< in T >.Save (
                string *filePath*)

Saves the collection to a binary file.

Parameters

filePath	The file path to save the collection to.

The documentation for this interface was generated from the following file:

- ManageableEntity.cs

6.12 SmartStay.Common.Models.ImportResult Class Reference

Represents the result of an accommodation import operation, summarizing the outcome of the process.

Public Member Functions

- override string ToString ()

  Returns a string representation of the import result, including the number of imported, replaced, and total accommodations processed.

Properties

- int ImportedCount [get, set]

  Gets or sets the number of accommodations successfully imported.
- int ReplacedCount [get, set]

  Gets or sets the number of accommodations that were replaced because they already existed in the collection.
- int TotalCount [get]

  Gets the total number of accommodations processed during the import operation. This is the sum of ImportedCount and ReplacedCount.

6.12.1 Detailed Description

Represents the result of an accommodation import operation, summarizing the outcome of the process.

Definition at line 19 of file ImportResult.cs.

6.12.2 Member Function Documentation

6.12.2.1 ToString()

```
override string SmartStay.Common.Models.ImportResult.ToString ( )  [inline]
```

Returns a string representation of the import result, including the number of imported, replaced, and total accommodations processed.

Returns

A string summarizing the import result.

Definition at line 44 of file ImportResult.cs.

6.12.3 Property Documentation

6.12.3.1 ImportedCount

```
int SmartStay.Common.Models.ImportResult.ImportedCount  [get], [set]
```

Gets or sets the number of accommodations successfully imported.

Definition at line 24 of file ImportResult.cs.

6.12.3.2 ReplacedCount

`int SmartStay.Common.Models.ImportResult.ReplacedCount [get], [set]`

Gets or sets the number of accommodations that were replaced because they already existed in the collection.

Definition at line 29 of file ImportResult.cs.

6.12.3.3 TotalCount

`int SmartStay.Common.Models.ImportResult.TotalCount [get]`

Gets the total number of accommodations processed during the import operation. This is the sum of ImportedCount and ReplacedCount.

Definition at line 35 of file ImportResult.cs.

The documentation for this class was generated from the following file:

- ImportResult.cs

6.13 SmartStay.Core.Models.Owner Class Reference

Defines the Owner class, which encapsulates the details of an accommodation owner, including personal information such as first name, last name, email address, phone number, and a list of owned accommodations. This class validates the provided data upon creation or when modifying specific properties, ensuring that all data is consistent and correct.

Public Member Functions

- Owner ()

    Initializes a new instance of the Owner class.
- Owner (string firstName, string lastName, string email)

    Constructor to initialize a new owner with basic details: first name, last name, and email. Validates the input parameters.
- Owner (string firstName, string lastName, string email, string phoneNumber, string address)

    Constructor to initialize a new owner with basic details (first name, last name, email) and additional details (phone number and address).
- Owner (int id, string firstName, string lastName, string email, string phoneNumber, string address, List< Accommodation > accommodationsOwned)

    *Constructor to initialize a new Owner with all details, including a manually specified ID, first name, last name, email, phone number, address, and a list of accommodations owned. **This constructor should be avoided in normal cases** as it allows manual assignment of the owner ID, which can lead to conflicts and issues with ID uniqueness. The system is designed to automatically handle unique ID assignment, and using other constructors is recommended for creating owner objects to ensure proper handling of IDs.*
    This constructor is marked with [JsonConstructor] so it will be used for JSON deserialization purposes, but it should not be used when creating new owner objects manually.
- bool AddAccommodation (Accommodation accommodation)

    Adds the specified accommodation to the list of accommodations owned by the owner.
- bool RemoveAccommodation (Accommodation accommodation)

    Removes the specified accommodation from the list of accommodations owned by the owner.
- override string ToString ()

    Overridden ToString method to provide owner information in a readable JSON format.

Properties

- static int LastAssignedId `[get, set]`

  Public getter and setter for the last assigned ID.
- int Id `[get]`

  Public getter for the owner ID.
- string FirstName `[get, set]`

  Public getter and setter for the FirstName. Sets the value after validating it.
- string LastName `[get, set]`

  Public getter and setter for the LastName. Sets the value after validating it.
- string Email `[get, set]`

  Public getter and setter for the Email. Sets the value after validating it.
- string PhoneNumber `[get, set]`

  Public getter and setter for the PhoneNumber. Sets the value after validating it.
- string Address `[get, set]`

  Public getter and setter for the Address. Sets the value after validating it.
- List< Accommodation > AccommodationsOwned `[get]`

  Public getter for the list of accommodations owned by the owner.

6.13.1 Detailed Description

Defines the Owner class, which encapsulates the details of an accommodation owner, including personal information such as first name, last name, email address, phone number, and a list of owned accommodations. This class validates the provided data upon creation or when modifying specific properties, ensuring that all data is consistent and correct.

Definition at line 30 of file Owner.cs.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 Owner() [1/4]

```
SmartStay.Core.Models.Owner.Owner ( ) [inline]
```

Initializes a new instance of the Owner class.

This constructor is required for Protobuf-net serialization/deserialization.

It should **not** be used directly in normal application code. Instead, use the constructor with parameters for creating instances of Owner.

Definition at line 105 of file Owner.cs.

6.13.2.2 Owner() [2/4]

```
SmartStay.Core.Models.Owner.Owner (
            string firstName,
            string lastName,
            string email ) [inline]
```

Constructor to initialize a new owner with basic details: first name, last name, and email. Validates the input parameters.

Parameters

firstName	The first name of the owner.
lastName	The last name of the owner.
email	The email address of the owner.

Exceptions

ValidationException	Thrown when any of the input parameters are invalid. Each validation has a specific error code: **InvalidName:** if the first or last name is invalid (from the basic constructor). **InvalidEmail:** if the email address is invalid (from the basic constructor).

Definition at line 123 of file Owner.cs.

6.13.2.3 Owner() [3/4]

```
SmartStay.Core.Models.Owner.Owner (
            string firstName,
            string lastName,
            string email,
            string phoneNumber,
            string address )  [inline]
```

Constructor to initialize a new owner with basic details (first name, last name, email) and additional details (phone number and address).

Parameters

firstName	The first name of the owner.
lastName	The last name of the owner.
email	The email address of the owner.
phoneNumber	The phone number of the owner.
address	The residential address of the owner.

Exceptions

ValidationException	Thrown when any of the input parameters are invalid. Each validation has a specific error code: **InvalidName:** if the first or last name is invalid (from the basic constructor). **InvalidEmail:** if the email address is invalid (from the basic constructor). **InvalidPhoneNumber:** if the phone number is invalid. **InvalidAddress:** if the address is invalid.

Definition at line 151 of file Owner.cs.

6.13.2.4 Owner() [4/4]

```
SmartStay.Core.Models.Owner.Owner (
            int id,
```

```
             string firstName,
             string lastName,
             string email,
             string phoneNumber,
             string address,
             List< Accommodation > accommodationsOwned )  [inline]
```

Constructor to initialize a new Owner with all details, including a manually specified ID, first name, last name, email, phone number, address, and a list of accommodations owned. **This constructor should be avoided in normal cases** as it allows manual assignment of the owner ID, which can lead to conflicts and issues with ID uniqueness. The system is designed to automatically handle unique ID assignment, and using other constructors is recommended for creating owner objects to ensure proper handling of IDs.

This constructor is marked with [JsonConstructor] so it will be used for JSON deserialization purposes, but it should not be used when creating new owner objects manually.

Parameters

id	The manually specified ID of the owner. This should not be used under normal circumstances as the system handles ID assignment automatically.
firstName	The first name of the owner.
lastName	The last name of the owner.
email	The email address of the owner.
phoneNumber	The phone number of the owner.
address	The residential address of the owner.
accommodationsOwned	The list of accommodations owned by the owner.

Definition at line 179 of file Owner.cs.

6.13.3 Member Function Documentation

6.13.3.1 AddAccommodation()

```
bool SmartStay.Core.Models.Owner.AddAccommodation (
             Accommodation accommodation )  [inline]
```

Adds the specified accommodation to the list of accommodations owned by the owner.

Parameters

accommodation	The accommodation to add. Cannot be null.

Returns

True if the accommodation was successfully added; false if the provided accommodation is null.

This method does not perform any additional checks to ensure the accommodation is unique or validate its state. Ensure external validation is performed if required.

Definition at line 283 of file Owner.cs.

6.13.3.2 RemoveAccommodation()

```
bool SmartStay.Core.Models.Owner.RemoveAccommodation (
            Accommodation accommodation )  [inline]
```

Removes the specified accommodation from the list of accommodations owned by the owner.

Parameters

accommodation	The accommodation to remove. Cannot be null.

Returns

> True if the accommodation was successfully removed; false if the provided accommodation is null or not found in the list.

This method assumes that the list allows duplicate entries. If duplicates exist, only the first occurrence of the accommodation will be removed. Ensure external validation is performed if additional checks are required.

Definition at line 305 of file Owner.cs.

6.13.3.3 ToString()

```
override string SmartStay.Core.Models.Owner.ToString ( )  [inline]
```

Overridden ToString method to provide owner information in a readable JSON format.

Returns

> A JSON string representation of the owner object.

Definition at line 356 of file Owner.cs.

6.13.4 Property Documentation

6.13.4.1 AccommodationsOwned

```
List<Accommodation> SmartStay.Core.Models.Owner.AccommodationsOwned  [get]
```

Public getter for the list of accommodations owned by the owner.

This property creates and returns a deep copy of the underlying accommodations collection. Modifications to the returned list or its elements will not affect the original data.

Performance Note: Creating a deep copy can incur a performance cost, especially for large collections. Use this property sparingly if performance is critical.

Definition at line 270 of file Owner.cs.

6.13.4.2 Address

`string SmartStay.Core.Models.Owner.Address [get], [set], [add]`

Public getter and setter for the Address. Sets the value after validating it.

Definition at line 253 of file Owner.cs.

6.13.4.3 Email

`string SmartStay.Core.Models.Owner.Email [get], [set]`

Public getter and setter for the Email. Sets the value after validating it.

Definition at line 233 of file Owner.cs.

6.13.4.4 FirstName

`string SmartStay.Core.Models.Owner.FirstName [get], [set]`

Public getter and setter for the FirstName. Sets the value after validating it.

Definition at line 213 of file Owner.cs.

6.13.4.5 Id

`int SmartStay.Core.Models.Owner.Id [get]`

Public getter for the owner ID.

Definition at line 207 of file Owner.cs.

6.13.4.6 LastAssignedId

`int SmartStay.Core.Models.Owner.LastAssignedId [static], [get], [set]`

Public getter and setter for the last assigned ID.

Definition at line 195 of file Owner.cs.

6.13.4.7 LastName

`string SmartStay.Core.Models.Owner.LastName [get], [set]`

Public getter and setter for the LastName. Sets the value after validating it.

Definition at line 223 of file Owner.cs.

6.13.4.8 PhoneNumber

`string SmartStay.Core.Models.Owner.PhoneNumber [get], [set]`

Public getter and setter for the PhoneNumber. Sets the value after validating it.

Definition at line 243 of file Owner.cs.
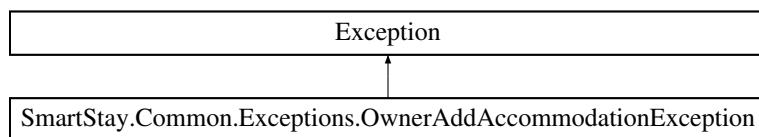
The documentation for this class was generated from the following file:

- Owner.cs

6.14 SmartStay.Common.Exceptions.OwnerAddAccommodation↩ Exception Class Reference

Represents an error that occurs when an accommodation cannot be added to an owner's list of accommodations. This exception is thrown when there is an issue with the adding process, such as validation failures, conflicts, or other reasons why the accommodation cannot be associated with the owner.

Inheritance diagram for SmartStay.Common.Exceptions.OwnerAddAccommodationException:

Public Member Functions

- OwnerAddAccommodationException (string message)

    Initializes a new instance of the OwnerAddAccommodationException class with a specified error message.
- OwnerAddAccommodationException (string message, Exception innerException)

    Initializes a new instance of the OwnerAddAccommodationException class with a specified error message and a reference to the inner exception that is the cause of this exception.
- override string ToString ()

    Returns a string representation of the OwnerAddAccommodationException instance, including the error message and any inner exceptions.

Properties

- override string Message [get]

    Gets the error message that explains the reason for the exception.

6.14.1 Detailed Description

Represents an error that occurs when an accommodation cannot be added to an owner's list of accommodations. This exception is thrown when there is an issue with the adding process, such as validation failures, conflicts, or other reasons why the accommodation cannot be associated with the owner.

The OwnerAddAccommodationException class extends the base Exception class, providing more specific context about errors encountered when adding an accommodation to the owner's list.

Definition at line 25 of file OwnerAddAccommodationException.cs.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 OwnerAddAccommodationException() [1/2]

```
SmartStay.Common.Exceptions.OwnerAddAccommodationException.OwnerAddAccommodationException (
            string message ) [inline]
```

Initializes a new instance of the OwnerAddAccommodationException class with a specified error message.

Parameters

message	The message that describes the error.

Definition at line 32 of file OwnerAddAccommodationException.cs.

6.14.2.2 OwnerAddAccommodationException() [2/2]

```
SmartStay.Common.Exceptions.OwnerAddAccommodationException.OwnerAddAccommodationException (
            string message,
            Exception innerException ) [inline]
```

Initializes a new instance of the OwnerAddAccommodationException class with a specified error message and a reference to the inner exception that is the cause of this exception.

Parameters

message	The message that describes the error.
innerException	The exception that is the cause of the current exception.

Definition at line 42 of file OwnerAddAccommodationException.cs.

6.14.3 Member Function Documentation

6.14.3.1 ToString()

```
override string SmartStay.Common.Exceptions.OwnerAddAccommodationException.ToString ( ) [inline]
```

Returns a string representation of the OwnerAddAccommodationException instance, including the error message and any inner exceptions.

Returns

A string that represents the current exception, typically including the error message and any inner exceptions.

For example, the string representation could look like: "Failed to add accommodation due to validation error."

Definition at line 69 of file OwnerAddAccommodationException.cs.

6.14.4 Property Documentation

6.14.4.1 Message

```
override string SmartStay.Common.Exceptions.OwnerAddAccommodationException.Message  [get]
```

Gets the error message that explains the reason for the exception.

The error message that describes the reason for the exception.

This property is inherited from the Exception class and can be used to retrieve the message passed when the exception was thrown.

Definition at line 56 of file OwnerAddAccommodationException.cs.
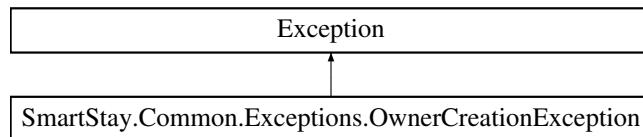
The documentation for this class was generated from the following file:

- OwnerAddAccommodationException.cs

6.15 SmartStay.Common.Exceptions.OwnerCreationException Class Reference

Represents an error that occurs during the owner creation process in the SmartStay application. This exception is thrown when there is an issue with validating or processing the owner's data.

Inheritance diagram for SmartStay.Common.Exceptions.OwnerCreationException:

```
┌─────────────────────────────────────────────────────────┐
│                        Exception                         │
└─────────────────────────────────────────────────────────┘
                             ▲
                             │
┌─────────────────────────────────────────────────────────┐
│   SmartStay.Common.Exceptions.OwnerCreationException     │
└─────────────────────────────────────────────────────────┘
```

Public Member Functions

- OwnerCreationException (string message)

    Initializes a new instance of the OwnerCreationException class with a specified error message.

- OwnerCreationException (string message, Exception innerException)

    Initializes a new instance of the OwnerCreationException class with a specified error message and a reference to the inner exception that is the cause of this exception.

- override string ToString ()

    Returns a string representation of the OwnerCreationException instance, including the error message and any inner exceptions.

Properties

- override string Message [get]

    Gets the error message that explains the reason for the exception.

6.15.1 Detailed Description

Represents an error that occurs during the owner creation process in the SmartStay application. This exception is thrown when there is an issue with validating or processing the owner's data.

The OwnerCreationException class extends the base Exception class, providing more specific context about errors encountered during the creation of an owner. This is typically used when validation or other errors occur while trying to create a new owner object.

Definition at line 25 of file OwnerCreationException.cs.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 OwnerCreationException() [1/2]

```
SmartStay.Common.Exceptions.OwnerCreationException.OwnerCreationException (
            string message ) [inline]
```

Initializes a new instance of the OwnerCreationException class with a specified error message.

Parameters

message	The message that describes the error.

Definition at line 31 of file OwnerCreationException.cs.

6.15.2.2 OwnerCreationException() [2/2]

```
SmartStay.Common.Exceptions.OwnerCreationException.OwnerCreationException (
            string message,
            Exception innerException ) [inline]
```

Initializes a new instance of the OwnerCreationException class with a specified error message and a reference to the inner exception that is the cause of this exception.

Parameters

message	The message that describes the error.
innerException	The exception that is the cause of the current exception.

Definition at line 41 of file OwnerCreationException.cs.

6.15.3 Member Function Documentation

6.15.3.1 ToString()

```
override string SmartStay.Common.Exceptions.OwnerCreationException.ToString ( ) [inline]
```

Returns a string representation of the OwnerCreationException instance, including the error message and any inner exceptions.

Returns

A string that represents the current exception, typically including the error message and any inner exceptions.

For example, the string representation could look like: "Owner creation failed due to invalid email format."

Definition at line 68 of file OwnerCreationException.cs.

6.15.4 Property Documentation

6.15.4.1 Message

```
override string SmartStay.Common.Exceptions.OwnerCreationException.Message  [get]
```

Gets the error message that explains the reason for the exception.

The error message that describes the reason for the exception.

This property is inherited from the Exception class and can be used to retrieve the message passed when the exception was thrown.

Definition at line 55 of file OwnerCreationException.cs.
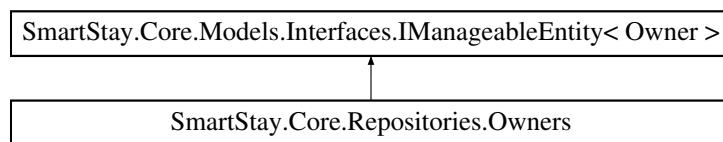
The documentation for this class was generated from the following file:

- OwnerCreationException.cs

6.16 SmartStay.Core.Repositories.Owners Class Reference

Represents a collection of Owner objects, managed in a dictionary for fast lookup by owner ID. Implements the IManageableEntity<Owner> interface for standardized management.

Inheritance diagram for SmartStay.Core.Repositories.Owners:

Public Member Functions

- bool Add (Owner owner)

  Attempts to add a new owner to the collection.
- bool Remove (Owner owner)

  Removes an owner from the collection.
- ImportResult Import (string data)

  Imports a list of owners from a JSON string. Replaces any existing owners with the same ID in the collection.
- string Export ()

  Exports the current list of owners to a JSON string.
- void Save (string filePath)

  Saves the current state of the owners collection to a file by serializing the object into a Protobuf format. If an error occurs during the saving process, it will be caught and logged.
- void Load (string filePath)

  Loads the collection from a binary file and assigns it to the current instance. If an error occurs during the loading process, it will be caught and logged.
- Owner? FindOwnerById (int id)

  Finds an owner by their unique ID.
- int CountOwners ()

  Counts the number of owners in the collection.

Public Member Functions inherited from
SmartStay.Core.Models.Interfaces.IManageableEntity< Owner >

- bool Add (T item)

    Adds a single entity of type T to the collection.
- bool Remove (T item)

    Removes a specified entity of type T from the collection.
- ImportResult Import (string data)

    Imports a list of items from a serialized string.
- string Export ()

    Exports the current list of items as a serialized string.
- void Save (string filePath)

    Saves the collection to a binary file.
- void Load (string filePath)

    Loads the collection from a binary file.

6.16.1 Detailed Description

Represents a collection of Owner objects, managed in a dictionary for fast lookup by owner ID. Implements the IManageableEntity<Owner> interface for standardized management.

Definition at line 30 of file Owners.cs.

6.16.2 Member Function Documentation

6.16.2.1 Add()

```
bool SmartStay.Core.Repositories.Owners.Add (
            Owner owner )  [inline]
```

Attempts to add a new owner to the collection.

Parameters

owner	The Owner to add to the collection.

Returns

`true` if the owner was successfully added to the collection; `false` if an owner with the same ID already exists in the collection.

Exceptions

ArgumentNullException	Thrown if *owner* is `null`.

Definition at line 57 of file Owners.cs.

6.16.2.2 CountOwners()

```
int SmartStay.Core.Repositories.Owners.CountOwners ( )    [inline]
```

Counts the number of owners in the collection.

Returns

The number of owners in the collection.

Definition at line 282 of file Owners.cs.

6.16.2.3 Export()

```
string SmartStay.Core.Repositories.Owners.Export ( )    [inline]
```

Exports the current list of owners to a JSON string.

Returns

A JSON string representation of the owners in the collection.

Definition at line 135 of file Owners.cs.

6.16.2.4 FindOwnerById()

```
Owner?  SmartStay.Core.Repositories.Owners.FindOwnerById (
            int id )    [inline]
```

Finds an owner by their unique ID.

Parameters

id	The unique ID of the owner to find.

Returns

Returns the Owner object if found; otherwise, `null`.

Definition at line 270 of file Owners.cs.

6.16.2.5 Import()

```
ImportResult SmartStay.Core.Repositories.Owners.Import (
            string data )    [inline]
```

Imports a list of owners from a JSON string. Replaces any existing owners with the same ID in the collection.

Parameters

data	The JSON string containing the list of owners.

Returns

An ImportResult summarizing the outcome of the import operation.

Exceptions

ArgumentException	Thrown if the data is null or empty.
ArgumentException	Thrown if deserialization of the data fails.

Definition at line 101 of file Owners.cs.

6.16.2.6 Load()

```
void SmartStay.Core.Repositories.Owners.Load (
            string filePath ) [inline]
```

Loads the collection from a binary file and assigns it to the current instance. If an error occurs during the loading process, it will be caught and logged.

Parameters

filePath	The file path to load the collection from.

Exceptions

IOException	Thrown when an I/O error occurs while loading the data.
SerializationException	Thrown when a deserialization error occurs while loading the data.

Definition at line 234 of file Owners.cs.

6.16.2.7 Remove()

```
bool SmartStay.Core.Repositories.Owners.Remove (
            Owner owner ) [inline]
```

Removes an owner from the collection.

Parameters

owner	The Owner object to remove from the collection.

Returns

> `true` if the owner was successfully removed from the collection; `false` if the owner was not found.

Exceptions

ArgumentNullException	Thrown if *owner* is `null`.

Definition at line 82 of file Owners.cs.

6.16.2.8 Save()

```
void SmartStay.Core.Repositories.Owners.Save (
            string filePath ) [inline]
```

Saves the current state of the owners collection to a file by serializing the object into a Protobuf format. If an error occurs during the saving process, it will be caught and logged.

Parameters

filePath	The path of the file to save the data.

Exceptions

IOException	Thrown when an I/O error occurs while saving the data.
SerializationException	Thrown when a serialization error occurs while saving the data.

Definition at line 201 of file Owners.cs.

The documentation for this class was generated from the following file:

- Owners.cs

6.17 SmartStay.Core.Models.Payment Class Reference

Represents a payment made in the SmartStay system, with details such as amount, date, method, and status.

Public Member Functions

- Payment ()

  Initializes a new instance of the Payment class.
- Payment (int reservationId, decimal amount, DateTime paymentDate, PaymentMethod paymentMethod, PaymentStatus paymentStatus)

  Initializes a new instance of the Payment class with specified details.
- Payment (int id, int reservationId, decimal amount, DateTime date, PaymentMethod method, PaymentStatus status)

*Constructor to initialize a new Payment with all details, including a manually specified ID, reservation ID, amount, date, payment method, and payment status. **This constructor should be avoided in normal cases** as it allows manual assignment of the payment ID, which can lead to conflicts and issues with ID uniqueness. The system is designed to automatically handle unique ID assignment, and other constructors should be used for creating payment objects to ensure proper handling of IDs.*
This constructor is marked with [JsonConstructor] so it will be used for JSON deserialization purposes, but it should not be used when creating new payment objects manually.

- Payment Clone ()

  Creates a deep copy of the current Payment instance.
- override string ToString ()

  Overridden ToString method to provide payment information in a readable JSON format.

Properties

- static int LastAssignedId `[get, set]`

  Public getter and setter for the last assigned ID.
- int Id `[get]`

  Public getter for the payment Id.
- int ReservationId `[get]`

  Public getter for the reservation Id being paid.
- decimal Amount `[get]`

  Public getter for the Amount.
- DateTime Date `[get]`

  Gets the date the payment was made.
- PaymentMethod Method `[get]`

  Gets the method used for the payment (e.g., PayPal, Bank Transfer).
- PaymentStatus Status `[get, set]`

  Gets or sets the status of the payment. When setting, validates the new status using Validator.ValidatePaymentStatus.

6.17.1 Detailed Description

Represents a payment made in the SmartStay system, with details such as amount, date, method, and status.

Definition at line 30 of file Payment.cs.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 Payment() [1/3]

```
SmartStay.Core.Models.Payment.Payment ( )  [inline]
```

Initializes a new instance of the Payment class.

This constructor is required for Protobuf-net serialization/deserialization.

It should **not** be used directly in normal application code. Instead, use the constructor with parameters for creating instances of Payment.

Definition at line 96 of file Payment.cs.

6.17.2.2 Payment() [2/3]

```
SmartStay.Core.Models.Payment.Payment (
            int reservationId,
            decimal amount,
            DateTime paymentDate,
            PaymentMethod paymentMethod,
            PaymentStatus paymentStatus )  [inline]
```

Initializes a new instance of the Payment class with specified details.

Parameters

amount	The amount for the payment.
paymentDate	The date when the payment was made.
paymentMethod	The method used for the payment.
paymentStatus	The status of the payment.

Exceptions

ValidationException	Thrown when the provided reservation id, amount, payment method, or payment status is invalid.

Definition at line 112 of file Payment.cs.

6.17.2.3 Payment() [3/3]

```
SmartStay.Core.Models.Payment.Payment (
            int id,
            int reservationId,
            decimal amount,
            DateTime date,
            PaymentMethod method,
            PaymentStatus status )  [inline]
```

Constructor to initialize a new Payment with all details, including a manually specified ID, reservation ID, amount, date, payment method, and payment status. **This constructor should be avoided in normal cases** as it allows manual assignment of the payment ID, which can lead to conflicts and issues with ID uniqueness. The system is designed to automatically handle unique ID assignment, and other constructors should be used for creating payment objects to ensure proper handling of IDs.

This constructor is marked with [JsonConstructor] so it will be used for JSON deserialization purposes, but it should not be used when creating new payment objects manually.

Parameters

id	The manually specified ID of the payment. This should not be used under normal circumstances as the system handles ID assignment automatically.
reservation↩Id	The ID of the reservation associated with the payment.
amount	The amount paid for the reservation.
date	The date when the payment was made.
method	The payment method used for the transaction.
status	The status of the payment (e.g., Pending, Completed, Failed).

Definition at line 145 of file Payment.cs.

6.17.3 Member Function Documentation

6.17.3.1 Clone()

`Payment SmartStay.Core.Models.Payment.Clone ( ) [inline]`

Creates a deep copy of the current Payment instance.

Returns

A new Payment instance with identical data to the current instance.

Definition at line 239 of file Payment.cs.

6.17.3.2 ToString()

`override string SmartStay.Core.Models.Payment.ToString ( ) [inline]`

Overridden ToString method to provide payment information in a readable JSON format.

Returns

A JSON string representation of the payment object.

Definition at line 255 of file Payment.cs.

6.17.4 Property Documentation

6.17.4.1 Amount

`decimal SmartStay.Core.Models.Payment.Amount [get]`

Public getter for the Amount.

Definition at line 181 of file Payment.cs.

6.17.4.2 Date

`DateTime SmartStay.Core.Models.Payment.Date [get]`

Gets the date the payment was made.

Definition at line 186 of file Payment.cs.

6.17.4.3 Id

```
int SmartStay.Core.Models.Payment.Id  [get]
```

Public getter for the payment Id.

Definition at line 171 of file Payment.cs.

6.17.4.4 LastAssignedId

```
int SmartStay.Core.Models.Payment.LastAssignedId  [static], [get], [set]
```

Public getter and setter for the last assigned ID.

Definition at line 159 of file Payment.cs.

6.17.4.5 Method

```
PaymentMethod SmartStay.Core.Models.Payment.Method  [get]
```

Gets the method used for the payment (e.g., PayPal, Bank Transfer).

Definition at line 191 of file Payment.cs.

6.17.4.6 ReservationId

```
int SmartStay.Core.Models.Payment.ReservationId  [get]
```

Public getter for the reservation Id being paid.

Definition at line 176 of file Payment.cs.

6.17.4.7 Status

```
PaymentStatus SmartStay.Core.Models.Payment.Status  [get], [set]
```

Gets or sets the status of the payment. When setting, validates the new status using Validator.ValidatePayment↩
Status.

Exceptions

ValidationException	Thrown when the provided status is invalid.

Definition at line 200 of file Payment.cs.

The documentation for this class was generated from the following file:

- Payment.cs

6.18 SmartStay.Core.Models.Reservation Class Reference

Defines the Reservation class, which encapsulates reservation details such as client ID, accommodation type, dates, and payment information. This class ensures data consistency by validating input parameters upon creation or when modifying specific properties.

Public Member Functions

- Reservation ()

    Initializes a new instance of the Reservation class.

- Reservation (int clientId, int accommodationId, int roomId, AccommodationType accommodationType, Date↩
    Time checkInDate, DateTime checkOutDate, decimal totalCost)

    Constructor to initialize a new reservation with essential details. Validates the input parameters.

- Reservation (int id, int clientId, int accommodationId, int roomId, AccommodationType accommodation↩
    Type, DateTime checkInDate, DateTime checkOutDate, ReservationStatus status, decimal totalCost, decimal
    amountPaid, List< Payment > payments)

    Constructor to initialize a new Reservation with all details, including a manually specified ID, client ID, accommodation ID, room ID, accommodation type, check-in and check-out dates, reservation status, total cost, amount paid, and associated payments. ***This constructor should be avoided in normal cases*** *as it allows manual assignment of the reservation ID, which can lead to conflicts and issues with ID uniqueness. The system is designed to automatically handle unique ID assignment, and other constructors should be used for creating reservation objects to ensure proper handling of IDs.*
    This constructor is marked with [JsonConstructor] so it will be used for JSON deserialization purposes, but it should not be used when creating new reservation objects manually.

- bool CheckIn ()

    Marks the reservation as checked in and updates the status to CheckedIn.

- bool CheckOut ()

    Marks the reservation as checked out and updates the status to CheckedOut.

- PaymentResult MakePayment (decimal paymentAmount, PaymentMethod paymentMethod)

    Makes a payment towards the reservation and adds a new Payment object to the payment list.

- bool IsFullyPaid ()

    Checks if the reservation is fully paid.

- override string ToString ()

    Overridden ToString method to provide reservation information in a readable JSON format.

Properties

- static int LastAssignedId `[get, set]`

    Public getter and setter for the last assigned ID.

- int Id `[get]`

    Gets the Reservation ID.

- int ClientId `[get]`

    Gets the Client ID associated with the reservation.

- int AccommodationId `[get]`

    Gets the Accommodation ID associated with the reservation.

- int RoomId `[get]`

    Gets the room ID associated with the reservation.

- AccommodationType AccommodationType `[get, set]`

    Gets or sets the Accommodation Type.

- DateTime CheckInDate `[get, set]`

    Gets or sets the Check-In Date.

- DateTime CheckOutDate `[get, set]`
    Gets or sets the Check-Out Date.
- ReservationStatus Status `[get, set]`
    Gets or sets the Reservation Status.
- decimal TotalCost `[get, set]`
    Gets or sets the Total Cost.
- decimal AmountPaid `[get, set]`
    Gets or sets the Amount Paid towards the reservation.
- List< Payment > Payments `[get]`
    Gets a deep copy of the list of payments made towards the reservation.

6.18.1   Detailed Description

Defines the Reservation class, which encapsulates reservation details such as client ID, accommodation type, dates, and payment information. This class ensures data consistency by validating input parameters upon creation or when modifying specific properties.

Definition at line 31 of file Reservation.cs.

6.18.2   Constructor & Destructor Documentation

6.18.2.1   Reservation() [1/3]

```
SmartStay.Core.Models.Reservation.Reservation ( )   [inline]
```

Initializes a new instance of the Reservation class.

This constructor is required for Protobuf-net serialization/deserialization.

It should **not** be used directly in normal application code. Instead, use the constructor with parameters for creating instances of Reservation.

Definition at line 128 of file Reservation.cs.

6.18.2.2   Reservation() [2/3]

```
SmartStay.Core.Models.Reservation.Reservation (
            int clientId,
            int accommodationId,
            int roomId,
            AccommodationType accommodationType,
            DateTime checkInDate,
            DateTime checkOutDate,
            decimal totalCost )   [inline]
```

Constructor to initialize a new reservation with essential details. Validates the input parameters.

Parameters

clientId	The ID of the client.
accommodationId	The ID of the accommodation.
roomId	The ID of the room.
accommodationType	The type of accommodation.
checkInDate	The check-in date.
checkOutDate	The check-out date.

Exceptions

ValidationException	Thrown when any of the input parameters are invalid. Each validation has a specific error code: **InvalidId:** if the client, accommodation or room ID is invalid. **InvalidTotalCost:** if total cost is invalid. **InvalidDateRange:** if the check-in date is later than the check-out date.

Definition at line 151 of file Reservation.cs.

6.18.2.3   Reservation() [3/3]

```
SmartStay.Core.Models.Reservation.Reservation (
            int id,
            int clientId,
            int accommodationId,
            int roomId,
            AccommodationType accommodationType,
            DateTime checkInDate,
            DateTime checkOutDate,
            ReservationStatus status,
            decimal totalCost,
            decimal amountPaid,
            List< Payment > payments )   [inline]
```

Constructor to initialize a new Reservation with all details, including a manually specified ID, client ID, accommodation ID, room ID, accommodation type, check-in and check-out dates, reservation status, total cost, amount paid, and associated payments. **This constructor should be avoided in normal cases** as it allows manual assignment of the reservation ID, which can lead to conflicts and issues with ID uniqueness. The system is designed to automatically handle unique ID assignment, and other constructors should be used for creating reservation objects to ensure proper handling of IDs.
This constructor is marked with [JsonConstructor] so it will be used for JSON deserialization purposes, but it should not be used when creating new reservation objects manually.

Parameters

id	The manually specified ID of the reservation. This should not be used under normal circumstances as the system handles ID assignment automatically.
clientId	The ID of the client making the reservation.
accommodationId	The ID of the accommodation being reserved.
roomId	The ID of the room being reserved.
accommodationType	The type of accommodation being reserved (e.g., hotel, apartment, etc.).
checkInDate	The check-in date for the reservation.
checkOutDate	The check-out date for the reservation.
status	The status of the reservation (e.g., Pending, Confirmed, Cancelled, etc.).
totalCost	The total cost of the reservation.
amountPaid	The amount that has been paid for the reservation.
payments	The list of payments associated with the reservation.

Definition at line 192 of file Reservation.cs.

6.18.3 Member Function Documentation

6.18.3.1 CheckIn()

```
bool SmartStay.Core.Models.Reservation.CheckIn ( )  [inline]
```

Marks the reservation as checked in and updates the status to CheckedIn.

Returns

True if the reservation status was successfully updated to CheckedIn; false if the current status is not Pending.

This method will not modify the reservation status if it is not in Pending state. Ensure the status is appropriately validated before calling this method if strict workflows are required.

Definition at line 320 of file Reservation.cs.

6.18.3.2 CheckOut()

```
bool SmartStay.Core.Models.Reservation.CheckOut ( )  [inline]
```

Marks the reservation as checked out and updates the status to CheckedOut.

Returns

True if the reservation status was successfully updated to CheckedOut; false if the current status is not CheckedIn.

This method will not modify the reservation status if it is not in CheckedIn state. Ensure the status is appropriately validated before calling this method if strict workflows are required.

Definition at line 341 of file Reservation.cs.

6.18.3.3 IsFullyPaid()

```
bool SmartStay.Core.Models.Reservation.IsFullyPaid ( )  [inline]
```

Checks if the reservation is fully paid.

Returns

True if the amount paid equals the total cost, otherwise false.

Definition at line 388 of file Reservation.cs.

6.18.3.4 MakePayment()

```
PaymentResult SmartStay.Core.Models.Reservation.MakePayment (
            decimal paymentAmount,
            PaymentMethod paymentMethod )  [inline]
```

Makes a payment towards the reservation and adds a new Payment object to the payment list.

Parameters

paymentAmount	The amount of the payment. Must be greater than zero.
paymentMethod	The payment method used for the payment.

Returns

A PaymentResult indicating the result of the payment attempt.

This method validates the payment amount and ensures the reservation is not overpaid or already fully paid. The payment method is also validated using the PaymentValidator.ValidatePaymentMethod method.

Definition at line 363 of file Reservation.cs.

6.18.3.5 ToString()

```
override string SmartStay.Core.Models.Reservation.ToString ( )  [inline]
```

Overridden ToString method to provide reservation information in a readable JSON format.

Returns

A JSON string representation of the reservation object.

Definition at line 433 of file Reservation.cs.

6.18.4 Property Documentation

6.18.4.1 AccommodationId

```
int SmartStay.Core.Models.Reservation.AccommodationId  [get]
```

Gets the Accommodation ID associated with the reservation.

Definition at line 235 of file Reservation.cs.

6.18.4.2 AccommodationType

```
AccommodationType SmartStay.Core.Models.Reservation.AccommodationType  [get], [set]
```

Gets or sets the Accommodation Type.

Definition at line 245 of file Reservation.cs.

6.18.4.3 AmountPaid

```
decimal SmartStay.Core.Models.Reservation.AmountPaid  [get], [set]
```

Gets or sets the Amount Paid towards the reservation.

Definition at line 290 of file Reservation.cs.

6.18.4.4 CheckInDate

`DateTime SmartStay.Core.Models.Reservation.CheckInDate  [get], [set]`

Gets or sets the Check-In Date.

Definition at line 254 of file Reservation.cs.

6.18.4.5 CheckOutDate

`DateTime SmartStay.Core.Models.Reservation.CheckOutDate  [get], [set]`

Gets or sets the Check-Out Date.

Definition at line 263 of file Reservation.cs.

6.18.4.6 ClientId

`int SmartStay.Core.Models.Reservation.ClientId  [get]`

Gets the Client ID associated with the reservation.

Definition at line 230 of file Reservation.cs.

6.18.4.7 Id

`int SmartStay.Core.Models.Reservation.Id  [get]`

Gets the Reservation ID.

Definition at line 225 of file Reservation.cs.

6.18.4.8 LastAssignedId

`int SmartStay.Core.Models.Reservation.LastAssignedId  [static], [get], [set]`

Public getter and setter for the last assigned ID.

Definition at line 213 of file Reservation.cs.

6.18.4.9 Payments

`List<Payment> SmartStay.Core.Models.Reservation.Payments  [get]`

Gets a deep copy of the list of payments made towards the reservation.

This property creates and returns a deep copy of the underlying payments collection. Modifications to the returned list or its elements will not affect the original data.

Performance Note: Creating a deep copy can incur a performance cost, especially for large collections. Use this property sparingly if performance is critical.

Definition at line 307 of file Reservation.cs.

6.18.4.10 RoomId

`int SmartStay.Core.Models.Reservation.RoomId [get]`

Gets the room ID associated with the reservation.

Definition at line 240 of file Reservation.cs.

6.18.4.11 Status

`ReservationStatus SmartStay.Core.Models.Reservation.Status [get], [set]`

Gets or sets the Reservation Status.

Definition at line 272 of file Reservation.cs.

6.18.4.12 TotalCost

`decimal SmartStay.Core.Models.Reservation.TotalCost [get], [set]`

Gets or sets the Total Cost.

Definition at line 281 of file Reservation.cs.
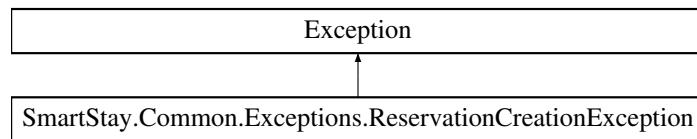
The documentation for this class was generated from the following file:

- Reservation.cs

6.19 SmartStay.Common.Exceptions.ReservationCreationException Class Reference

Represents an error that occurs during the reservation creation process in the SmartStay application. This exception is thrown when there is an issue with validating or processing the reservation's data, such as invalid dates, cost calculations, or availability.

Inheritance diagram for SmartStay.Common.Exceptions.ReservationCreationException:

```
┌─────────────────────────────────────────────────────────────┐
│                          Exception                           │
└─────────────────────────────────────────────────────────────┘
                              ▲
┌─────────────────────────────────────────────────────────────┐
│ SmartStay.Common.Exceptions.ReservationCreationException     │
└─────────────────────────────────────────────────────────────┘
```

Public Member Functions

- ReservationCreationException (string message)

  Initializes a new instance of the ReservationCreationException class with a specified error message.
- ReservationCreationException (string message, Exception innerException)

  Initializes a new instance of the ReservationCreationException class with a specified error message and a reference to the inner exception that is the cause of this exception.
- override string ToString ()

  Returns a string representation of the ReservationCreationException instance, including the error message and any inner exceptions.

Properties

- override string Message `[get]`

  Gets the error message that explains the reason for the exception.

6.19.1 Detailed Description

Represents an error that occurs during the reservation creation process in the SmartStay application. This exception is thrown when there is an issue with validating or processing the reservation's data, such as invalid dates, cost calculations, or availability.

The ReservationCreationException class extends the base Exception class, providing more specific context about errors encountered during the creation of a reservation. This is typically used when validation or other errors occur while trying to create a new reservation object.

Definition at line 26 of file ReservationCreationException.cs.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 ReservationCreationException() [1/2]

```
SmartStay.Common.Exceptions.ReservationCreationException.ReservationCreationException (
            string message )  [inline]
```

Initializes a new instance of the ReservationCreationException class with a specified error message.

Parameters

message	The message that describes the error.

Definition at line 33 of file ReservationCreationException.cs.

6.19.2.2 ReservationCreationException() [2/2]

```
SmartStay.Common.Exceptions.ReservationCreationException.ReservationCreationException (
            string message,
            Exception innerException )  [inline]
```

Initializes a new instance of the ReservationCreationException class with a specified error message and a reference to the inner exception that is the cause of this exception.

Parameters

message	The message that describes the error.
innerException	The exception that is the cause of the current exception.

Definition at line 43 of file ReservationCreationException.cs.

6.19.3 Member Function Documentation

6.19.3.1 ToString()

```
override string SmartStay.Common.Exceptions.ReservationCreationException.ToString ( )  [inline]
```

Returns a string representation of the ReservationCreationException instance, including the error message and any inner exceptions.

Returns

A string that represents the current exception, typically including the error message and any inner exceptions.

For example, the string representation could look like: "Reservation creation failed due to unavailable room for the specified dates."

Definition at line 70 of file ReservationCreationException.cs.

6.19.4 Property Documentation

6.19.4.1 Message

```
override string SmartStay.Common.Exceptions.ReservationCreationException.Message  [get]
```

Gets the error message that explains the reason for the exception.

The error message that describes the reason for the exception.

This property is inherited from the Exception class and can be used to retrieve the message passed when the exception was thrown.

Definition at line 57 of file ReservationCreationException.cs.

The documentation for this class was generated from the following file:

- ReservationCreationException.cs

6.20 SmartStay.Core.Repositories.Reservations Class Reference

Represents a collection of Reservation objects, managed in a dictionary for fast lookup by reservation ID.

Inheritance diagram for SmartStay.Core.Repositories.Reservations:

```
┌─────────────────────────────────────────────────────────────────┐
│ SmartStay.Core.Models.Interfaces.IManageableEntity< Reservation > │
└─────────────────────────────────────────────────────────────────┘
                                 ▲
┌─────────────────────────────────────────────────────────────────┐
│            SmartStay.Core.Repositories.Reservations               │
└─────────────────────────────────────────────────────────────────┘
```

Public Member Functions

- bool Add (Reservation reservation)

  Attempts to add a new reservation to the collection.
- bool Remove (Reservation reservation)

  Removes a reservation from the collection.
- ImportResult Import (string data)

  Imports reservations from a JSON string into the collection, replacing any existing reservations with the same ID.
- string Export ()

  Exports the current list of reservations to a JSON string.
- void Save (string filePath)

  Saves the current state of the reservations collection to a file by serializing the object into a Protobuf format. If an error occurs during the saving process, it will be caught and logged.
- void Load (string filePath)

  Loads the collection from a binary file and assigns it to the current instance. If an error occurs during the loading process, it will be caught and logged.
- Reservation? FindReservationById (int reservationId)

  Finds a reservation by its unique ID.
- IEnumerable< Reservation > FindReservationsByClientId (int clientId)

  Finds all reservations associated with a client by their unique client ID.
- IEnumerable< Reservation > FindReservationsByAccommodationId (int accommodationId)

  Finds all reservations associated with an accommodation by its unique accommodation ID.
- IEnumerable< Reservation > GetFutureReservations (int accommodationId)

  Retrieves all reservations for a given accommodation, with check-in dates after the current time.
- int CountReservations ()

  Counts the number of reservations in the collection.

Public Member Functions inherited from SmartStay.Core.Models.Interfaces.IManageableEntity< Reservation >

- bool Add (T item)

  Adds a single entity of type T to the collection.
- bool Remove (T item)

  Removes a specified entity of type T from the collection.
- ImportResult Import (string data)

  Imports a list of items from a serialized string.
- string Export ()

  Exports the current list of items as a serialized string.
- void Save (string filePath)

  Saves the collection to a binary file.
- void Load (string filePath)

  Loads the collection from a binary file.

6.20.1 Detailed Description

Represents a collection of Reservation objects, managed in a dictionary for fast lookup by reservation ID.

Definition at line 30 of file Reservations.cs.

6.20.2 Member Function Documentation

6.20.2.1 Add()

```
bool SmartStay.Core.Repositories.Reservations.Add (
            Reservation reservation ) [inline]
```

Attempts to add a new reservation to the collection.

Parameters

reservation	The Reservation to add to the collection.

Returns

> `true` if the reservation was successfully added to the collection; `false` if a reservation with the same ID already exists in the collection.

Exceptions

ArgumentNullException	Thrown if *reservation* is `null`.

Definition at line 57 of file Reservations.cs.

6.20.2.2 CountReservations()

```
int SmartStay.Core.Repositories.Reservations.CountReservations ( ) [inline]
```

Counts the number of reservations in the collection.

Returns

> The number of reservations in the collection.

Definition at line 322 of file Reservations.cs.

6.20.2.3 Export()

```
string SmartStay.Core.Repositories.Reservations.Export ( ) [inline]
```

Exports the current list of reservations to a JSON string.

Returns

> A JSON string representation of the reservations in the collection.

Definition at line 137 of file Reservations.cs.

6.20.2.4 FindReservationById()

```
Reservation? SmartStay.Core.Repositories.Reservations.FindReservationById (
            int reservationId ) [inline]
```

Finds a reservation by its unique ID.

Parameters

reservation↩ Id	The unique ID of the reservation to find.

Returns

Returns the Reservation object if found; otherwise, `null`.

Definition at line 273 of file Reservations.cs.

6.20.2.5 FindReservationsByAccommodationId()

```
IEnumerable< Reservation > SmartStay.Core.Repositories.Reservations.FindReservationsByAccommodation↩
Id (
            int accommodationId ) [inline]
```

Finds all reservations associated with an accommodation by its unique accommodation ID.

Parameters

accommodation↩ Id	The unique ID of the accommodation whose reservations to find.

Returns

A list of Reservation objects for the given accommodation. Returns an empty list if no reservations are found.

Definition at line 297 of file Reservations.cs.

6.20.2.6 FindReservationsByClientId()

```
IEnumerable< Reservation > SmartStay.Core.Repositories.Reservations.FindReservationsByClientId
(
            int clientId ) [inline]
```

Finds all reservations associated with a client by their unique client ID.

Parameters

client↩ Id	The unique ID of the client whose reservations to find.

Returns

A list of Reservation objects for the given client.

Definition at line 284 of file Reservations.cs.

6.20.2.7 GetFutureReservations()

IEnumerable< Reservation > SmartStay.Core.Repositories.Reservations.GetFutureReservations (
            int *accommodationId*) [inline]

Retrieves all reservations for a given accommodation, with check-in dates after the current time.

Parameters

accommodation↩ Id	The accommodation ID to filter by.

Returns

A list of future reservations for the given accommodation.

Definition at line 307 of file Reservations.cs.

6.20.2.8 Import()

ImportResult SmartStay.Core.Repositories.Reservations.Import (
            string *data*) [inline]

Imports reservations from a JSON string into the collection, replacing any existing reservations with the same ID.

Parameters

data	The JSON string containing the list of reservations.

Returns

An ImportResult summarizing the outcome of the import operation.

Exceptions

ArgumentException	Thrown if the data is null or empty.
ArgumentException	Thrown if deserialization of the data fails.

Definition at line 103 of file Reservations.cs.

6.20.2.9 Load()

void SmartStay.Core.Repositories.Reservations.Load (
            string *filePath*) [inline]

Loads the collection from a binary file and assigns it to the current instance. If an error occurs during the loading process, it will be caught and logged.

Parameters

filePath	The file path to load the collection from.

Exceptions

IOException	Thrown when an I/O error occurs while loading the data.
SerializationException	Thrown when a deserialization error occurs while loading the data.

Definition at line 237 of file Reservations.cs.

6.20.2.10 Remove()

```
bool SmartStay.Core.Repositories.Reservations.Remove (
            Reservation reservation )  [inline]
```

Removes a reservation from the collection.

Parameters

reservation	The Reservation to remove from the collection.

Returns

> `true` if the reservation was successfully removed from the collection; `false` if the reservation was not found.

Exceptions

ArgumentNullException	Thrown if *reservation* is `null`.

Definition at line 82 of file Reservations.cs.

6.20.2.11 Save()

```
void SmartStay.Core.Repositories.Reservations.Save (
            string filePath )  [inline]
```

Saves the current state of the reservations collection to a file by serializing the object into a Protobuf format. If an error occurs during the saving process, it will be caught and logged.

Parameters

filePath	The path of the file to save the data.

Exceptions

IOException	Thrown when an I/O error occurs while saving the data.

Exceptions

SerializationException	Thrown when a serialization error occurs while saving the data.

Definition at line 204 of file Reservations.cs.

The documentation for this class was generated from the following file:

- Reservations.cs

6.21 SmartStay.Core.Models.Room Class Reference

Defines the Room class, which encapsulates the details of a room within an accommodation, such as its type, price per night, and reservation details. This class provides methods for updating availability and calculating the total cost for a stay.

Public Member Functions

- Room ()

  Initializes a new instance of the Room class.
- Room (RoomType type, decimal pricePerNight)

  Initializes a new instance of the Room class with the specified details: type and price per night.
- Room (int id, RoomType type, decimal pricePerNight, SortedSet< DateRange > reservationDates)

  Constructor to initialize a new Room with all details, including a manually specified ID, room type, price per night, and a set of reservation dates. **This constructor should be avoided in normal cases** *as it allows manual assignment of the room ID, which can lead to conflicts and issues with ID uniqueness. The system is designed to automatically handle unique ID assignment, and other constructors should be used for creating room objects to ensure proper handling of IDs.*

  This constructor is marked with [JsonConstructor] so it will be used for JSON deserialization purposes, but it should not be used when creating new room objects manually.
- bool IsAvailable (DateTime startDate, DateTime endDate, DateRange? existingReservationRange=null)

  Checks if a given date range is available for a new reservation, ensuring there are no overlaps with existing reservations.
- bool AddReservation (DateTime startDate, DateTime endDate)

  Adds a new reservation to the accommodation.
- bool RemoveReservation (DateTime startDate, DateTime endDate)

  Removes an existing reservation from the accommodation.
- decimal CalculateTotalCost (DateTime startDate, DateTime endDate)

  Calculates the total cost for a given stay duration.
- Room Clone ()

  Creates a deep copy of the current Room instance.
- override string ToString ()

  Overridden ToString method to provide room information in a readable JSON format.

Properties

- static int LastAssignedId [get, set]

  Public getter and setter for the last assigned ID.
- int Id [get]

  Public getter for the room ID.
- RoomType Type [get, set]

  Public getter and setter for the Type.
- decimal PricePerNight [get, set]

  Public getter and setter for the PricePerNight.
- SortedSet< DateRange > ReservationDates [get]

  Gets a deep copy of the collection of reservation dates for the accommodation.

6.21.1 Detailed Description

Defines the Room class, which encapsulates the details of a room within an accommodation, such as its type, price per night, and reservation details. This class provides methods for updating availability and calculating the total cost for a stay.

Definition at line 32 of file Room.cs.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 Room() [1/3]

```
SmartStay.Core.Models.Room.Room ( ) [inline]
```

Initializes a new instance of the Room class.

This constructor is required for Protobuf-net serialization/deserialization.

It should **not** be used directly in normal application code. Instead, use the constructor with parameters for creating instances of Room.

Definition at line 86 of file Room.cs.

6.21.2.2 Room() [2/3]

```
SmartStay.Core.Models.Room.Room (
            RoomType type,
            decimal pricePerNight ) [inline]
```

Initializes a new instance of the Room class with the specified details: type and price per night.

Parameters

type	The type of the room (e.g., Single, Double).
pricePerNight	The nightly price of the room.

Exceptions

ValidationException	Thrown if the room type is invalid.
ValidationException	Thrown if the price per night is invalid.

Definition at line 98 of file Room.cs.

6.21.2.3 Room() [3/3]

```
SmartStay.Core.Models.Room.Room (
            int id,
            RoomType type,
            decimal pricePerNight,
            SortedSet< DateRange > reservationDates )  [inline]
```

Constructor to initialize a new Room with all details, including a manually specified ID, room type, price per night, and a set of reservation dates. **This constructor should be avoided in normal cases** as it allows manual assignment of the room ID, which can lead to conflicts and issues with ID uniqueness. The system is designed to automatically handle unique ID assignment, and other constructors should be used for creating room objects to ensure proper handling of IDs.

This constructor is marked with [JsonConstructor] so it will be used for JSON deserialization purposes, but it should not be used when creating new room objects manually.

Parameters

id	The manually specified ID of the room. This should not be used under normal circumstances as the system handles ID assignment automatically.
type	The type of the room (e.g., Single, Double, Suite).
pricePerNight	The price charged per night for the room.
reservationDates	The list of reserved date ranges for the room.

Definition at line 123 of file Room.cs.

6.21.3 Member Function Documentation

6.21.3.1 AddReservation()

```
bool SmartStay.Core.Models.Room.AddReservation (
            DateTime startDate,
            DateTime endDate )  [inline]
```

Adds a new reservation to the accommodation.

Parameters

startDate	The start date of the reservation.
endDate	The end date of the reservation.

Returns

Returns `true` if the reservation was successfully added. If the date range is unavailable (overlap), returns `false`.

This method adds the reservation to a SortedSet<T> that maintains ordered reservations by date range. If the date range overlaps with an existing reservation, the method will return `false`.

Definition at line 245 of file Room.cs.

6.21.3.2 CalculateTotalCost()

```
decimal SmartStay.Core.Models.Room.CalculateTotalCost (
            DateTime startDate,
            DateTime endDate )  [inline]
```

Calculates the total cost for a given stay duration.

Parameters

startDate	The start date of the stay.
endDate	The end date of the stay.

Returns

The total cost for the stay based on the price per night.

Exceptions

ArgumentException	Thrown when the end date is before the start date.

Definition at line 288 of file Room.cs.

6.21.3.3 Clone()

```
Room SmartStay.Core.Models.Room.Clone ( )  [inline]
```

Creates a deep copy of the current Room instance.

Returns

A new Room instance with identical data to the current instance.

Definition at line 337 of file Room.cs.

6.21.3.4 IsAvailable()

```
bool SmartStay.Core.Models.Room.IsAvailable (
            DateTime startDate,
            DateTime endDate,
            DateRange? existingReservationRange = null )  [inline]
```

Checks if a given date range is available for a new reservation, ensuring there are no overlaps with existing reservations.

Parameters

startDate	The start date of the new reservation.
endDate	The end date of the new reservation.
existingReservationRange	Optional parameter representing an existing reservation that can be ignored during the availability check, used for modifying reservations.

Returns

Returns `true` if the accommodation is available during the specified date range; otherwise, returns `false`.

Exceptions

ArgumentException	Thrown if the *endDate* is less than or equal to *startDate* .

This method uses a SortedSet<T> to efficiently find potential conflicting reservations by leveraging the `Get↩`
`ViewBetween` method, which narrows down the search space to reservations potentially overlapping with the requested dates. Overlapping reservations are identified based on whether the requested range intersects with any existing reservation.

Definition at line 200 of file Room.cs.

6.21.3.5 RemoveReservation()

```
bool SmartStay.Core.Models.Room.RemoveReservation (
            DateTime startDate,
            DateTime endDate )  [inline]
```

Removes an existing reservation from the accommodation.

Parameters

startDate	The start date of the reservation to be removed.
endDate	The end date of the reservation to be removed.

Returns

Returns `true` if the reservation was successfully removed; otherwise, returns `false` if the specified reservation was not found.

This method uses the SortedSet<T>.Remove method to delete a specific reservation by matching its DateRange. It ensures efficient removal operations due to the underlying data structure.

Definition at line 269 of file Room.cs.

6.21.3.6 ToString()

```
override string SmartStay.Core.Models.Room.ToString ( )  [inline]
```

Overridden ToString method to provide room information in a readable JSON format.

Returns

A JSON string representation of the room object.

Definition at line 352 of file Room.cs.

6.21.4 Property Documentation

6.21.4.1 Id

```
int SmartStay.Core.Models.Room.Id  [get]
```

Public getter for the room ID.

Definition at line 147 of file Room.cs.

6.21.4.2 LastAssignedId

```
int SmartStay.Core.Models.Room.LastAssignedId  [static], [get], [set]
```

Public getter and setter for the last assigned ID.

Definition at line 135 of file Room.cs.

6.21.4.3 PricePerNight

```
decimal SmartStay.Core.Models.Room.PricePerNight  [get], [set]
```

Public getter and setter for the PricePerNight.

Definition at line 161 of file Room.cs.

6.21.4.4 ReservationDates

```
SortedSet<DateRange> SmartStay.Core.Models.Room.ReservationDates  [get]
```

Gets a deep copy of the collection of reservation dates for the accommodation.

This property creates and returns a deep copy of the underlying reservation dates collection. Modifications to the returned collection or its elements will not affect the original data.

Performance Note: Creating a deep copy can incur a performance cost, especially for large collections. Use this property sparingly if performance is critical.

Definition at line 178 of file Room.cs.

6.21.4.5 Type

RoomType SmartStay.Core.Models.Room.Type [get], [set]

Public getter and setter for the Type.
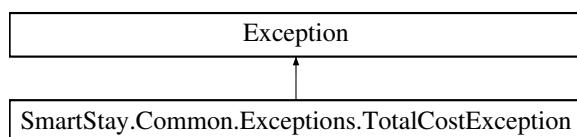
Definition at line 152 of file Room.cs.

The documentation for this class was generated from the following file:

- Room.cs

6.22 SmartStay.Common.Exceptions.TotalCostException Class Reference

Represents an error that occurs during the calculation or validation of the total cost in the SmartStay application. This exception is thrown when there is an issue with calculating the total cost of a reservation, such as invalid dates or incorrect cost calculations.

Inheritance diagram for SmartStay.Common.Exceptions.TotalCostException:

Public Member Functions

- TotalCostException (string message)

   Initializes a new instance of the TotalCostException class with a specified error message.
- TotalCostException (string message, Exception innerException)

   Initializes a new instance of the TotalCostException class with a specified error message and a reference to the inner exception that is the cause of this exception.
- override string ToString ()

   Returns a string representation of the TotalCostException instance, including the error message and any inner exceptions.

Properties

- override string Message [get]

   Gets the error message that explains the reason for the exception.

6.22.1 Detailed Description

Represents an error that occurs during the calculation or validation of the total cost in the SmartStay application. This exception is thrown when there is an issue with calculating the total cost of a reservation, such as invalid dates or incorrect cost calculations.

The TotalCostException class extends the base Exception class, providing more specific context about errors encountered during the calculation or validation of total cost. This is typically used when the cost cannot be calculated due to invalid dates or incorrect pricing.

Definition at line 26 of file TotalCostException.cs.

6.22.2 Constructor & Destructor Documentation

6.22.2.1 TotalCostException() [1/2]

```
SmartStay.Common.Exceptions.TotalCostException.TotalCostException (
            string message )  [inline]
```

Initializes a new instance of the TotalCostException class with a specified error message.

Parameters

message	The message that describes the error.

Definition at line 32 of file TotalCostException.cs.

6.22.2.2 TotalCostException() [2/2]

```
SmartStay.Common.Exceptions.TotalCostException.TotalCostException (
            string message,
            Exception innerException )  [inline]
```

Initializes a new instance of the TotalCostException class with a specified error message and a reference to the inner exception that is the cause of this exception.

Parameters

message	The message that describes the error.
innerException	The exception that is the cause of the current exception.

Definition at line 42 of file TotalCostException.cs.

6.22.3 Member Function Documentation

6.22.3.1 ToString()

```
override string SmartStay.Common.Exceptions.TotalCostException.ToString ( )  [inline]
```

Returns a string representation of the TotalCostException instance, including the error message and any inner exceptions.

Returns

A string that represents the current exception, typically including the error message and any inner exceptions.

For example, the string representation could look like: "Total cost calculation failed due to invalid date range or negative cost."

Definition at line 69 of file TotalCostException.cs.

6.22.4 Property Documentation

6.22.4.1 Message

```
override string SmartStay.Common.Exceptions.TotalCostException.Message [get]
```

Gets the error message that explains the reason for the exception.

The error message that describes the reason for the exception.

This property is inherited from the Exception class and can be used to retrieve the message passed when the exception was thrown.
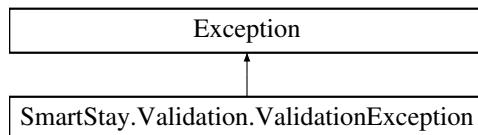
Definition at line 56 of file TotalCostException.cs.

The documentation for this class was generated from the following file:

- TotalCostException.cs

6.23 SmartStay.Validation.ValidationException Class Reference

Represents an exception that is thrown when a validation error occurs. This exception contains an error code that corresponds to a specific validation failure. The error message is retrieved from the resource files based on the error code and the current culture.

Inheritance diagram for SmartStay.Validation.ValidationException:

Public Member Functions

- ValidationException (ValidationErrorCode errorCode)

  Initializes a new instance of the ValidationException class with the provided error code. The error message is automatically fetched based on the error code and the current culture.

Properties

- ValidationErrorCode ErrorCode [get]

  Gets the error code that corresponds to the specific validation failure.

6.23.1 Detailed Description

Represents an exception that is thrown when a validation error occurs. This exception contains an error code that corresponds to a specific validation failure. The error message is retrieved from the resource files based on the error code and the current culture.

Definition at line 24 of file ValidationException.cs.

6.23.2 Constructor & Destructor Documentation

6.23.2.1 ValidationException()

```
SmartStay.Validation.ValidationException.ValidationException (
            ValidationErrorCode errorCode )  [inline]
```

Initializes a new instance of the ValidationException class with the provided error code. The error message is automatically fetched based on the error code and the current culture.

Parameters

errorCode	The error code from the ValidationErrorCode enum that indicates the type of validation error.

Definition at line 37 of file ValidationException.cs.

6.23.3 Property Documentation

6.23.3.1 ErrorCode

```
ValidationErrorCode SmartStay.Validation.ValidationException.ErrorCode  [get]
```

Gets the error code that corresponds to the specific validation failure.

Definition at line 29 of file ValidationException.cs.

The documentation for this class was generated from the following file:

- ValidationException.cs

Chapter 7

File Documentation

7.1 AccommodationType.cs File Reference

Namespaces

- namespace SmartStay
- namespace SmartStay.Common
- namespace SmartStay.Common.Enums

  This namespace contains enumerations used within the SmartStay application.

Enumerations

- enum SmartStay.Common.Enums.AccommodationType {
  SmartStay.Common.Enums.None , SmartStay.Common.Enums.Hotel , SmartStay.Common.Enums.House ,
  SmartStay.Common.Enums.Apartment ,
  SmartStay.Common.Enums.Villa , SmartStay.Common.Enums.BedAndBreakfast , SmartStay.Common.Enums.Hostel
  , SmartStay.Common.Enums.Resort ,
  SmartStay.Common.Enums.Cottage , SmartStay.Common.Enums.Cabin , SmartStay.Common.Enums.Guesthouse
  , SmartStay.Common.Enums.Chalet ,
  SmartStay.Common.Enums.Lodge }

  Enumeration representing different types of accommodations available for booking.

7.2 AccommodationType.cs

Go to the documentation of this file.
```
00001
00010
00014 namespace SmartStay.Common.Enums
00015 {
00019 public enum AccommodationType
00020 {
00024     None,
00025
00029     Hotel,
00030
00034     House,
00035
00040     Apartment,
00041
00045     Villa,
00046
00051     BedAndBreakfast,
```

```
00052
00057     Hostel,
00058
00063     Resort,
00064
00068     Cottage,
00069
00073     Cabin,
00074
00079     Guesthouse,
00080
00085     Chalet,
00086
00090     Lodge
00091 }
00092 }
```

7.3 CancellationResult.cs File Reference

Namespaces

- namespace SmartStay
- namespace SmartStay.Common
- namespace SmartStay.Common.Enums

    This namespace contains enumerations used within the SmartStay application.

Enumerations

- enum SmartStay.Common.Enums.CancellationResult {
  SmartStay.Common.Enums.Success , SmartStay.Common.Enums.ReservationNotFound , SmartStay.Common.Enums.Accom
  , SmartStay.Common.Enums.RoomNotFound ,
  SmartStay.Common.Enums.Error }

    Enumeration representing the possible outcomes of a reservation cancellation attempt.

7.4 CancellationResult.cs

Go to the documentation of this file.

```
00001
00010
00014 namespace SmartStay.Common.Enums
00015 {
00019 public enum CancellationResult
00020 {
00024     Success,
00025
00030     ReservationNotFound,
00031
00035     AccommodationNotFound,
00036
00040     RoomNotFound,
00041
00045     Error
00046 }
00047 }
```

7.5 PaymentMethod.cs File Reference

Namespaces

- namespace SmartStay
- namespace SmartStay.Common
- namespace SmartStay.Common.Enums

    This namespace contains enumerations used within the SmartStay application.

Enumerations

- enum SmartStay.Common.Enums.PaymentMethod {
  SmartStay.Common.Enums.Unchanged , SmartStay.Common.Enums.None , SmartStay.Common.Enums.PayPal
  , SmartStay.Common.Enums.MultiBanco ,
  SmartStay.Common.Enums.BankTransfer }

    Enumeration representing the possible payment methods available for transactions.

7.6 PaymentMethod.cs

Go to the documentation of this file.
```
00001
00010
00014 namespace SmartStay.Common.Enums
00015 {
00019 public enum PaymentMethod
00020 {
00025     Unchanged,
00026
00030     None,
00031
00035     PayPal,
00036
00040     MultiBanco,
00041
00045     BankTransfer
00046 }
00047 }
```

7.7 PaymentResult.cs File Reference

Namespaces

- namespace SmartStay
- namespace SmartStay.Common
- namespace SmartStay.Common.Enums

    This namespace contains enumerations used within the SmartStay application.

Enumerations

- enum SmartStay.Common.Enums.PaymentResult {
  SmartStay.Common.Enums.Success , SmartStay.Common.Enums.InvalidAmount , SmartStay.Common.Enums.AlreadyFullyPa
  , SmartStay.Common.Enums.AmountExceedsTotal ,
  SmartStay.Common.Enums.InvalidPaymentMethod , SmartStay.Common.Enums.Error }

    Enumeration representing the possible outcomes of a payment attempt.

7.8 PaymentResult.cs

Go to the documentation of this file.
```
00001
00010
00014 namespace SmartStay.Common.Enums
00015 {
00019 public enum PaymentResult
00020 {
00024     Success,
00025
00029     InvalidAmount,
00030
00034     AlreadyFullyPaid,
00035
00039     AmountExceedsTotal,
00040
00044     InvalidPaymentMethod,
00045
00049     Error
00050 }
00051 }
```

7.9 PaymentStatus.cs File Reference

Namespaces

- namespace SmartStay
- namespace SmartStay.Common
- namespace SmartStay.Common.Enums

  This namespace contains enumerations used within the SmartStay application.

Enumerations

- enum SmartStay.Common.Enums.PaymentStatus {
  SmartStay.Common.Enums.Unpaid , SmartStay.Common.Enums.Pending , SmartStay.Common.Enums.Completed
  , SmartStay.Common.Enums.PartiallyPaid ,
  SmartStay.Common.Enums.Rejected , SmartStay.Common.Enums.Refunded , SmartStay.Common.Enums.Cancelled
  }

  Enumerator representing payment status.

7.10 PaymentStatus.cs

Go to the documentation of this file.
```
00001
00010
00014 namespace SmartStay.Common.Enums
00015 {
00019 public enum PaymentStatus
00020 {
00024     Unpaid,
00025
00029     Pending,
00030
00034     Completed,
00035
00039     PartiallyPaid,
00040
00044     Rejected,
00045
00049     Refunded,
00050
00054     Cancelled
00055 }
00056 }
```

7.11 RemoveAccommodationResult.cs File Reference

Namespaces

- namespace SmartStay
- namespace SmartStay.Common
- namespace SmartStay.Common.Enums

  This namespace contains enumerations used within the SmartStay application.

Enumerations

- enum SmartStay.Common.Enums.RemoveAccommodationResult {
  SmartStay.Common.Enums.Success , SmartStay.Common.Enums.AccommodationNotFound , SmartStay.Common.Enums.Ow
  , SmartStay.Common.Enums.AccommodationRemovalFailed ,
  SmartStay.Common.Enums.AccommodationDisassociationFailed , SmartStay.Common.Enums.Error }

  *Enumeration representing the results of the accommodation removal process. This enum is used to indicate the
  outcome of the removal operation for an accommodation.*

7.12 RemoveAccommodationResult.cs

Go to the documentation of this file.
```
00001
00010
00014 namespace SmartStay.Common.Enums
00015 {
00020 public enum RemoveAccommodationResult
00021 {
00025     Success,
00026
00030     AccommodationNotFound,
00031
00035     OwnerNotFound,
00036
00040     AccommodationRemovalFailed,
00041
00045     AccommodationDisassociationFailed,
00046
00050     Error
00051 }
00052 }
```

7.13 ReservationStatus.cs File Reference

Namespaces

- namespace SmartStay
- namespace SmartStay.Common
- namespace SmartStay.Common.Enums

  This namespace contains enumerations used within the SmartStay application.

Enumerations

- enum SmartStay.Common.Enums.ReservationStatus {
  SmartStay.Common.Enums.Pending , SmartStay.Common.Enums.CheckedIn , SmartStay.Common.Enums.CheckedOut
  , SmartStay.Common.Enums.Cancelled ,
  SmartStay.Common.Enums.NoShow , SmartStay.Common.Enums.Confirmed , SmartStay.Common.Enums.Declined
  }

  Enumeration representing the current status of a reservation.

7.14 ReservationStatus.cs

Go to the documentation of this file.
```
00001
00010
00014 namespace SmartStay.Common.Enums
00015 {
00019 public enum ReservationStatus
00020 {
00024     Pending,
00025
00029     CheckedIn,
00030
00034     CheckedOut,
00035
00039     Cancelled,
00040
00044     NoShow,
00045
00049     Confirmed,
00050
00054     Declined
00055 }
00056 }
```

7.15 RoomType.cs File Reference

Namespaces

- namespace SmartStay
- namespace SmartStay.Common
- namespace SmartStay.Common.Enums

    This namespace contains enumerations used within the SmartStay application.

Enumerations

- enum SmartStay.Common.Enums.RoomType {
  SmartStay.Common.Enums.None , SmartStay.Common.Enums.Single , SmartStay.Common.Enums.Double
  , SmartStay.Common.Enums.Twin ,
  SmartStay.Common.Enums.Suite , SmartStay.Common.Enums.Family , SmartStay.Common.Enums.Studio
  , SmartStay.Common.Enums.Deluxe ,
  SmartStay.Common.Enums.Penthouse , SmartStay.Common.Enums.Dormitory , SmartStay.Common.Enums.Accessible
  , SmartStay.Common.Enums.PresidentialSuite }

    Enumeration representing different types of rooms available within accommodations.

7.16 RoomType.cs

Go to the documentation of this file.
```
00001
00010
00014 namespace SmartStay.Common.Enums
00015 {
00019 public enum RoomType
00020 {
00024     None,
00025
00029     Single,
00030
00034     Double,
00035
00039     Twin,
00040
```

```
00044      Suite,
00045
00049      Family,
00050
00055      Studio,
00056
00060      Deluxe,
00061
00065      Penthouse,
00066
00070      Dormitory,
00071
00076      Accessible,
00077
00082      PresidentialSuite
00083 }
00084 }
```

7.17 UpdateAccommodationResult.cs File Reference

Namespaces

- namespace SmartStay
- namespace SmartStay.Common
- namespace SmartStay.Common.Enums

    This namespace contains enumerations used within the SmartStay application.

Enumerations

- enum SmartStay.Common.Enums.UpdateAccommodationResult {
    SmartStay.Common.Enums.Success , SmartStay.Common.Enums.AccommodationNotFound , SmartStay.Common.Enums.Inv
    , SmartStay.Common.Enums.InvalidName ,
    SmartStay.Common.Enums.InvalidAddress , SmartStay.Common.Enums.Error }

    Enumeration representing the results of the accommodation update process. This enum is used to indicate the outcome of the update operation for an accommodation.

7.18 UpdateAccommodationResult.cs

Go to the documentation of this file.
```
00001
00010
00014 namespace SmartStay.Common.Enums
00015 {
00020 public enum UpdateAccommodationResult
00021 {
00025      Success,
00026
00030      AccommodationNotFound,
00031
00035      InvalidType,
00036
00040      InvalidName,
00041
00045      InvalidAddress,
00046
00050      Error
00051 }
00052 }
```

7.19 UpdateClientResult.cs File Reference

Namespaces

- namespace SmartStay
- namespace SmartStay.Common
- namespace SmartStay.Common.Enums

    This namespace contains enumerations used within the SmartStay application.

Enumerations

- enum SmartStay.Common.Enums.UpdateClientResult {
  SmartStay.Common.Enums.Success , SmartStay.Common.Enums.ClientNotFound , SmartStay.Common.Enums.InvalidFirstNa
  , SmartStay.Common.Enums.InvalidLastName ,
  SmartStay.Common.Enums.InvalidEmail , SmartStay.Common.Enums.InvalidPhoneNumber , SmartStay.Common.Enums.Inval
  , SmartStay.Common.Enums.InvalidPaymentMethod ,
  SmartStay.Common.Enums.Error }

    *Enumeration representing the results of the client update process. This enum is used to indicate the outcome of the
    update operation for a client.*

7.20 UpdateClientResult.cs

Go to the documentation of this file.
```
00001
00010
00014 namespace SmartStay.Common.Enums
00015 {
00020 public enum UpdateClientResult
00021 {
00025     Success,
00026
00030     ClientNotFound,
00031
00035     InvalidFirstName,
00036
00040     InvalidLastName,
00041
00045     InvalidEmail,
00046
00050     InvalidPhoneNumber,
00051
00055     InvalidAddress,
00056
00060     InvalidPaymentMethod,
00061
00065     Error
00066 }
00067 }
```

7.21 UpdateOwnerResult.cs File Reference

Namespaces

- namespace SmartStay
- namespace SmartStay.Common
- namespace SmartStay.Common.Enums

    This namespace contains enumerations used within the SmartStay application.

Enumerations

- enum [SmartStay.Common.Enums.UpdateOwnerResult](#) {
  [SmartStay.Common.Enums.Success](#) , [SmartStay.Common.Enums.OwnerNotFound](#) , [SmartStay.Common.Enums.InvalidFirstN](#)
  , [SmartStay.Common.Enums.InvalidLastName](#) ,
  [SmartStay.Common.Enums.InvalidEmail](#) , [SmartStay.Common.Enums.InvalidPhoneNumber](#) , [SmartStay.Common.Enums.Inval](#)
  }

  Enum representing the result of an owner update operation.

7.22 UpdateOwnerResult.cs

[Go to the documentation of this file.](#)
```
00001
00010
00014 namespace SmartStay.Common.Enums
00015 {
00019 public enum UpdateOwnerResult
00020 {
00024     Success,
00025
00029     OwnerNotFound,
00030
00034     InvalidFirstName,
00035
00039     InvalidLastName,
00040
00044     InvalidEmail,
00045
00049     InvalidPhoneNumber,
00050
00054     InvalidAddress
00055 }
00056
00057 }
```

7.23 UpdateReservationResult.cs File Reference

Namespaces

- namespace [SmartStay](#)
- namespace [SmartStay.Common](#)
- namespace [SmartStay.Common.Enums](#)

  This namespace contains enumerations used within the [SmartStay](#) application.

Enumerations

- enum [SmartStay.Common.Enums.UpdateReservationResult](#) {
  [SmartStay.Common.Enums.Success](#) , [SmartStay.Common.Enums.ReservationNotFound](#) , [SmartStay.Common.Enums.Accom](#)
  , [SmartStay.Common.Enums.RoomNotFound](#) ,
  [SmartStay.Common.Enums.RoomIsNull](#) , [SmartStay.Common.Enums.DatesUnavailable](#) , [SmartStay.Common.Enums.InvalidDa](#)
  , [SmartStay.Common.Enums.Error](#) }

  Enumeration representing the results of the reservation update process. This enum is used to indicate the outcome of the update operation for a reservation.

7.24 UpdateReservationResult.cs

Go to the documentation of this file.
```
00001
00010
00014 namespace SmartStay.Common.Enums
00015 {
00020 public enum UpdateReservationResult
00021 {
00025     Success,
00026
00030     ReservationNotFound,
00031
00035     AccommodationNotFound,
00036
00040     RoomNotFound,
00041
00045     RoomIsNull,
00046
00050     DatesUnavailable,
00051
00055     InvalidDates,
00056
00060     Error
00061 }
00062 }
```

7.25 AccommodationCreationException.cs File Reference

Data Structures

- class SmartStay.Common.Exceptions.AccommodationCreationException

  Represents an error that occurs during the accommodation creation process in the SmartStay application. This exception is thrown when there is an issue with validating or processing the accommodation's data.

Namespaces

- namespace SmartStay
- namespace SmartStay.Common
- namespace SmartStay.Common.Exceptions

  This namespace contains custom exceptions used within the SmartStay application.

7.26 AccommodationCreationException.cs

Go to the documentation of this file.
```
00001
00010
00014 namespace SmartStay.Common.Exceptions
00015 {
00025 public class AccommodationCreationException : Exception
00026 {
00032     public AccommodationCreationException(string message) : base(message)
00033     {
00034     }
00035
00042     public AccommodationCreationException(string message, Exception innerException) : base(message,
    innerException)
00043     {
00044     }
00045
00056     public override string Message => base.Message;
00057
00069     public override string ToString()
00070     {
00071         return $"{base.ToString()}";
00072     }
00073 }
00074 }
```

7.27 AddAccommodationSystemException.cs File Reference

Data Structures

- class SmartStay.Common.Exceptions.AddAccommodationSystemException

  Represents an error that occurs when an accommodation cannot be added to the system. This exception is thrown when there is an issue during the addition process, such as conflicts, system-level errors, or other reasons preventing the accommodation from being added.

Namespaces

- namespace SmartStay
- namespace SmartStay.Common
- namespace SmartStay.Common.Exceptions

  This namespace contains custom exceptions used within the SmartStay application.

7.28 AddAccommodationSystemException.cs

Go to the documentation of this file.
```
00001
00010
00014 namespace SmartStay.Common.Exceptions
00015 {
00025 public class AddAccommodationSystemException : Exception
00026 {
00032     public AddAccommodationSystemException(string message) : base(message)
00033     {
00034     }
00035
00042     public AddAccommodationSystemException(string message, Exception innerException) : base(message,
     innerException)
00043     {
00044     }
00045
00056     public override string Message => base.Message;
00057
00069     public override string ToString()
00070     {
00071         return $"{base.ToString()}";
00072     }
00073 }
00074 }
```

7.29 ClientCreationException.cs File Reference

Data Structures

- class SmartStay.Common.Exceptions.ClientCreationException

  Represents an error that occurs during the client creation process in the SmartStay application. This exception is thrown when there is an issue with validating or processing the client's data.

Namespaces

- namespace SmartStay
- namespace SmartStay.Common
- namespace SmartStay.Common.Exceptions

  This namespace contains custom exceptions used within the SmartStay application.

7.30 ClientCreationException.cs

Go to the documentation of this file.
```
00001
00010
00014 namespace SmartStay.Common.Exceptions
00015 {
00025 public class ClientCreationException : Exception
00026 {
00031     public ClientCreationException(string message) : base(message)
00032     {
00033     }
00034
00041     public ClientCreationException(string message, Exception innerException) : base(message,
     innerException)
00042     {
00043     }
00044
00055     public override string Message => base.Message;
00056
00068     public override string ToString()
00069     {
00070         return $"{base.ToString()}";
00071     }
00072 }
00073 }
```

7.31 EntityNotFoundException.cs File Reference

Data Structures

- class SmartStay.Common.Exceptions.EntityNotFoundException

  Represents an error that occurs when an entity is not found in the system. This exception is thrown when an operation cannot proceed because the specified entity (e.g., Reservation, Room) does not exist in the system.

Namespaces

- namespace SmartStay
- namespace SmartStay.Common
- namespace SmartStay.Common.Exceptions

  This namespace contains custom exceptions used within the SmartStay application.

7.32 EntityNotFoundException.cs

Go to the documentation of this file.
```
00001
00010
00014 namespace SmartStay.Common.Exceptions
00015 {
00026 public class EntityNotFoundException : Exception
00027 {
00038     public string EntityType { get; }
00039
00050     public int EntityId { get; }
00051
00058     public EntityNotFoundException(string entityType, int entityId)
00059         : base($"{entityType} with ID {entityId} was not found.")
00060     {
00061         EntityType = entityType;
00062         EntityId = entityId;
00063     }
00064
00072     public EntityNotFoundException(string entityType, int entityId, string message)
00073         : base($"{entityType} with ID {entityId} was not found. {message}")
00074     {
```

```
00075            EntityType = entityType;
00076            EntityId = entityId;
00077        }
00078
00087      public EntityNotFoundException(string entityType, int entityId, string message, Exception
      innerException)
00088            : base($"{entityType} with ID {entityId} was not found. {message}", innerException)
00089        {
00090            EntityType = entityType;
00091            EntityId = entityId;
00092        }
00093
00105      public override string ToString()
00106        {
00107            return $"{base.ToString()}";
00108        }
00109 }
00110 }
```

7.33 OwnerAddAccommodationException.cs File Reference

Data Structures

- class SmartStay.Common.Exceptions.OwnerAddAccommodationException

  Represents an error that occurs when an accommodation cannot be added to an owner's list of accommodations. This exception is thrown when there is an issue with the adding process, such as validation failures, conflicts, or other reasons why the accommodation cannot be associated with the owner.

Namespaces

- namespace SmartStay
- namespace SmartStay.Common
- namespace SmartStay.Common.Exceptions

  This namespace contains custom exceptions used within the SmartStay application.

7.34 OwnerAddAccommodationException.cs

Go to the documentation of this file.
```
00001
00010
00014 namespace SmartStay.Common.Exceptions
00015 {
00025 public class OwnerAddAccommodationException : Exception
00026 {
00032      public OwnerAddAccommodationException(string message) : base(message)
00033        {
00034        }
00035
00042      public OwnerAddAccommodationException(string message, Exception innerException) : base(message,
      innerException)
00043        {
00044        }
00045
00056      public override string Message => base.Message;
00057
00069      public override string ToString()
00070        {
00071            return $"{base.ToString()}";
00072        }
00073 }
00074 }
```

7.35 OwnerCreationException.cs File Reference

Data Structures

- class SmartStay.Common.Exceptions.OwnerCreationException

  Represents an error that occurs during the owner creation process in the SmartStay application. This exception is thrown when there is an issue with validating or processing the owner's data.

Namespaces

- namespace SmartStay
- namespace SmartStay.Common
- namespace SmartStay.Common.Exceptions

  This namespace contains custom exceptions used within the SmartStay application.

7.36 OwnerCreationException.cs

Go to the documentation of this file.
```
00001
00010
00014 namespace SmartStay.Common.Exceptions
00015 {
00025 public class OwnerCreationException : Exception
00026 {
00031     public OwnerCreationException(string message) : base(message)
00032     {
00033     }
00034
00041     public OwnerCreationException(string message, Exception innerException) : base(message,
      innerException)
00042     {
00043     }
00044
00055     public override string Message => base.Message;
00056
00068     public override string ToString()
00069     {
00070         return $"{base.ToString()}";
00071     }
00072 }
00073 }
```

7.37 ReservationCreationException.cs File Reference

Data Structures

- class SmartStay.Common.Exceptions.ReservationCreationException

  Represents an error that occurs during the reservation creation process in the SmartStay application. This exception is thrown when there is an issue with validating or processing the reservation's data, such as invalid dates, cost calculations, or availability.

Namespaces

- namespace SmartStay
- namespace SmartStay.Common
- namespace SmartStay.Common.Exceptions

  This namespace contains custom exceptions used within the SmartStay application.

7.38 ReservationCreationException.cs

Go to the documentation of this file.
```
00001
00010
00014 namespace SmartStay.Common.Exceptions
00015 {
00026 public class ReservationCreationException : Exception
00027 {
00033     public ReservationCreationException(string message) : base(message)
00034     {
00035     }
00036
00043     public ReservationCreationException(string message, Exception innerException) : base(message,
     innerException)
00044     {
00045     }
00046
00057     public override string Message => base.Message;
00058
00070     public override string ToString()
00071     {
00072         return $"{base.ToString()}";
00073     }
00074 }
00075 }
```

7.39 TotalCostException.cs File Reference

Data Structures

- class SmartStay.Common.Exceptions.TotalCostException

  Represents an error that occurs during the calculation or validation of the total cost in the SmartStay application. This exception is thrown when there is an issue with calculating the total cost of a reservation, such as invalid dates or incorrect cost calculations.

Namespaces

- namespace SmartStay
- namespace SmartStay.Common
- namespace SmartStay.Common.Exceptions

  This namespace contains custom exceptions used within the SmartStay application.

7.40 TotalCostException.cs

Go to the documentation of this file.
```
00001
00010
00014 namespace SmartStay.Common.Exceptions
00015 {
00026 public class TotalCostException : Exception
00027 {
00032     public TotalCostException(string message) : base(message)
00033     {
00034     }
00035
00042     public TotalCostException(string message, Exception innerException) : base(message,
     innerException)
00043     {
00044     }
00045
00056     public override string Message => base.Message;
00057
00069     public override string ToString()
00070     {
00071         return $"{base.ToString()}";
00072     }
00073 }
00074 }
```

7.41 ImportResult.cs File Reference

Data Structures

- class SmartStay.Common.Models.ImportResult

    Represents the result of an accommodation import operation, summarizing the outcome of the process.

Namespaces

- namespace SmartStay
- namespace SmartStay.Common
- namespace SmartStay.Common.Models

    This namespace contains common models used within the SmartStay application.

7.42 ImportResult.cs

Go to the documentation of this file.
```
00001
00010
00014 namespace SmartStay.Common.Models
00015 {
00019 public class ImportResult
00020 {
00024     public int ImportedCount { get; set; }
00025
00029     public int ReplacedCount { get; set; }
00030
00035     public int TotalCount => ImportedCount + ReplacedCount;
00036
00044     public override string ToString()
00045     {
00046         return $"Imported: {ImportedCount}, Replaced: {ReplacedCount}, Total: {TotalCount}";
00047     }
00048 }
00049 }
```

7.43 SmartStay.Common/obj/Debug/net8.0/.NETCore↩ App,Version=v8.0.AssemblyAttributes.cs File Reference

7.44 SmartStay.Common/obj/Debug/net8.0/.NETCore↩ App,Version=v8.0.AssemblyAttributes.cs

Go to the documentation of this file.
```
00001 // <autogenerated />
00002 using System;
00003 using System.Reflection;
00004 [assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v8.0",
     FrameworkDisplayName = ".NET 8.0")]
```

7.45 SmartStay.Core/obj/Debug/net8.0/.NETCore↩ App,Version=v8.0.AssemblyAttributes.cs File Reference

7.46 SmartStay.Core/obj/Debug/net8.0/.NETCore↩ App,Version=v8.0.AssemblyAttributes.cs

Go to the documentation of this file.
```
00001 // <autogenerated />
00002 using System;
00003 using System.Reflection;
00004 [assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v8.0",
      FrameworkDisplayName = ".NET 8.0")]
```

7.47 SmartStay.IO/obj/Debug/net8.0/.NETCore↩ App,Version=v8.0.AssemblyAttributes.cs File Reference

7.48 SmartStay.IO/obj/Debug/net8.0/.NETCore↩ App,Version=v8.0.AssemblyAttributes.cs

Go to the documentation of this file.
```
00001 // <autogenerated />
00002 using System;
00003 using System.Reflection;
00004 [assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v8.0",
      FrameworkDisplayName = ".NET 8.0")]
```

7.49 SmartStay.Validation/obj/Debug/net8.0/.NETCore↩ App,Version=v8.0.AssemblyAttributes.cs File Reference

7.50 SmartStay.Validation/obj/Debug/net8.0/.NETCore↩ App,Version=v8.0.AssemblyAttributes.cs

Go to the documentation of this file.
```
00001 // <autogenerated />
00002 using System;
00003 using System.Reflection;
00004 [assembly: global::System.Runtime.Versioning.TargetFrameworkAttribute(".NETCoreApp,Version=v8.0",
      FrameworkDisplayName = ".NET 8.0")]
```

7.51 SmartStay.Common.AssemblyInfo.cs File Reference

7.52 SmartStay.Common.AssemblyInfo.cs

Go to the documentation of this file.
```
00001 //------------------------------------------------------------------------------
00002 // <auto-generated>
00003 //     This code was generated by a tool.
```

```
00004 //     Runtime Version:4.0.30319.42000
00005 //
00006 //     Changes to this file may cause incorrect behavior and will be lost if
00007 //     the code is regenerated.
00008 // </auto-generated>
00009 //------------------------------------------------------------------------------
00010
00011 using System;
00012 using System.Reflection;
00013
00014 [assembly: System.Reflection.AssemblyCompanyAttribute("SmartStay.Common")]
00015 [assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]
00016 [assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]
00017 [assembly:
      System.Reflection.AssemblyInformationalVersionAttribute("1.0.0+c366ac03947932e5126b804e73253b4d5f5e0e8d")]
00018 [assembly: System.Reflection.AssemblyProductAttribute("SmartStay.Common")]
00019 [assembly: System.Reflection.AssemblyTitleAttribute("SmartStay.Common")]
00020 [assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]
00021
00022 // Generated by the MSBuild WriteCodeFragment class.
00023
```

7.53 SmartStay.Common.GlobalUsings.g.cs File Reference

7.54 SmartStay.Common.GlobalUsings.g.cs

Go to the documentation of this file.

```
00001 // <auto-generated/>
00002 global using global::System;
00003 global using global::System.Collections.Generic;
00004 global using global::System.IO;
00005 global using global::System.Linq;
00006 global using global::System.Net.Http;
00007 global using global::System.Threading;
00008 global using global::System.Threading.Tasks;
```

7.55 Accommodation.cs File Reference

Data Structures

- class SmartStay.Core.Models.Accommodation

  Defines the Accommodation class, which encapsulates the details of an accommodation, such as its type, name, address, nightly price, and availability status. This class provides methods to update availability and calculate total cost.

Namespaces

- namespace SmartStay
- namespace SmartStay.Core
- namespace SmartStay.Core.Models

  The *SmartStay.Core.Models* *namespace contains the primary data models used within the* *SmartStay* *application. These models represent core entities and structures essential for managing application data.*

7.56 Accommodation.cs

Go to the documentation of this file.
```
00001
00011 using System.Text.Encodings.Web;
00012 using System.Text.Json;
00013 using System.Text.Json.Serialization;
00014 using ProtoBuf;
00015 using SmartStay.Common.Enums;
00016 using SmartStay.Validation;
00017 using SmartStay.Validation.Validators;
00018
00023 namespace SmartStay.Core.Models
00024 {
00030 [ProtoContract]
00031 public class Accommodation
00032 {
00036     static int _lastAccommodationId = 0; // Last assigned accommodation ID
00037
00050     static readonly JsonSerializerOptions _jsonOptions =
00051         new() { WriteIndented = true, Encoder = JavaScriptEncoder.UnsafeRelaxedJsonEscaping,
00052                 Converters = { new JsonStringEnumConverter() } };
00053
00057     [ProtoMember(1)]
00058     readonly int _id; // ID of the accommodation
00059
00063     [ProtoMember(2)]
00064     int _ownerId; // ID of the owner
00065
00069     [ProtoMember(3)]
00070     AccommodationType _type; // Type of accommodation (Hotel, House, etc.)
00071
00075     [ProtoMember(4)]
00076     string _name; // Name of the accommodation
00077
00081     [ProtoMember(5)]
00082     string _address; // Address of the accommodation
00083
00088     [ProtoMember(6)]
00089     readonly List<Room> _rooms = []; // List of rooms
00090
00097 #pragma warning disable CS8618
00098     public Accommodation()
00099 #pragma warning restore CS8618
00100     {
00101         // This constructor is intentionally empty and only needed for Protobuf-net deserialization.
00102     }
00103
00123     public Accommodation(int ownerId, AccommodationType type, string name, string address)
00124     {
00125         AccommodationValidator.ValidateAccommodationType(type);
00126         NameValidator.ValidateAccommodationName(name);
00127         AddressValidator.ValidateAddress(address);
00128
00129         _id = GenerateAccommodationId();
00130         _ownerId = ownerId;
00131         _type = type;
00132         _name = name;
00133         _address = address;
00134     }
00135
00156     [JsonConstructor]
00157     public Accommodation(int id, int ownerId, AccommodationType type, string name, string address,
     List<Room> rooms)
00158     {
00159         _id = id;
00160         UpdateLastAccommodationId(id);
00161         _ownerId = ownerId;
00162         _type = type;
00163         _name = name;
00164         _address = address;
00165         _rooms = rooms;
00166     }
00167
00171     public static int LastAssignedId
00172     {
00173         get => _lastAccommodationId;
00174         set {
00175             if (_lastAccommodationId < value)
00176                 _lastAccommodationId = value;
00177         }
00178     }
00179
00183     public int Id => _id;
00184
```

```
00188      public int OwnerId
00189      {
00190          get => _ownerId;
00191          set => _ownerId = OwnerValidator.ValidateOwnerId(value);
00192      }
00193
00197      public AccommodationType Type
00198      {
00199          get => _type;
00200          set => _type = AccommodationValidator.ValidateAccommodationType(value);
00201      }
00202
00206      public string Name
00207      {
00208          get => _name;
00209          set => _name = NameValidator.ValidateAccommodationName(value);
00210      }
00211
00215      public string Address
00216      {
00217          get => _address;
00218          set => _address = AddressValidator.ValidateAddress(value);
00219      }
00220
00232      public List<Room> Rooms => GetRoomsCopy();
00233
00239      public Room? FindRoomById(int roomId)
00240      {
00241          return _rooms.Find(room => room.Id == roomId);
00242      }
00243
00249      public bool AddRoom(Room room)
00250      {
00251          if (room == null)
00252          {
00253              return false; // If the room is null, return false
00254          }
00255
00256          _rooms.Add(room);
00257          return true; // Successfully added the room
00258      }
00259
00265      public bool DeleteRoom(int roomId)
00266      {
00267          var roomToDelete = _rooms.Find(r => r.Id == roomId);
00268
00269          if (roomToDelete == null)
00270          {
00271              return false; // Return false if the room with the given ID is not found
00272          }
00273
00274          _rooms.Remove(roomToDelete);
00275          return true; // Successfully removed the room
00276      }
00277
00283      private static int GenerateAccommodationId()
00284      {
00285          // Check if the current value exceeds the max limit of int (2,147,483,647)
00286          if (_lastAccommodationId >= int.MaxValue)
00287          {
00288              throw new InvalidOperationException("Accommodation ID limit exceeded.");
00289          }
00290
00291          return Interlocked.Increment(ref _lastAccommodationId);
00292      }
00293
00298      private static void UpdateLastAccommodationId(int id)
00299      {
00300          if (id > _lastAccommodationId)
00301          {
00302              _lastAccommodationId = id; // Update the last assigned client ID if the new ID is larger
00303          }
00304      }
00305
00310      private List<Room> GetRoomsCopy()
00311      {
00312          // Deep copy each room
00313          return _rooms.Select(room => room.Clone()).ToList();
00314      }
00315
00320      public Accommodation Clone()
00321      {
00322          // Create a new instance of Accommodation and deep copy the fields
00323          return new Accommodation(_id,                                    // Immutable
00324                                   _ownerId,                               // Immutable
00325                                   _type,                                  // Enum, can be
       directly copied
```

```
00326                                        _name,                                     // String, can be
     directly copied
00327                                        _address,                                  // String, can be
     directly copied
00328                                        new List<Room>(_rooms.Select(room => room.Clone())) // Deep copy of
     Room objects
00329            );
00330       }
00331
00336     public override string ToString()
00337     {
00338         // Create a dictionary for the properties you want to serialize
00339         var accommodationData = new { Id = _id, Type = _type.ToString(), Name = _name, Address =
     _address,
00340                                        Rooms = _rooms.Select(room => new {
00341                                            room.Id,
00342                                            room.PricePerNight,
00343                                            room.Type,
00344                                        }) };
00345
00346         // Serialize the dictionary into a JSON string, which will include Rooms as an array
00347         return JsonSerializer.Serialize(accommodationData, _jsonOptions);
00348     }
00349 }
00350 }
```

7.57 Client.cs File Reference

Data Structures

- class SmartStay.Core.Models.Client

  Defines the Client class, which encapsulates the details of a client including personal information such as first name,
  last name, email address, phone number, residential address, and preferred payment method. This class validates
  the provided data upon creation or when modifying specific properties, ensuring that all data is consistent and correct.

Namespaces

- namespace SmartStay
- namespace SmartStay.Core
- namespace SmartStay.Core.Models

  The *SmartStay.Core.Models* *namespace contains the primary data models used within the* *SmartStay* *appli-*
  cation. These models represent core entities and structures essential for managing application data.

7.58 Client.cs

Go to the documentation of this file.
```
00001
00011 using System.Text.Encodings.Web;
00012 using System.Text.Json;
00013 using System.Text.Json.Serialization;
00014 using ProtoBuf;
00015 using SmartStay.Common.Enums;
00016 using SmartStay.Core.Repositories;
00017 using SmartStay.Validation.Validators;
00018
00023 namespace SmartStay.Core.Models
00024 {
00032 [ProtoContract]
00033 public class Client
00034 {
00038     static int _lastClientId = 0; // Last assigned client ID
00039
00052     static readonly JsonSerializerOptions _jsonOptions =
00053         new() { WriteIndented = true, Encoder = JavaScriptEncoder.UnsafeRelaxedJsonEscaping,
00054                 Converters = { new JsonStringEnumConverter() } };
00055
```

```
00059     [ProtoMember(1)]
00060     readonly int _id; // ID of the client
00061
00065     [ProtoMember(2)]
00066     string _firstName; // First name of the client
00067
00071     [ProtoMember(3)]
00072     string _lastName; // Last name of the client
00073
00077     [ProtoMember(4)]
00078     string _email; // Email address of the client
00079
00084     [ProtoMember(5)]
00085     string _phoneNumber = string.Empty; // Phone number of the client
00086
00091     [ProtoMember(6)]
00092     string _address = string.Empty; // Address of the client
00093
00098     [ProtoMember(7)]
00099     PaymentMethod _preferredPaymentMethod = PaymentMethod.None; // Preferred payment method of the
      client
00100
00107 #pragma warning disable CS8618
00108     public Client()
00109 #pragma warning restore CS8618
00110     {
00111         // This constructor is intentionally empty and only needed for Protobuf-net deserialization.
00112     }
00113
00127     public Client(string firstName, string lastName, string email)
00128     {
00129         NameValidator.ValidateName(firstName);
00130         NameValidator.ValidateName(lastName);
00131         EmailValidator.ValidateEmail(email);
00132
00133         _id = GenerateClientId();
00134         _firstName = firstName;
00135         _lastName = lastName;
00136         _email = email;
00137     }
00138
00156     public Client(string firstName, string lastName, string email, string phoneNumber, string address)
00157         : this(firstName, lastName, email)
00158     {
00159         PhoneNumberValidator.ValidatePhoneNumber(phoneNumber);
00160         AddressValidator.ValidateAddress(address);
00161
00162         _phoneNumber = phoneNumber;
00163         _address = address;
00164     }
00165
00183     public Client(string firstName, string lastName, string email, string phoneNumber, string address,
00184                 PaymentMethod preferredPaymentMethod)
00185         : this(firstName, lastName, email, phoneNumber, address)
00186     {
00187         PaymentValidator.ValidatePaymentMethod(preferredPaymentMethod);
00188
00189         _preferredPaymentMethod = preferredPaymentMethod;
00190     }
00191
00209     [JsonConstructor]
00210     public Client(int id, string firstName, string lastName, string email, string phoneNumber, string
      address,
00211                 PaymentMethod preferredPaymentMethod)
00212     {
00213         _id = id;
00214         UpdateLastClientId(id);
00215         _firstName = firstName;
00216         _lastName = lastName;
00217         _email = email;
00218         _phoneNumber = phoneNumber;
00219         _address = address;
00220         _preferredPaymentMethod = preferredPaymentMethod;
00221     }
00222
00226     public static int LastAssignedId
00227     {
00228         get => _lastClientId;
00229         set {
00230             if (_lastClientId < value)
00231                 _lastClientId = value;
00232         }
00233     }
00234
00238     public int Id => _id;
00239
00244     public string FirstName
```

```
00245     {
00246         get => _firstName;
00247         set => _firstName = NameValidator.ValidateName(value);
00248     }
00249
00254     public string LastName
00255     {
00256         get => _lastName;
00257         set => _lastName = NameValidator.ValidateName(value);
00258     }
00259
00264     public string Email
00265     {
00266         get => _email;
00267         set => _email = EmailValidator.ValidateEmail(value);
00268     }
00269
00274     public string PhoneNumber
00275     {
00276         get => _phoneNumber;
00277         set => _phoneNumber = PhoneNumberValidator.ValidatePhoneNumber(value);
00278     }
00279
00284     public string Address
00285     {
00286         get => _address;
00287         set => _address = AddressValidator.ValidateAddress(value);
00288     }
00289
00294     public PaymentMethod PreferredPaymentMethod
00295     {
00296         get => _preferredPaymentMethod;
00297         set => _preferredPaymentMethod = PaymentValidator.ValidatePaymentMethod(value);
00298     }
00299
00305     private static int GenerateClientId()
00306     {
00307         // Check if the current value exceeds the max limit of int (2,147,483,647)
00308         if (_lastClientId >= int.MaxValue)
00309         {
00310             throw new InvalidOperationException("Client ID limit exceeded.");
00311         }
00312
00313         return Interlocked.Increment(ref _lastClientId);
00314     }
00315
00320     private static void UpdateLastClientId(int id)
00321     {
00322         if (id > _lastClientId)
00323         {
00324             _lastClientId = id; // Update the last assigned client ID if the new ID is larger
00325         }
00326     }
00327
00332     public override string ToString()
00333     {
00334         return JsonSerializer.Serialize(this, _jsonOptions);
00335     }
00336 }
00337 }
```

7.59 ManageableEntity.cs File Reference

Data Structures

- interface SmartStay.Core.Models.Interfaces.IManageableEntity< in T >

    Defines the IManageableEntity<T> interface for managing a collection of entities of type T . This interface standardizes methods for adding, removing, importing, and exporting entities.

Namespaces

- namespace SmartStay
- namespace SmartStay.Core
- namespace SmartStay.Core.Models

The `SmartStay.Core.Models` namespace contains the primary data models used within the SmartStay appli-cation. These models represent core entities and structures essential for managing application data.

- namespace SmartStay.Core.Models.Interfaces

This namespace contains interfaces used within the SmartStay application.

7.60 ManageableEntity.cs

Go to the documentation of this file.
```
00001
00014
00018 namespace SmartStay.Core.Models.Interfaces
00019 {
00020 using SmartStay.Common.Models;
00021
00028 public interface IManageableEntity<in T>
00029 {
00035     bool Add(T item);
00036
00042     bool Remove(T item);
00043
00048     ImportResult Import(string data);
00049
00054     string Export();
00055
00060     void Save(string filePath);
00061
00066     void Load(string filePath);
00067 }
00068 }
```

7.61 Owner.cs File Reference

Data Structures

- class SmartStay.Core.Models.Owner

Defines the Owner class, which encapsulates the details of an accommodation owner, including personal information such as first name, last name, email address, phone number, and a list of owned accommodations. This class validates the provided data upon creation or when modifying specific properties, ensuring that all data is consistent and correct.

Namespaces

- namespace SmartStay
- namespace SmartStay.Core
- namespace SmartStay.Core.Models

The `SmartStay.Core.Models` namespace contains the primary data models used within the SmartStay appli-cation. These models represent core entities and structures essential for managing application data.

7.62 Owner.cs

Go to the documentation of this file.
```
00001
00011 using System.Text.Encodings.Web;
00012 using System.Text.Json;
00013 using System.Text.Json.Serialization;
00014 using ProtoBuf;
00015 using SmartStay.Validation.Validators;
00016
```

```
00021 namespace SmartStay.Core.Models
00022 {
00029 [ProtoContract]
00030 public class Owner
00031 {
00035     static int _lastOwnerId = 0;
00036
00049     static readonly JsonSerializerOptions _jsonOptions =
00050         new() { WriteIndented = true, Encoder = JavaScriptEncoder.UnsafeRelaxedJsonEscaping,
00051                 Converters = { new JsonStringEnumConverter() } };
00052
00056     [ProtoMember(1)]
00057     readonly int _id; // ID of the owner
00058
00062     [ProtoMember(2)]
00063     string _firstName; // First name of the owner
00064
00068     [ProtoMember(3)]
00069     string _lastName; // Last name of the owner
00070
00074     [ProtoMember(4)]
00075     string _email; // Email address of the owner
00076
00081     [ProtoMember(5)]
00082     string _phoneNumber = string.Empty; // Phone number of the owner
00083
00088     [ProtoMember(6)]
00089     string _address = string.Empty; // Address of the owner
00090
00095     [ProtoMember(7)]
00096     readonly List<Accommodation> _accommodationsOwned = []; // List of accommodations owned by the
    owner
00097
00104 #pragma warning disable CS8618
00105     public Owner()
00106 #pragma warning restore CS8618
00107     {
00108         // This constructor is intentionally empty and only needed for Protobuf-net deserialization.
00109     }
00110
00123     public Owner(string firstName, string lastName, string email)
00124     {
00125         NameValidator.ValidateName(firstName);
00126         NameValidator.ValidateName(lastName);
00127         EmailValidator.ValidateEmail(email);
00128
00129         _id = GenerateOwnerId();
00130         _firstName = firstName;
00131         _lastName = lastName;
00132         _email = email;
00133     }
00134
00151     public Owner(string firstName, string lastName, string email, string phoneNumber, string address)
00152         : this(firstName, lastName, email)
00153     {
00154         PhoneNumberValidator.ValidatePhoneNumber(phoneNumber);
00155         AddressValidator.ValidateAddress(address);
00156
00157         _phoneNumber = phoneNumber;
00158         _address = address;
00159     }
00160
00178     [JsonConstructor]
00179     public Owner(int id, string firstName, string lastName, string email, string phoneNumber, string
    address,
00180                  List<Accommodation> accommodationsOwned)
00181     {
00182         _id = id;
00183         UpdateLastOwnerId(id);
00184         _firstName = firstName;
00185         _lastName = lastName;
00186         _email = email;
00187         _phoneNumber = phoneNumber;
00188         _address = address;
00189         _accommodationsOwned = accommodationsOwned;
00190     }
00191
00195     public static int LastAssignedId
00196     {
00197         get => _lastOwnerId;
00198         set {
00199             if (_lastOwnerId < value)
00200                 _lastOwnerId = value;
00201         }
00202     }
00203
00207     public int Id => _id;
```

```
00208
00213     public string FirstName
00214     {
00215         get => _firstName;
00216         set => _firstName = NameValidator.ValidateName(value);
00217     }
00218
00223     public string LastName
00224     {
00225         get => _lastName;
00226         set => _lastName = NameValidator.ValidateName(value);
00227     }
00228
00233     public string Email
00234     {
00235         get => _email;
00236         set => _email = EmailValidator.ValidateEmail(value);
00237     }
00238
00243     public string PhoneNumber
00244     {
00245         get => _phoneNumber;
00246         set => _phoneNumber = PhoneNumberValidator.ValidatePhoneNumber(value);
00247     }
00248
00253     public string Address
00254     {
00255         get => _address;
00256         set => _address = AddressValidator.ValidateAddress(value);
00257     }
00258
00270     public List<Accommodation> AccommodationsOwned => GetAccommodationsCopy();
00271
00283     public bool AddAccommodation(Accommodation accommodation)
00284     {
00285         if (accommodation == null)
00286             return false;
00287
00288         _accommodationsOwned.Add(accommodation);
00289         return true;
00290     }
00291
00305     public bool RemoveAccommodation(Accommodation accommodation)
00306     {
00307         if (accommodation == null)
00308             return false;
00309
00310         _accommodationsOwned.Remove(accommodation);
00311         return true;
00312     }
00313
00319     private static int GenerateOwnerId()
00320     {
00321         // Check if the current value exceeds the max limit of int (2,147,483,647)
00322         if (_lastOwnerId >= int.MaxValue)
00323         {
00324             throw new InvalidOperationException("Owner ID limit exceeded.");
00325         }
00326
00327         return Interlocked.Increment(ref _lastOwnerId);
00328     }
00329
00334     private static void UpdateLastOwnerId(int id)
00335     {
00336         if (id > _lastOwnerId)
00337         {
00338             _lastOwnerId = id; // Update the last assigned owner ID if the new ID is larger
00339         }
00340     }
00341
00346     private List<Accommodation> GetAccommodationsCopy()
00347     {
00348         // Deep copy each room
00349         return _accommodationsOwned.Select(accommodation => accommodation.Clone()).ToList();
00350     }
00351
00356     public override string ToString()
00357     {
00358         return JsonSerializer.Serialize(this, _jsonOptions);
00359     }
00360 }
00361 }
```

7.63   Payment.cs File Reference

Data Structures

- class SmartStay.Core.Models.Payment

  Represents a payment made in the SmartStay system, with details such as amount, date, method, and status.

Namespaces

- namespace SmartStay
- namespace SmartStay.Core
- namespace SmartStay.Core.Models

  The `SmartStay.Core.Models` namespace contains the primary data models used within the SmartStay application. These models represent core entities and structures essential for managing application data.

7.64   Payment.cs

Go to the documentation of this file.
```
00001
00011 using System.ComponentModel.DataAnnotations;
00012 using System.Text.Encodings.Web;
00013 using System.Text.Json;
00014 using System.Text.Json.Serialization;
00015 using ProtoBuf;
00016 using SmartStay.Common.Enums;
00017 using SmartStay.Validation;
00018 using SmartStay.Validation.Validators;
00019
00024 namespace SmartStay.Core.Models
00025 {
00029 [ProtoContract]
00030 public class Payment
00031 {
00035     static int _lastPaymentId = 0;
00036
00049     static readonly JsonSerializerOptions _jsonOptions =
00050         new() { WriteIndented = true, Encoder = JavaScriptEncoder.UnsafeRelaxedJsonEscaping,
00051               Converters = { new JsonStringEnumConverter() } };
00052
00056     [ProtoMember(1)]
00057     readonly int _id; // ID of the payment
00058
00062     [ProtoMember(2)]
00063     int _reservationId; // ID of the reservation being paid
00064
00068     [ProtoMember(3)]
00069     decimal _amount; // Amount of the payment
00070
00074     [ProtoMember(4)]
00075     DateTime _date; // Date the payment was made
00076
00080     [ProtoMember(5)]
00081     PaymentMethod _method; // Payment Method used
00082
00086     [ProtoMember(6)]
00087     PaymentStatus _status; // Status of the payment
00088
00095 #pragma warning disable CS8618
00096     public Payment()
00097 #pragma warning restore CS8618
00098     {
00099         // This constructor is intentionally empty and only needed for Protobuf-net deserialization.
00100     }
00101
00112     public Payment(int reservationId, decimal amount, DateTime paymentDate, PaymentMethod
    paymentMethod,
00113                 PaymentStatus paymentStatus)
00114     {
00115         ReservationValidator.ValidateReservationId(reservationId);
00116         PaymentValidator.ValidatePayment(amount);
00117         PaymentValidator.ValidatePaymentMethod(paymentMethod);
```

```
00118            PaymentValidator.ValidatePaymentStatus(paymentStatus);
00119
00120            _id = GeneratePaymentId();
00121            _reservationId = reservationId;
00122            _amount = amount;
00123            _date = paymentDate;
00124            _method = paymentMethod;
00125            _status = paymentStatus;
00126        }
00127
00144        [JsonConstructor]
00145        public Payment(int id, int reservationId, decimal amount, DateTime date, PaymentMethod method,
     PaymentStatus status)
00146        {
00147            _id = id;
00148            UpdateLastPaymentId(id);
00149            _reservationId = reservationId;
00150            _amount = amount;
00151            _date = date;
00152            _method = method;
00153            _status = status;
00154        }
00155
00159        public static int LastAssignedId
00160        {
00161            get => _lastPaymentId;
00162            set {
00163                if (_lastPaymentId < value)
00164                    _lastPaymentId = value;
00165            }
00166        }
00167
00171        public int Id => _id;
00172
00176        public int ReservationId => _reservationId;
00177
00181        public decimal Amount => _amount;
00182
00186        public DateTime Date => _date;
00187
00191        public PaymentMethod Method => _method;
00192
00200        public PaymentStatus Status
00201        {
00202
00203            get => _status;
00204            set => _status = PaymentValidator.ValidatePaymentStatus(value);
00205        }
00206
00212        private static int GeneratePaymentId()
00213        {
00214            // Check if the current value exceeds the max limit of int (2,147,483,647)
00215            if (_lastPaymentId >= int.MaxValue)
00216            {
00217                throw new InvalidOperationException("Payment ID limit exceeded.");
00218            }
00219
00220            return Interlocked.Increment(ref _lastPaymentId);
00221        }
00222
00227        private static void UpdateLastPaymentId(int id)
00228        {
00229            if (id > _lastPaymentId)
00230            {
00231                _lastPaymentId = id; // Update the last assigned owner ID if the new ID is larger
00232            }
00233        }
00234
00239        public Payment Clone()
00240        {
00241            // Create a new instance of Payment and deep copy the fields
00242            return new Payment(_id,              // Immutable, directly copy
00243                               _reservationId, // Value type, directly copy
00244                               _amount,         // Value type, directly copy
00245                               _date,           // Value type (DateTime), directly copy
00246                               _method,         // Enum type, directly copy
00247                               _status          // Enum type, directly copy
00248            );
00249        }
00250
00255        public override string ToString()
00256        {
00257            return JsonSerializer.Serialize(this, _jsonOptions);
00258        }
00259 }
00260 }
```

7.65   Reservation.cs File Reference

Data Structures

- class SmartStay.Core.Models.Reservation

  Defines the Reservation class, which encapsulates reservation details such as client ID, accommodation type, dates, and payment information. This class ensures data consistency by validating input parameters upon creation or when modifying specific properties.

Namespaces

- namespace SmartStay
- namespace SmartStay.Core
- namespace SmartStay.Core.Models

  The `SmartStay.Core.Models` *namespace contains the primary data models used within the* SmartStay *application. These models represent core entities and structures essential for managing application data.*

7.66   Reservation.cs

Go to the documentation of this file.
```
00001
00011 using System.Text.Encodings.Web;
00012 using System.Text.Json;
00013 using System.Text.Json.Serialization;
00014 using ProtoBuf;
00015 using SmartStay.Common.Enums;
00016 using SmartStay.Validation;
00017 using SmartStay.Validation.Validators;
00018
00023 namespace SmartStay.Core.Models
00024 {
00030 [ProtoContract]
00031 public class Reservation
00032 {
00036     static int _lastReservationId = 0;
00037
00050     static readonly JsonSerializerOptions _jsonOptions =
00051         new() { WriteIndented = true, Encoder = JavaScriptEncoder.UnsafeRelaxedJsonEscaping,
00052                 Converters = { new JsonStringEnumConverter() } };
00053
00057     [ProtoMember(1)]
00058     readonly int _reservationId; // ID of the reservation
00059
00063     [ProtoMember(2)]
00064     readonly int _clientId; // ID of the client making the reservation
00065
00069     [ProtoMember(3)]
00070     readonly int _accommodationId; // ID of the accommodation
00071
00075     [ProtoMember(4)]
00076     readonly int _roomId; // ID of the room
00077
00081     [ProtoMember(5)]
00082     AccommodationType _accommodationType; // Type of accommodation (e.g., Room, Suite, etc.)
00083
00087     [ProtoMember(6)]
00088     DateTime _checkInDate; // Check-in date for the reservation
00089
00093     [ProtoMember(7)]
00094     DateTime _checkOutDate; // Check-out date for the reservation
00095
00100     [ProtoMember(8)]
00101     ReservationStatus _status = ReservationStatus.Pending; // Current reservation status
00102
00106     [ProtoMember(9)]
00107     decimal _totalCost; // Total cost of the reservation
00108
00112     [ProtoMember(10)]
00113     decimal _amountPaid = 0; // Amount paid towards the reservation
00114
```

```
00118     [ProtoMember(11)]
00119     readonly List<Payment> _payments = []; // List of payments made for the reservation
00120
00127 #pragma warning disable CS8618
00128     public Reservation()
00129 #pragma warning restore CS8618
00130     {
00131         // This constructor is intentionally empty and only needed for Protobuf-net deserialization.
00132     }
00133
00151     public Reservation(int clientId, int accommodationId, int roomId, AccommodationType
    accommodationType,
00152                        DateTime checkInDate, DateTime checkOutDate, decimal totalCost)
00153     {
00154         ClientValidator.ValidateClientId(clientId);
00155         AccommodationValidator.ValidateAccommodationId(accommodationId);
00156         RoomValidator.ValidateRoomId(roomId);
00157         PaymentValidator.ValidateTotalCost(totalCost);
00158         if (!DateValidator.IsValidDateRange(checkInDate, checkOutDate))
00159             throw new ValidationException(ValidationErrorCode.InvalidDateRange);
00160
00161         _reservationId = GenerateReservationId();
00162         _clientId = clientId;
00163         _accommodationId = accommodationId;
00164         _roomId = roomId;
00165         _accommodationType = accommodationType;
00166         _checkInDate = checkInDate;
00167         _checkOutDate = checkOutDate;
00168         _totalCost = totalCost;
00169     }
00170
00191     [JsonConstructor]
00192     public Reservation(int id, int clientId, int accommodationId, int roomId, AccommodationType
    accommodationType,
00193                        DateTime checkInDate, DateTime checkOutDate, ReservationStatus status, decimal
    totalCost,
00194                        decimal amountPaid, List<Payment> payments)
00195     {
00196         _reservationId = id;
00197         UpdateLastReservationId(id);
00198         _clientId = clientId;
00199         _accommodationId = accommodationId;
00200         _roomId = roomId;
00201         _accommodationType = accommodationType;
00202         _checkInDate = checkInDate;
00203         _checkOutDate = checkOutDate;
00204         _status = status;
00205         _totalCost = totalCost;
00206         _amountPaid = amountPaid;
00207         _payments = payments;
00208     }
00209
00213     public static int LastAssignedId
00214     {
00215         get => _lastReservationId;
00216         set {
00217             if (_lastReservationId < value)
00218                 _lastReservationId = value;
00219         }
00220     }
00221
00225     public int Id => _reservationId;
00226
00230     public int ClientId => _clientId;
00231
00235     public int AccommodationId => _accommodationId;
00236
00240     public int RoomId => _roomId;
00241
00245     public AccommodationType AccommodationType
00246     {
00247         get => _accommodationType;
00248         set => _accommodationType = AccommodationValidator.ValidateAccommodationType(value);
00249     }
00250
00254     public DateTime CheckInDate
00255     {
00256         get => _checkInDate;
00257         set => _checkInDate = DateValidator.ValidateCheckInDate(value);
00258     }
00259
00263     public DateTime CheckOutDate
00264     {
00265         get => _checkOutDate;
00266         set => _checkOutDate = DateValidator.ValidateCheckOutDate(value, _checkInDate);
00267     }
00268
```

```
00272      public ReservationStatus Status
00273      {
00274          get => _status;
00275          set => _status = ReservationValidator.ValidateReservationStatus(value);
00276      }
00277
00281      public decimal TotalCost
00282      {
00283          get => _totalCost;
00284          set => _totalCost = PaymentValidator.ValidateTotalCost(value);
00285      }
00286
00290      public decimal AmountPaid
00291      {
00292          get => _amountPaid;
00293          set => _amountPaid = PaymentValidator.ValidatePayment(value);
00294      }
00295
00307      public List<Payment> Payments => GetPaymentsCopy();
00308
00320      public bool CheckIn()
00321      {
00322          if (_status != ReservationStatus.Pending)
00323          {
00324              return false;
00325          }
00326          _status = ReservationStatus.CheckedIn;
00327          return true;
00328      }
00329
00341      public bool CheckOut()
00342      {
00343          if (_status != ReservationStatus.CheckedIn)
00344          {
00345              return false;
00346          }
00347          _status = ReservationStatus.CheckedOut;
00348          return true;
00349      }
00350
00363      public PaymentResult MakePayment(decimal paymentAmount, PaymentMethod paymentMethod)
00364      {
00365          if (paymentAmount <= 0)
00366              return PaymentResult.InvalidAmount;
00367          if (IsFullyPaid())
00368              return PaymentResult.AlreadyFullyPaid;
00369          if (paymentAmount > _totalCost - _amountPaid)
00370              return PaymentResult.AmountExceedsTotal;
00371          if (!PaymentValidator.IsValidPaymentMethod(paymentMethod))
00372              return PaymentResult.InvalidPaymentMethod;
00373
00374          // Create a new Payment instance and add it to the list
00375          var payment = new Payment(_reservationId, paymentAmount, DateTime.Now, paymentMethod,
       PaymentStatus.Completed);
00376          _payments.Add(payment);
00377
00378          // Update the amount paid
00379          _amountPaid += paymentAmount;
00380
00381          return PaymentResult.Success;
00382      }
00383
00388      public bool IsFullyPaid()
00389      {
00390          return _amountPaid >= _totalCost;
00391      }
00392
00398      private static int GenerateReservationId()
00399      {
00400          if (_lastReservationId >= int.MaxValue)
00401          {
00402              throw new InvalidOperationException("Reservation ID limit exceeded.");
00403          }
00404          return Interlocked.Increment(ref _lastReservationId);
00405      }
00406
00411      private static void UpdateLastReservationId(int id)
00412      {
00413          if (id > _lastReservationId)
00414          {
00415              _lastReservationId = id; // Update the last assigned owner ID if the new ID is larger
00416          }
00417      }
00418
00423      private List<Payment> GetPaymentsCopy()
00424      {
00425          // Deep copy each payment
```

```
00426            return _payments.Select(payment => payment.Clone()).ToList();
00427       }
00428
00433       public override string ToString()
00434       {
00435            return JsonSerializer.Serialize(this, _jsonOptions);
00436       }
00437 }
00438 }
```

7.67 Room.cs File Reference

Data Structures

- class SmartStay.Core.Models.Room

  Defines the Room class, which encapsulates the details of a room within an accommodation, such as its type, price per night, and reservation details. This class provides methods for updating availability and calculating the total cost for a stay.

Namespaces

- namespace SmartStay
- namespace SmartStay.Core
- namespace SmartStay.Core.Models

  The SmartStay.Core.Models namespace contains the primary data models used within the SmartStay application. These models represent core entities and structures essential for managing application data.

7.68 Room.cs

Go to the documentation of this file.
```
00001
00011 using System.Text.Encodings.Web;
00012 using System.Text.Json;
00013 using System.Text.Json.Serialization;
00014 using ProtoBuf;
00015 using SmartStay.Common.Enums;
00016 using SmartStay.Core.Utilities;
00017 using SmartStay.Validation;
00018 using SmartStay.Validation.Validators;
00019
00024 namespace SmartStay.Core.Models
00025 {
00031 [ProtoContract]
00032 public class Room
00033 {
00037       static int _lastRoomId = 0; // Last assigned room ID
00038
00051       static readonly JsonSerializerOptions _jsonOptions =
00052           new() { WriteIndented = true, Encoder = JavaScriptEncoder.UnsafeRelaxedJsonEscaping,
00053                   Converters = { new JsonStringEnumConverter() } };
00054
00058       [ProtoMember(1)]
00059       readonly int _id; // ID of the room
00060
00064       [ProtoMember(2)]
00065       RoomType _type; // Type of room (Single, Double, Suite, etc.)
00066
00070       [ProtoMember(3)]
00071       decimal _pricePerNight; // Price per night for the room
00072
00077       [ProtoMember(4)]
00078       readonly SortedSet<DateRange> _reservationDates = []; // Sorted set for efficient availability
      check
00079
00086       public Room()
00087       {
```

```
00088          // This constructor is intentionally empty and only needed for Protobuf-net deserialization.
00089      }
00090
00098      public Room(RoomType type, decimal pricePerNight)
00099      {
00100          RoomValidator.ValidateRoomType(type);
00101          PaymentValidator.ValidatePrice(pricePerNight);
00102
00103          _id = GenerateRoomId();
00104          _type = type;
00105          _pricePerNight = pricePerNight;
00106      }
00107
00122      [JsonConstructor]
00123      public Room(int id, RoomType type, decimal pricePerNight, SortedSet<DateRange> reservationDates)
00124      {
00125          _id = id;
00126          UpdateLastRoomId(id);
00127          _type = type;
00128          _pricePerNight = pricePerNight;
00129          _reservationDates = reservationDates;
00130      }
00131
00135      public static int LastAssignedId
00136      {
00137          get => _lastRoomId;
00138          set {
00139              if (_lastRoomId < value)
00140                  _lastRoomId = value;
00141          }
00142      }
00143
00147      public int Id => _id;
00148
00152      public RoomType Type
00153      {
00154          get => _type;
00155          set => _type = RoomValidator.ValidateRoomType(value);
00156      }
00157
00161      public decimal PricePerNight
00162      {
00163          get => _pricePerNight;
00164          set => _pricePerNight = PaymentValidator.ValidatePrice(value);
00165      }
00166
00178      public SortedSet<DateRange> ReservationDates => GetReservationDatesCopy();
00179
00200      public bool IsAvailable(DateTime startDate, DateTime endDate, DateRange? existingReservationRange
    = null)
00201      {
00202          if (endDate <= startDate)
00203              throw new ArgumentException("End date must be after the start date.");
00204
00205          var newReservation = new DateRange(startDate, endDate);
00206
00207          // Get potential conflicting reservations within the requested range
00208          var potentialConflicts = _reservationDates.GetViewBetween(
00209              new DateRange(DateTime.MinValue, startDate), // All reservations that end before the start
00210              new DateRange(DateTime.MaxValue, endDate)    // All reservations that start after the end
00211          );
00212
00213          // Check if there are any overlapping reservations
00214          foreach (var existingReservation in potentialConflicts)
00215          {
00216              // Skip the existing reservation if it's the one we're trying to modify
00217              if (existingReservation.Equals(existingReservationRange))
00218              {
00219                  continue; // Skip this reservation as it's the one we're modifying
00220              }
00221
00222              // An overlap occurs if the start date is before the end date, and the end date is after
    the start date
00223              if ((newReservation.Start < existingReservation.End) && (newReservation.End >
    existingReservation.Start))
00224              {
00225                  return false; // There's an overlap, so the accommodation is not available
00226              }
00227          }
00228
00229          return true; // No overlap found, accommodation is available
00230      }
00231
00245      public bool AddReservation(DateTime startDate, DateTime endDate)
00246      {
00247          var newReservation = new DateRange(startDate, endDate);
00248
```

```
00249         // Attempt to add the new reservation
00250         bool addedSuccessfully = _reservationDates.Add(newReservation);
00251
00252         // Return the result of the Add operation (true if added, false if it already exists due to
       overlap)
00253         return addedSuccessfully;
00254     }
00255
00269     public bool RemoveReservation(DateTime startDate, DateTime endDate)
00270     {
00271         // Create the date range object to remove
00272         var reservationToRemove = new DateRange(startDate, endDate);
00273
00274         // Remove the reservation from the SortedSet
00275         bool removed = _reservationDates.Remove(reservationToRemove);
00276
00277         // Return whether the reservation was successfully removed
00278         return removed;
00279     }
00280
00288     public decimal CalculateTotalCost(DateTime startDate, DateTime endDate)
00289     {
00290         if (endDate <= startDate)
00291         {
00292             throw new ArgumentException("End date must be after the start date.");
00293         }
00294
00295         int nights = (endDate - startDate).Days;
00296         return nights * _pricePerNight;
00297     }
00298
00303     private static int GenerateRoomId()
00304     {
00305         if (_lastRoomId >= int.MaxValue)
00306             throw new InvalidOperationException("Room ID limit exceeded.");
00307
00308         return Interlocked.Increment(ref _lastRoomId);
00309     }
00310
00315     private static void UpdateLastRoomId(int id)
00316     {
00317         if (id > _lastRoomId)
00318         {
00319             _lastRoomId = id; // Update the last assigned owner ID if the new ID is larger
00320         }
00321     }
00322
00327     private SortedSet<DateRange> GetReservationDatesCopy()
00328     {
00329         // Deep copy each DateRange in the collection
00330         return new SortedSet<DateRange>(_reservationDates.Select(dateRange => dateRange.Clone()));
00331     }
00332
00337     public Room Clone()
00338     {
00339         // Create a new instance of Room and deep copy the fields
00340         return new Room(
00341             _id,                                                          // Immutable
00342             _type,                                                        // Immutable
00343             _pricePerNight,                                               // Value type,
       directly copy
00344             new SortedSet<DateRange>(_reservationDates.Select(dr => dr.Clone())) // Deep copy of
       DateRange objects
00345         );
00346     }
00347
00352     public override string ToString()
00353     {
00354         return JsonSerializer.Serialize(this, _jsonOptions);
00355     }
00356 }
00357 }
```

7.69 SmartStay.Core.AssemblyInfo.cs File Reference

7.70 SmartStay.Core.AssemblyInfo.cs

Go to the documentation of this file.

```
00001 //------------------------------------------------------------------------------
00002 // <auto-generated>
00003 //     This code was generated by a tool.
00004 //     Runtime Version:4.0.30319.42000
00005 //
00006 //     Changes to this file may cause incorrect behavior and will be lost if
00007 //     the code is regenerated.
00008 // </auto-generated>
00009 //------------------------------------------------------------------------------
00010
00011 using System;
00012 using System.Reflection;
00013
00014 [assembly: System.Reflection.AssemblyCompanyAttribute("SmartStay.Core")]
00015 [assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]
00016 [assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]
00017 [assembly:
      System.Reflection.AssemblyInformationalVersionAttribute("1.0.0+dae448e0a56b3719559d0544250df1ecd986a2d4")]
00018 [assembly: System.Reflection.AssemblyProductAttribute("SmartStay.Core")]
00019 [assembly: System.Reflection.AssemblyTitleAttribute("SmartStay.Core")]
00020 [assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]
00021
00022 // Generated by the MSBuild WriteCodeFragment class.
00023
```

7.71 SmartStay.Core.GlobalUsings.g.cs File Reference

7.72 SmartStay.Core.GlobalUsings.g.cs

Go to the documentation of this file.
```
00001 // <auto-generated/>
00002 global using global::System;
00003 global using global::System.Collections.Generic;
00004 global using global::System.IO;
00005 global using global::System.Linq;
00006 global using global::System.Net.Http;
00007 global using global::System.Threading;
00008 global using global::System.Threading.Tasks;
```

7.73 Accommodations.cs File Reference

Data Structures

- class SmartStay.Core.Repositories.Accommodations

  Represents a collection of Accommodation objects, managed in a dictionary for fast lookup by accommodation ID.

Namespaces

- namespace SmartStay
- namespace SmartStay.Core
- namespace SmartStay.Core.Repositories

  The `SmartStay.Repositories` *namespace provides data access layers for retrieving and storing application data. It contains repositories that manage database interactions for various entities within the SmartStay application.*

7.74 Accommodations.cs

Go to the documentation of this file.
```
00001
00013 #nullable enable
00014 using System.Runtime.Serialization;
00015 using ProtoBuf;
00016 using SmartStay.Common.Models;
00017 using SmartStay.Core.Models;
00018 using SmartStay.Core.Models.Interfaces;
00019 using SmartStay.Core.Utilities;
00020
00025 namespace SmartStay.Core.Repositories
00026 {
00031 [ProtoContract]
00032 public class Accommodations : IManageableEntity<Accommodation>
00033 {
00037     readonly Dictionary<int, Accommodation> _accommodationDictionary = new();
00038
00045     [ProtoMember(1)] // Serialize the list of accommodations
00046     List<Accommodation> _accommodationList = new();
00047
00057     public bool Add(Accommodation accommodation)
00058     {
00059         if (accommodation == null)
00060         {
00061             throw new ArgumentNullException(nameof(accommodation), "Accommodation cannot be null");
00062         }
00063
00064         if (_accommodationDictionary.ContainsKey(accommodation.Id))
00065         {
00066             return false; // Accommodation already exists
00067         }
00068
00069         _accommodationDictionary[accommodation.Id] = accommodation;
00070         return true; // Accommodation added successfully
00071     }
00072
00082     public bool Remove(Accommodation accommodation)
00083     {
00084         if (accommodation == null)
00085         {
00086             throw new ArgumentNullException(nameof(accommodation), "Accommodation cannot be null");
00087         }
00088
00089         return _accommodationDictionary.Remove(accommodation.Id); // Remove using accommodation ID
00090     }
00091
00102     public ImportResult Import(string data)
00103     {
00104         if (string.IsNullOrEmpty(data))
00105         {
00106             throw new ArgumentException("Data cannot be null or empty", nameof(data));
00107         }
00108
00109         // Deserialize the data into a List<Accommodation> instead of a single Accommodation
00110         var accommodations =
00111             JsonHelper.DeserializeFromJson<Accommodation>(data) ??
00112             throw new ArgumentException("Deserialized accommodation data cannot be null",
    nameof(data));
00113
00114         int replacedCount = 0;
00115         int importedCount = 0;
00116
00117         foreach (var accommodation in accommodations)
00118         {
00119             if (_accommodationDictionary.ContainsKey(accommodation.Id))
00120             {
00121                 replacedCount++;
00122             }
00123             else
00124             {
00125                 importedCount++;
00126             }
00127             _accommodationDictionary[accommodation.Id] = accommodation; // Direct insertion for
    efficiency
00128         }
00129
00130         return new ImportResult { ImportedCount = importedCount, ReplacedCount = replacedCount };
00131     }
00132
00137     public string Export()
00138     {
00139         return JsonHelper.SerializeToJson(_accommodationDictionary.Values);
00140     }
```

```
00141
00147     [ProtoBeforeSerialization]
00148     private void PrepareForSerialization()
00149     {
00150         // Clear the temporary list to ensure no leftover data
00151         _accommodationList.Clear();
00152
00153         // Add all accommodations from the dictionary to the temporary list
00154         foreach (var accommodation in _accommodationDictionary.Values)
00155         {
00156             _accommodationList.Add(accommodation);
00157         }
00158     }
00159
00164     [ProtoAfterDeserialization]
00165     [System.Diagnostics.CodeAnalysis.SuppressMessage("CodeQuality", "IDE0051:Remove unused private
      members",
00166                                                      Justification =
00167                                                          "IDE Error, this is called automatically by
      protobuf-net.")]
00168     private void AfterDeserialization()
00169     {
00170         // Clear the dictionary before rebuilding
00171         _accommodationDictionary.Clear();
00172
00173         // Rebuild the dictionary using the data from the list
00174         foreach (var accommodation in _accommodationList)
00175         {
00176             _accommodationDictionary[accommodation.Id] = accommodation;
00177         }
00178
00179         // Clear the temporary list once the dictionary is rebuilt
00180         _accommodationList.Clear();
00181
00182         // Set _lastAccommodationId to the highest ID in the deserialized data
00183         if (_accommodationDictionary.Count > 0)
00184         {
00185             // Find the highest ID from the loaded accommodations
00186             Accommodation.LastAssignedId = _accommodationDictionary.Values.Max(a => a.Id);
00187         }
00188         else
00189         {
00190             // If no accommodations, reset to 0
00191             Accommodation.LastAssignedId = 0;
00192         }
00193     }
00194
00204     public void Save(string filePath)
00205     {
00206         try
00207         {
00208             // Prepare for serialization by copying the dictionary contents to the temporary list
00209             PrepareForSerialization();
00210
00211             // Open the file stream for saving the data to the specified file
00212             using (var fileStream = File.Create(filePath))
00213             {
00214                 // Serialize the accommodations object and write it to the file stream
00215                 Serializer.Serialize(fileStream, this);
00216             }
00217         }
00218         catch (IOException ioEx)
00219         {
00220             throw new IOException("An error occurred while saving the accommodations data.", ioEx);
00221         }
00222         catch (SerializationException serEx)
00223         {
00224             throw new SerializationException(
00225                 "An error occurred during serialization while saving the accommodations data.",
      serEx);
00226         }
00227     }
00228
00237     public void Load(string filePath)
00238     {
00239         try
00240         {
00241             // Open the file stream for reading
00242             using (var fileStream = File.OpenRead(filePath))
00243             {
00244                 // Deserialize the accommodations object from the file
00245                 var accommodations = Serializer.Deserialize<Accommodations>(fileStream);
00246
00247                 // Copy the data from the deserialized object to the current instance
00248                 _accommodationDictionary.Clear();
00249                 foreach (var accommodation in accommodations._accommodationDictionary)
00250                 {
```

```
00251                    _accommodationDictionary[accommodation.Key] = accommodation.Value;
00252                }
00253            }
00254        }
00255        catch (IOException ioEx)
00256        {
00257            throw new IOException("An error occurred while loading the accommodations data.", ioEx);
00258        }
00259        catch (SerializationException serEx)
00260        {
00261            throw new SerializationException(
00262                "An error occurred during deserialization while loading the accommodations data.",
      serEx);
00263        }
00264    }
00265
00273    public Accommodation? FindAccommodationById(int accommodationId)
00274    {
00275        _accommodationDictionary.TryGetValue(accommodationId, out Accommodation? accommodation);
00276        return accommodation;
00277    }
00278
00285    public int CountAccommodations()
00286    {
00287        return _accommodationDictionary.Count;
00288    }
00289 }
00290 }
```

7.75 Clients.cs File Reference

Data Structures

- class SmartStay.Core.Repositories.Clients

    Represents a collection of Client objects, managed in a dictionary for fast lookup by client ID. Implements the IManageableEntity<Client> interface for standardized management.

Namespaces

- namespace SmartStay
- namespace SmartStay.Core
- namespace SmartStay.Core.Repositories

    The `SmartStay.Repositories` *namespace provides data access layers for retrieving and storing application data. It contains repositories that manage database interactions for various entities within the SmartStay application.*

7.76 Clients.cs

Go to the documentation of this file.
```
00001
00011 #nullable enable
00012 using System;
00013 using System.Collections.Generic;
00014 using System.Linq;
00015 using System.Runtime.Serialization;
00016 using ProtoBuf;
00017 using SmartStay.Common.Models;
00018 using SmartStay.Core.Models;
00019 using SmartStay.Core.Models.Interfaces;
00020 using SmartStay.Core.Utilities;
00021
00026 namespace SmartStay.Core.Repositories
00027 {
00032 [ProtoContract]
00033 public class Clients : IManageableEntity<Client>
00034 {
00038    readonly Dictionary<int, Client> _clientDictionary = new();
00039
```

```
00046       [ProtoMember(1)] // Serialize the list of clients
00047       List<Client> _clientList = new();
00048
00060     public bool Add(Client client)
00061     {
00062         if (client == null)
00063         {
00064             throw new ArgumentNullException(nameof(client), "Client cannot be null");
00065         }
00066
00067         if (_clientDictionary.ContainsKey(client.Id))
00068         {
00069             return false; // Client already exists
00070         }
00071
00072         _clientDictionary[client.Id] = client;
00073         return true; // Client added successfully
00074     }
00075
00085     public bool Remove(Client client)
00086     {
00087         if (client == null)
00088         {
00089             throw new ArgumentNullException(nameof(client), "Client cannot be null");
00090         }
00091
00092         return _clientDictionary.Remove(client.Id); // Remove by client ID
00093     }
00094
00104     public ImportResult Import(string data)
00105     {
00106         if (string.IsNullOrEmpty(data))
00107         {
00108             throw new ArgumentException("Data cannot be null or empty", nameof(data));
00109         }
00110
00111         // Deserialize the data into a List<Client> instead of a single Client
00112         var clients = JsonHelper.DeserializeFromJson<Client>(data) ??
00113                       throw new ArgumentException("Deserialized client data cannot be null",
      nameof(data));
00114
00115         int replacedCount = 0;
00116         int importedCount = 0;
00117
00118         foreach (var client in clients)
00119         {
00120             if (_clientDictionary.ContainsKey(client.Id))
00121             {
00122                 replacedCount++;
00123             }
00124             else
00125             {
00126                 importedCount++;
00127             }
00128             _clientDictionary[client.Id] = client; // Direct insertion for efficiency
00129         }
00130
00131         return new ImportResult { ImportedCount = importedCount, ReplacedCount = replacedCount };
00132     }
00133
00138     public string Export()
00139     {
00140         return JsonHelper.SerializeToJson(_clientDictionary.Values ?? Enumerable.Empty<Client>());
00141     }
00142
00148       [ProtoBeforeSerialization]
00149     private void PrepareForSerialization()
00150     {
00151         // Clear the temporary list to ensure no leftover data
00152         _clientList.Clear();
00153
00154         // Add all clients from the dictionary to the temporary list
00155         foreach (var client in _clientDictionary.Values)
00156         {
00157             _clientList.Add(client);
00158         }
00159     }
00160
00165       [ProtoAfterDeserialization]
00166       [System.Diagnostics.CodeAnalysis.SuppressMessage("CodeQuality", "IDE0051:Remove unused private
      members",
00167                                                       Justification =
00168                                                       "IDE Error, this is called automatically by
      protobuf-net.")]
00169     private void AfterDeserialization()
00170     {
00171         // Clear the dictionary before rebuilding
```

```
00172            _clientDictionary.Clear();
00173
00174            // Rebuild the dictionary using the data from the list
00175            foreach (var client in _clientList)
00176            {
00177                _clientDictionary[client.Id] = client;
00178            }
00179
00180            // Clear the temporary list once the dictionary is rebuilt
00181            _clientList.Clear();
00182        }
00183
00193    public void Save(string filePath)
00194    {
00195        try
00196        {
00197            // Prepare for serialization by copying the dictionary contents to the temporary list
00198            PrepareForSerialization();
00199
00200            // Open the file stream for saving the data to the specified file
00201            using (var fileStream = File.Create(filePath))
00202            {
00203                // Serialize the clients object and write it to the file stream
00204                Serializer.Serialize(fileStream, this);
00205            }
00206        }
00207        catch (IOException ioEx)
00208        {
00209            throw new IOException("An error occurred while saving the clients data.", ioEx);
00210        }
00211        catch (SerializationException serEx)
00212        {
00213            throw new SerializationException("An error occurred during serialization while saving the
    clients data.",
00214                                                serEx);
00215        }
00216    }
00217
00226    public void Load(string filePath)
00227    {
00228        try
00229        {
00230            // Open the file stream for reading
00231            using (var fileStream = File.OpenRead(filePath))
00232            {
00233                // Deserialize the clients object from the file
00234                var clients = Serializer.Deserialize<Clients>(fileStream);
00235
00236                // Copy the data from the deserialized object to the current instance
00237                _clientDictionary.Clear();
00238                foreach (var client in clients._clientDictionary)
00239                {
00240                    _clientDictionary[client.Key] = client.Value;
00241                }
00242            }
00243        }
00244        catch (IOException ioEx)
00245        {
00246            throw new IOException("An error occurred while loading the clients data.", ioEx);
00247        }
00248        catch (SerializationException serEx)
00249        {
00250            throw new SerializationException("An error occurred during deserialization while loading
    the clients data.",
00251                                                serEx);
00252        }
00253    }
00254
00262    public Client? FindClientById(int id)
00263    {
00264        _clientDictionary.TryGetValue(id, out Client? client);
00265        return client;
00266    }
00267
00274    public int CountClients()
00275    {
00276        return _clientDictionary.Count;
00277    }
00278 }
00279 }
```

7.77   Owners.cs File Reference

Data Structures

- class SmartStay.Core.Repositories.Owners

  Represents a collection of Owner objects, managed in a dictionary for fast lookup by owner ID. Implements the IManageableEntity<Owner> interface for standardized management.

Namespaces

- namespace SmartStay
- namespace SmartStay.Core
- namespace SmartStay.Core.Repositories

  The `SmartStay.Repositories` *namespace provides data access layers for retrieving and storing application data. It contains repositories that manage database interactions for various entities within the SmartStay application.*

7.78   Owners.cs

Go to the documentation of this file.
```
00001
00011 #nullable enable
00012 using System.Runtime.Serialization;
00013 using ProtoBuf;
00014 using SmartStay.Common.Models;
00015 using SmartStay.Core.Models;
00016 using SmartStay.Core.Models.Interfaces;
00017 using SmartStay.Core.Utilities;
00018
00023 namespace SmartStay.Core.Repositories
00024 {
00029 [ProtoContract]
00030 public class Owners : IManageableEntity<Owner>
00031 {
00035     readonly Dictionary<int, Owner> _ownerDictionary = new();
00036
00043     [ProtoMember(1)] // Serialize the list of owners
00044     List<Owner> _ownerList = new();
00045
00057     public bool Add(Owner owner)
00058     {
00059         if (owner == null)
00060         {
00061             throw new ArgumentNullException(nameof(owner), "Owner cannot be null");
00062         }
00063
00064         if (_ownerDictionary.ContainsKey(owner.Id))
00065         {
00066             return false; // Owner already exists
00067         }
00068
00069         _ownerDictionary[owner.Id] = owner;
00070         return true; // Owner added successfully
00071     }
00072
00082     public bool Remove(Owner owner)
00083     {
00084         if (owner == null)
00085         {
00086             throw new ArgumentNullException(nameof(owner), "Owner cannot be null");
00087         }
00088
00089         return _ownerDictionary.Remove(owner.Id); // Remove by owner ID
00090     }
00091
00101     public ImportResult Import(string data)
00102     {
00103         if (string.IsNullOrEmpty(data))
00104         {
00105             throw new ArgumentException("Data cannot be null or empty", nameof(data));
00106         }
```

```
00107
00108            // Deserialize the data into a List<Owner> instead of a single Owner
00109            var owners = JsonHelper.DeserializeFromJson<Owner>(data) ??
00110                     throw new ArgumentException("Deserialized owner data cannot be null",
      nameof(data));
00111
00112            int replacedCount = 0;
00113            int importedCount = 0;
00114
00115            foreach (var owner in owners)
00116            {
00117                if (_ownerDictionary.ContainsKey(owner.Id))
00118                {
00119                    replacedCount++;
00120                }
00121                else
00122                {
00123                    importedCount++;
00124                }
00125                _ownerDictionary[owner.Id] = owner; // Direct insertion for efficiency
00126            }
00127
00128            return new ImportResult { ImportedCount = importedCount, ReplacedCount = replacedCount };
00129        }
00130
00135     public string Export()
00136     {
00137            return JsonHelper.SerializeToJson(_ownerDictionary.Values ?? Enumerable.Empty<Owner>());
00138     }
00139
00144     [ProtoBeforeSerialization]
00145     private void PrepareForSerialization()
00146     {
00147            // Clear the temporary list to ensure no leftover data
00148            _ownerList.Clear();
00149
00150            // Add all owners from the dictionary to the temporary list
00151            foreach (var owner in _ownerDictionary.Values)
00152            {
00153                _ownerList.Add(owner);
00154            }
00155     }
00156
00161     [ProtoAfterDeserialization]
00162     [System.Diagnostics.CodeAnalysis.SuppressMessage("CodeQuality", "IDE0051:Remove unused private
      members",
00163                                                    Justification =
00164                                                    "IDE Error, this is called automatically by
      protobuf-net.")]
00165     private void AfterDeserialization()
00166     {
00167            // Clear the dictionary before rebuilding
00168            _ownerDictionary.Clear();
00169
00170            // Rebuild the dictionary using the data from the list
00171            foreach (var owner in _ownerList)
00172            {
00173                _ownerDictionary[owner.Id] = owner;
00174            }
00175
00176            // Clear the temporary list once the dictionary is rebuilt
00177            _ownerList.Clear();
00178
00179            // Set _lastOwnerId to the highest ID in the deserialized data
00180            if (_ownerDictionary.Count > 0)
00181            {
00182                // Find the highest ID from the loaded owners
00183                Owner.LastAssignedId = _ownerDictionary.Values.Max(o => o.Id);
00184            }
00185            else
00186            {
00187                // If no owners, reset to 0
00188                Owner.LastAssignedId = 0;
00189            }
00190     }
00191
00201     public void Save(string filePath)
00202     {
00203            try
00204            {
00205                // Prepare for serialization by copying the dictionary contents to the temporary list
00206                PrepareForSerialization();
00207
00208                // Open the file stream for saving the data to the specified file
00209                using (var fileStream = File.Create(filePath))
00210                {
00211                    // Serialize the owners object and write it to the file stream
```

```
00212                    Serializer.Serialize(fileStream, this);
00213              }
00214          }
00215          catch (IOException ioEx)
00216          {
00217              throw new IOException("An error occurred while saving the owners data.", ioEx);
00218          }
00219          catch (SerializationException serEx)
00220          {
00221              throw new SerializationException("An error occurred during serialization while saving the
      owners data.",
00222                                               serEx);
00223          }
00224      }
00225
00234      public void Load(string filePath)
00235      {
00236          try
00237          {
00238              // Open the file stream for reading
00239              using (var fileStream = File.OpenRead(filePath))
00240              {
00241                  // Deserialize the owners object from the file
00242                  var owners = Serializer.Deserialize<Owners>(fileStream);
00243
00244                  // Copy the data from the deserialized object to the current instance
00245                  _ownerDictionary.Clear();
00246                  foreach (var owner in owners._ownerDictionary)
00247                  {
00248                      _ownerDictionary[owner.Key] = owner.Value;
00249                  }
00250              }
00251          }
00252          catch (IOException ioEx)
00253          {
00254              throw new IOException("An error occurred while loading the owners data.", ioEx);
00255          }
00256          catch (SerializationException serEx)
00257          {
00258              throw new SerializationException("An error occurred during deserialization while loading
      the owners data.",
00259                                               serEx);
00260          }
00261      }
00262
00270      public Owner? FindOwnerById(int id)
00271      {
00272        _ownerDictionary.TryGetValue(id, out Owner? owner);
00273        return owner;
00274      }
00275
00282      public int CountOwners()
00283      {
00284          return _ownerDictionary.Count;
00285      }
00286 }
00287 }
```

7.79 Reservations.cs File Reference

Data Structures

- class SmartStay.Core.Repositories.Reservations

  Represents a collection of Reservation objects, managed in a dictionary for fast lookup by reservation ID.

Namespaces

- namespace SmartStay
- namespace SmartStay.Core
- namespace SmartStay.Core.Repositories

  The `SmartStay.Repositories` namespace provides data access layers for retrieving and storing application data. It contains repositories that manage database interactions for various entities within the SmartStay application.

7.80 Reservations.cs

Go to the documentation of this file.

```
00001
00011 #nullable enable
00012 using System.Runtime.Serialization;
00013 using ProtoBuf;
00014 using SmartStay.Common.Models;
00015 using SmartStay.Core.Models;
00016 using SmartStay.Core.Models.Interfaces;
00017 using SmartStay.Core.Utilities;
00018
00023 namespace SmartStay.Core.Repositories
00024 {
00029 [ProtoContract]
00030 public class Reservations : IManageableEntity<Reservation>
00031 {
00035     readonly Dictionary<int, Reservation> _reservationDictionary = new Dictionary<int, Reservation>();
00036
00043     [ProtoMember(1)] // Serialize the list of accommodations
00044     List<Reservation> _reservationList = new();
00045
00057     public bool Add(Reservation reservation)
00058     {
00059         if (reservation == null)
00060         {
00061             throw new ArgumentNullException(nameof(reservation), "Reservation cannot be null");
00062         }
00063
00064         if (_reservationDictionary.ContainsKey(reservation.Id))
00065         {
00066             return false; // Reservation already exists
00067         }
00068
00069         _reservationDictionary[reservation.Id] = reservation;
00070         return true; // Reservation added successfully
00071     }
00072
00082     public bool Remove(Reservation reservation)
00083     {
00084         if (reservation == null)
00085         {
00086             throw new ArgumentNullException(nameof(reservation), "Reservation cannot be null");
00087         }
00088
00089         // Remove the reservation using its ID
00090         return _reservationDictionary.Remove(reservation.Id);
00091     }
00092
00103     public ImportResult Import(string data)
00104     {
00105         if (string.IsNullOrEmpty(data))
00106         {
00107             throw new ArgumentException("Data cannot be null or empty", nameof(data));
00108         }
00109
00110         // Deserialize the data into a List<Reservation> instead of a single Reservation
00111         var reservations = JsonHelper.DeserializeFromJson<Reservation>(data) ??
00112                             throw new ArgumentException("Deserialized reservation data cannot be null",
00113     nameof(data));
00114         int replacedCount = 0;
00115         int importedCount = 0;
00116
00117         foreach (var reservation in reservations)
00118         {
00119             if (_reservationDictionary.ContainsKey(reservation.Id))
00120             {
00121                 replacedCount++;
00122             }
00123             else
00124             {
00125                 importedCount++;
00126             }
00127             _reservationDictionary[reservation.Id] = reservation; // Direct insertion for efficiency
00128         }
00129
00130         return new ImportResult { ImportedCount = importedCount, ReplacedCount = replacedCount };
00131     }
00132
00137     public string Export()
00138     {
00139         return JsonHelper.SerializeToJson<Reservation>(_reservationDictionary.Values);
00140     }
00141
```

```
00147      [ProtoBeforeSerialization]
00148      private void PrepareForSerialization()
00149      {
00150          // Clear the temporary list to ensure no leftover data
00151          _reservationList.Clear();
00152
00153          // Add all reservations from the dictionary to the temporary list
00154          foreach (var reservation in _reservationDictionary.Values)
00155          {
00156              _reservationList.Add(reservation);
00157          }
00158      }
00159
00164      [ProtoAfterDeserialization]
00165      [System.Diagnostics.CodeAnalysis.SuppressMessage("CodeQuality", "IDE0051:Remove unused private
     members",
00166                                                    Justification =
00167                                                    "IDE Error, this is called automatically by
     protobuf-net.")]
00168      private void AfterDeserialization()
00169      {
00170          // Clear the dictionary before rebuilding
00171          _reservationDictionary.Clear();
00172
00173          // Rebuild the dictionary using the data from the list
00174          foreach (var reservation in _reservationList)
00175          {
00176              _reservationDictionary[reservation.Id] = reservation;
00177          }
00178
00179          // Clear the temporary list once the dictionary is rebuilt
00180          _reservationList.Clear();
00181
00182          // Set _lastReservationId to the highest ID in the deserialized data
00183          if (_reservationDictionary.Count > 0)
00184          {
00185              // Find the highest ID from the loaded reservations
00186              Reservation.LastAssignedId = _reservationDictionary.Values.Max(r => r.Id);
00187          }
00188          else
00189          {
00190              // If no reservations, reset to 0
00191              Reservation.LastAssignedId = 0;
00192          }
00193      }
00194
00204      public void Save(string filePath)
00205      {
00206          try
00207          {
00208              // Prepare for serialization by copying the dictionary contents to the temporary list
00209              PrepareForSerialization();
00210
00211              // Open the file stream for saving the data to the specified file
00212              using (var fileStream = File.Create(filePath))
00213              {
00214                  // Serialize the reservations object and write it to the file stream
00215                  Serializer.Serialize(fileStream, this);
00216              }
00217          }
00218          catch (IOException ioEx)
00219          {
00220              throw new IOException("An error occurred while saving the reservations data.", ioEx);
00221          }
00222          catch (SerializationException serEx)
00223          {
00224              throw new SerializationException(
00225                  "An error occurred during serialization while saving the reservations data.", serEx);
00226          }
00227      }
00228
00237      public void Load(string filePath)
00238      {
00239          try
00240          {
00241              // Open the file stream for reading
00242              using (var fileStream = File.OpenRead(filePath))
00243              {
00244                  // Deserialize the reservations object from the file
00245                  var reservations = Serializer.Deserialize<Reservations>(fileStream);
00246
00247                  // Copy the data from the deserialized object to the current instance
00248                  _reservationDictionary.Clear();
00249                  foreach (var reservation in reservations._reservationDictionary)
00250                  {
00251                      _reservationDictionary[reservation.Key] = reservation.Value;
00252                  }
```

```
00253                }
00254            }
00255            catch (IOException ioEx)
00256            {
00257                throw new IOException("An error occurred while loading the reservations data.", ioEx);
00258            }
00259            catch (SerializationException serEx)
00260            {
00261                throw new SerializationException(
00262                    "An error occurred during deserialization while loading the reservations data.",
        serEx);
00263            }
00264        }
00265
00273    public Reservation? FindReservationById(int reservationId)
00274    {
00275        _reservationDictionary.TryGetValue(reservationId, out Reservation? reservation);
00276        return reservation;
00277    }
00278
00284    public IEnumerable<Reservation> FindReservationsByClientId(int clientId)
00285    {
00286        return _reservationDictionary.Values.Where(r => r.ClientId == clientId);
00287    }
00288
00297    public IEnumerable<Reservation> FindReservationsByAccommodationId(int accommodationId)
00298    {
00299        return _reservationDictionary.Values.Where(r => r.AccommodationId == accommodationId);
00300    }
00301
00307    public IEnumerable<Reservation> GetFutureReservations(int accommodationId)
00308    {
00309        // Use LINQ to filter the dictionary's values directly without copying to a list.
00310        return _reservationDictionary.Values
00311            .Where(reservation =>
00312                        reservation.AccommodationId == accommodationId && reservation.CheckInDate >=
        DateTime.Now)
00313            .ToList();
00314    }
00315
00322    public int CountReservations()
00323    {
00324        return _reservationDictionary.Count;
00325    }
00326 }
00327 }
```

7.81 BookingManager.cs File Reference

Data Structures

- class SmartStay.Core.Services.BookingManager

  Provides a facade for managing clients, reservations, and accommodations in the booking system. This class centralizes all operations for adding, removing, importing, and exporting data for these entities. It interacts with internal repositories to simplify the main API and ensure a standardized approach.

Namespaces

- namespace SmartStay
- namespace SmartStay.Core
- namespace SmartStay.Core.Services

  The `Core.Services` *namespace contains service classes that implement business logic for the SmartStay application. These services coordinate actions between repositories and models to fulfill application requirements.*

7.82 BookingManager.cs

```
00001
00011 #nullable enable
00012 using Microsoft.Extensions.Logging;
00013 using SmartStay.Common.Enums;
00014 using SmartStay.Common.Exceptions;
00015 using SmartStay.Core.Models;
00016 using SmartStay.Core.Repositories;
00017 using SmartStay.Core.Utilities;
00018 using SmartStay.Validation;
00019
00024 namespace SmartStay.Core.Services
00025 {
00035 public class BookingManager
00036 {
00037 #region Fields and Properties
00038
00042     readonly ILogger<BookingManager> _logger;
00043
00047     internal readonly Owners _owners = new();
00048
00052     internal readonly Clients _clients = new();
00053
00057     internal readonly Reservations _reservations = new();
00058
00062     internal readonly Accommodations _accommodations = new();
00063
00064 #endregion
00065
00066 #region Constructor
00067
00072     public BookingManager(ILogger<BookingManager> logger)
00073     {
00074         _logger = logger ?? throw new ArgumentNullException(nameof(logger));
00075         _logger.LogInformation("BookingManager initialized.");
00076     }
00077
00078 #endregion
00079
00080 #region Accessors for Repositories
00081
00085     public Owners Owners => _owners;
00086
00090     public Clients Clients => _clients;
00091
00095     public Reservations Reservations => _reservations;
00096
00100     public Accommodations Accommodations => _accommodations;
00101
00102 #endregion
00103
00104 #region Helper Functions
00105
00113     private void SetField<T>(T fieldValue, T defaultValue, Action<T> setterAction)
00114     {
00115         // Only set the field if the value is not the default value
00116         if (!EqualityComparer<T>.Default.Equals(fieldValue, defaultValue))
00117         {
00118             setterAction(fieldValue); // Call the setter action with the field value
00119             _logger.LogInformation("Field set to {FieldValue}.", fieldValue); // Log the field change
00120         }
00121         else
00122         {
00123             _logger.LogInformation($"Field value is unchanged, remaining as default.");
00124         }
00125     }
00126
00127 #endregion
00128
00129 #region System Management
00130
00134     public void SaveAll(string dataFolder)
00135     {
00136         try
00137         {
00138             // Define file paths for each repository
00139             string clientsFilePath = Path.Combine(dataFolder, "clients.dat");
00140             string accommodationsFilePath = Path.Combine(dataFolder, "accommodations.dat");
00141             string reservationsFilePath = Path.Combine(dataFolder, "reservations.dat");
00142             string ownersFilePath = Path.Combine(dataFolder, "owners.dat");
00143
00144             // Save each repository to its corresponding file
00145             _clients.Save(clientsFilePath);
```

```
00146                  _accommodations.Save(accommodationsFilePath);
00147                  _reservations.Save(reservationsFilePath);
00148                  _owners.Save(ownersFilePath);
00149
00150                  _logger.LogInformation("All repositories saved successfully.");
00151              }
00152          catch (Exception ex)
00153          {
00154                  _logger.LogError(ex, "Error occurred while saving all repositories:");
00155                  throw new InvalidOperationException("Error occurred while saving all repositories.", ex);
00156          }
00157      }
00158
00162      public void LoadAll(string dataFolder)
00163      {
00164          try
00165          {
00166              // Define file paths for each repository
00167              string clientsFilePath = Path.Combine(dataFolder, "clients.dat");
00168              string accommodationsFilePath = Path.Combine(dataFolder, "accommodations.dat");
00169              string reservationsFilePath = Path.Combine(dataFolder, "reservations.dat");
00170              string ownersFilePath = Path.Combine(dataFolder, "owners.dat");
00171
00172              // Load each repository from its corresponding file
00173              if (File.Exists(clientsFilePath))
00174                  _clients.Load(clientsFilePath);
00175              if (File.Exists(accommodationsFilePath))
00176                  _accommodations.Load(accommodationsFilePath);
00177              if (File.Exists(reservationsFilePath))
00178                  _reservations.Load(reservationsFilePath);
00179              if (File.Exists(ownersFilePath))
00180                  _owners.Load(ownersFilePath);
00181
00182              _logger.LogInformation("All repositories loaded successfully.");
00183          }
00184          catch (Exception ex)
00185          {
00186                  _logger.LogError(ex, "Error occurred while loading all repositories.");
00187                  throw new InvalidOperationException("Error occurred while loading all repositories.", ex);
00188          }
00189      }
00190
00191 #endregion
00192
00193 #region Client Management
00194
00207      public Client CreateBasicClient(string firstName, string lastName, string email)
00208      {
00209          try
00210          {
00211              // Log information before creating the client
00212              _logger.LogInformation(
00213                  "Attempting to create a new client with Name: {FirstName} {LastName}, Email:
    {Email}.", firstName,
00214                      lastName, email);
00215
00216              // Create a new client
00217              Client client = new(firstName, lastName, email); // May throw exception
00218
00219              // Add client to the system
00220              _clients.Add(client);
00221
00222              // Log success information
00223              _logger.LogInformation("Successfully created client: {FirstName} {LastName}, Email:
    {Email}.", firstName,
00224                                        lastName, email);
00225
00226              // Return the created client
00227              return client;
00228          }
00229          catch (ValidationException ex)
00230          {
00231              // Log the exception with details
00232              _logger.LogError(ex, "Error while creating client with Name: {FirstName} {LastName},
    Email: {Email}.",
00233                                  firstName, lastName, email);
00234
00235              // Throw a new exception with more context
00236              throw new ClientCreationException("An error occurred while creating the client due to
    invalid input.", ex);
00237          }
00238      }
00239
00254      public Client CreateCompleteClient(string firstName, string lastName, string email, string
    phoneNumber,
00255                                          string address)
00256      {
```

```
00257          try
00258          {
00259              // Log information before creating the client
00260              _logger.LogInformation(
00261                  "Attempting to create a new client with Name: {FirstName} {LastName}, Email: {Email},
      " +
00262                      "Phone Number: {PhoneNumber}, Address: {Address}.",
00263                  firstName, lastName, email, phoneNumber, address);
00264
00265              // Create a new client with the provided information
00266              Client client = new(firstName, lastName, email, phoneNumber, address); // May throw
      exception
00267
00268              // Add client to the system
00269              _clients.Add(client);
00270
00271              // Log success information
00272              _logger.LogInformation("Successfully created client: {FirstName} {LastName}, Email:
      {Email}, Phone: " +
00273                                  "{PhoneNumber}, Address: {Address}.",
00274                                  firstName, lastName, email, phoneNumber, address);
00275
00276              // Return the created client
00277              return client;
00278          }
00279          catch (ValidationException ex)
00280          {
00281              // Log the exception with details
00282              _logger.LogError(ex,
00283                          "Error while creating client with Name: {FirstName} {LastName}, Email:
      {Email}, " +
00284                          "Phone Number: {PhoneNumber}, Address: {Address}.",
00285                          firstName, lastName, email, phoneNumber, address);
00286
00287              // Throw a new exception with more context
00288              throw new ClientCreationException("An error occurred while creating the client due to
      invalid input.", ex);
00289          }
00290      }
00291
00298      public Client FindClientById(int clientId)
00299      {
00300          _logger.LogInformation("Attempting to find client with ID: {ClientId}.", clientId);
00301
00302          var client = _clients.FindClientById(clientId);
00303
00304          if (client != null)
00305          {
00306              _logger.LogInformation("Successfully found client with ID: {ClientId}.", clientId);
00307              return client;
00308          }
00309          else
00310          {
00311              _logger.LogError("Error finding client with ID: {ClientId}.", clientId);
00312              throw new ArgumentException($"Client with ID {clientId} not found.");
00313          }
00314      }
00315
00333      public UpdateClientResult UpdateClient(int clientId, string? firstName = null, string? lastName =
      null,
00334                                          string? email = null, string? phoneNumber = null, string?
      address = null,
00335                                          PaymentMethod paymentMethod = PaymentMethod.Unchanged)
00336      {
00337          _logger.LogInformation("Attempting to update client with ID: {ClientId}.", clientId);
00338
00339          // Find the client by ID
00340          var client = _clients.FindClientById(clientId);
00341          if (client == null)
00342          {
00343              _logger.LogWarning("Client with ID: {ClientId} not found.", clientId);
00344              return UpdateClientResult.ClientNotFound;
00345          }
00346
00347          // Validate information before updating (only if not null or default value)
00348          if (firstName != null && !Validation.Validators.NameValidator.IsValidName(firstName))
00349          {
00350              _logger.LogError("Invalid first name provided for client ID: {ClientId}.", clientId);
00351              return UpdateClientResult.InvalidFirstName;
00352          }
00353
00354          if (lastName != null && !Validation.Validators.NameValidator.IsValidName(lastName))
00355          {
00356              _logger.LogError("Invalid last name provided for client ID: {ClientId}.", clientId);
00357              return UpdateClientResult.InvalidLastName;
00358          }
00359
```

```
00360          if (email != null && !Validation.Validators.EmailValidator.IsValidEmail(email))
00361          {
00362              _logger.LogError("Invalid email provided for client ID: {ClientId}.", clientId);
00363              return UpdateClientResult.InvalidEmail;
00364          }
00365
00366          if (phoneNumber != null &&
      !Validation.Validators.PhoneNumberValidator.IsValidPhoneNumber(phoneNumber))
00367          {
00368              _logger.LogError("Invalid phone number provided for client ID: {ClientId}.", clientId);
00369              return UpdateClientResult.InvalidPhoneNumber;
00370          }
00371
00372          if (address != null && !Validation.Validators.AddressValidator.IsValidAddress(address))
00373          {
00374              _logger.LogError("Invalid address provided for client ID: {ClientId}.", clientId);
00375              return UpdateClientResult.InvalidAddress;
00376          }
00377
00378          if (paymentMethod != PaymentMethod.Unchanged &&
00379              !Validation.Validators.PaymentValidator.IsValidPaymentMethod(paymentMethod))
00380          {
00381              _logger.LogError("Invalid payment method provided for client ID: {ClientId}.", clientId);
00382              return UpdateClientResult.InvalidPaymentMethod;
00383          }
00384
00385          // Log success before updating
00386          _logger.LogInformation("Validations passed. Updating details for client ID: {ClientId}.",
      clientId);
00387
00388          // Update client information with given fields, if not null or default
00389          SetField(firstName, null,
00390                  value =>
00391                  {
00392                      if (firstName != null && value != null)
00393                          client.FirstName = value;
00394                  });
00395          SetField(lastName, null,
00396                  value =>
00397                  {
00398                      if (lastName != null && value != null)
00399                          client.LastName = value;
00400                  });
00401          SetField(email, null,
00402                  value =>
00403                  {
00404                      if (email != null && value != null)
00405                          client.Email = value;
00406                  });
00407          SetField(phoneNumber, null,
00408                  value =>
00409                  {
00410                      if (phoneNumber != null && value != null)
00411                          client.PhoneNumber = value;
00412                  });
00413          SetField(address, null,
00414                  value =>
00415                  {
00416                      if (address != null && value != null)
00417                          client.Address = value;
00418                  });
00419          SetField(paymentMethod, PaymentMethod.Unchanged,
00420                  value =>
00421                  {
00422                      if (paymentMethod != PaymentMethod.Unchanged)
00423                          client.PreferredPaymentMethod = value;
00424                  });
00425
00426          _logger.LogInformation("Successfully updated client details for client ID: {ClientId}.",
      clientId);
00427
00428          return UpdateClientResult.Success;
00429      }
00430
00436      public bool RemoveClient(int clientId)
00437      {
00438          _logger.LogInformation("Attempting to remove client with ID: {ClientId}.", clientId);
00439
00440          // Find the client by ID
00441          var client = _clients.FindClientById(clientId);
00442
00443          if (client == null)
00444          {
00445              _logger.LogWarning("Client with ID: {ClientId} not found. No removal performed.",
      clientId);
00446              return false;
00447          }
```

```
00448
00449          // Remove the client from the list
00450          _clients.Remove(client);
00451          _logger.LogInformation("Successfully removed client with ID: {ClientId}.", clientId);
00452
00453          return true;
00454      }
00455
00456 #endregion
00457
00458 #region Owner Management
00459
00472      public Owner CreateBasicOwner(string firstName, string lastName, string email)
00473      {
00474          try
00475          {
00476              // Log information before creating the owner
00477              _logger.LogInformation(
00478                  "Attempting to create a new owner with Name: {FirstName} {LastName}, Email: {Email}.",
      firstName,
00479                  lastName, email);
00480
00481              // Create a new owner
00482              Owner owner = new(firstName, lastName, email); // May throw exception
00483
00484              // Add owner to the system
00485              _owners.Add(owner);
00486
00487              // Log success information
00488              _logger.LogInformation("Successfully created owner: {FirstName} {LastName}, Email:
      {Email}.", firstName,
00489                                       lastName, email);
00490
00491              // Return the created owner
00492              return owner;
00493          }
00494          catch (ValidationException ex)
00495          {
00496              // Log the exception with details
00497              _logger.LogError(ex, "Error while creating owner with Name: {FirstName} {LastName}, Email:
      {Email}.",
00498                                 firstName, lastName, email);
00499
00500              // Throw a new exception with more context
00501              throw new OwnerCreationException("An error occurred while creating the owner due to
      invalid input.", ex);
00502          }
00503      }
00504
00519      public Owner CreateCompleteOwner(string firstName, string lastName, string email, string
      phoneNumber,
00520                                        string address)
00521      {
00522          try
00523          {
00524              // Log information before creating the owner
00525              _logger.LogInformation(
00526                  "Attempting to create a new owner with Name: {FirstName} {LastName}, Email: {Email}, "
      +
00527                      "Phone Number: {PhoneNumber}, Address: {Address}.",
00528                  firstName, lastName, email, phoneNumber, address);
00529
00530              // Create a new owner with the provided information
00531              Owner owner = new(firstName, lastName, email, phoneNumber, address); // May throw
      exception
00532
00533              // Add owner to the system
00534              _owners.Add(owner);
00535
00536              // Log success information
00537              _logger.LogInformation("Successfully created owner: {FirstName} {LastName}, Email:
      {Email}.", firstName,
00538                                       lastName, email);
00539
00540              // Return the created owner
00541              return owner;
00542          }
00543          catch (ValidationException ex)
00544          {
00545              // Log the exception with details
00546              _logger.LogError(ex,
00547                                 "Error while creating owner with Name: {FirstName} {LastName}, Email:
      {Email}, " +
00548                                     "Phone Number: {PhoneNumber}, Address: {Address}.",
00549                                 firstName, lastName, email, phoneNumber, address);
00550
00551              // Throw a new exception with more context
```

```
00552                    throw new OwnerCreationException("An error occurred while creating the owner due to
       invalid input.", ex);
00553            }
00554       }
00555
00562       public Owner FindOwnerById(int ownerId)
00563       {
00564            // Log information before attempting to find the owner
00565            _logger.LogInformation("Attempting to find owner with ID: {OwnerId}.", ownerId);
00566
00567            var owner = _owners.FindOwnerById(ownerId);
00568
00569            if (owner != null)
00570            {
00571                // Log success information if the owner is found
00572                _logger.LogInformation("Successfully found owner with ID: {OwnerId}.", ownerId);
00573                return owner;
00574            }
00575            else
00576            {
00577                // Log error if the owner is not found
00578                _logger.LogError("Error finding owner with ID: {OwnerId}.", ownerId);
00579                throw new ArgumentException($"Owner with ID {ownerId} not found.");
00580            }
00581       }
00582
00599       public UpdateOwnerResult UpdateOwner(int ownerId, string? firstName = null, string? lastName =
       null,
00600                                          string? email = null, string? phoneNumber = null, string?
       address = null)
00601       {
00602            _logger.LogInformation("Attempting to update owner with ID: {OwnerId}.", ownerId);
00603
00604            // Find the owner by ID
00605            var owner = _owners.FindOwnerById(ownerId);
00606            if (owner == null)
00607            {
00608                _logger.LogWarning("Owner with ID: {OwnerId} not found.", ownerId);
00609                return UpdateOwnerResult.OwnerNotFound;
00610            }
00611
00612            // Validate information before updating (only if not null)
00613            if (firstName != null && !Validation.Validators.NameValidator.IsValidName(firstName))
00614            {
00615                _logger.LogError("Invalid first name provided for owner ID: {OwnerId}.", ownerId);
00616                return UpdateOwnerResult.InvalidFirstName;
00617            }
00618
00619            if (lastName != null && !Validation.Validators.NameValidator.IsValidName(lastName))
00620            {
00621                _logger.LogError("Invalid last name provided for owner ID: {OwnerId}.", ownerId);
00622                return UpdateOwnerResult.InvalidLastName;
00623            }
00624
00625            if (email != null && !Validation.Validators.EmailValidator.IsValidEmail(email))
00626            {
00627                _logger.LogError("Invalid email provided for owner ID: {OwnerId}.", ownerId);
00628                return UpdateOwnerResult.InvalidEmail;
00629            }
00630
00631            if (phoneNumber != null &&
       !Validation.Validators.PhoneNumberValidator.IsValidPhoneNumber(phoneNumber))
00632            {
00633                _logger.LogError("Invalid phone number provided for owner ID: {OwnerId}.", ownerId);
00634                return UpdateOwnerResult.InvalidPhoneNumber;
00635            }
00636
00637            if (address != null && !Validation.Validators.AddressValidator.IsValidAddress(address))
00638            {
00639                _logger.LogError("Invalid address provided for owner ID: {OwnerId}.", ownerId);
00640                return UpdateOwnerResult.InvalidAddress;
00641            }
00642
00643            // Log success before updating
00644            _logger.LogInformation("Validations passed. Updating details for owner ID: {OwnerId}.",
       ownerId);
00645
00646            // Update owner information with given fields, only if not null
00647            SetField(firstName, null,
00648                    value =>
00649                    {
00650                        if (firstName != null && value != null)
00651                            owner.FirstName = value;
00652                    });
00653            SetField(lastName, null,
00654                    value =>
00655                    {
```

```
00656                         if (lastName != null && value != null)
00657                             owner.LastName = value;
00658                     });
00659             SetField(email, null,
00660                     value =>
00661                     {
00662                         if (email != null && value != null)
00663                             owner.Email = value;
00664                     });
00665             SetField(phoneNumber, null,
00666                     value =>
00667                     {
00668                         if (phoneNumber != null && value != null)
00669                             owner.PhoneNumber = value;
00670                     });
00671             SetField(address, null,
00672                     value =>
00673                     {
00674                         if (address != null && value != null)
00675                             owner.Address = value;
00676                     });
00677
00678         _logger.LogInformation("Successfully updated owner details for owner ID: {OwnerId}.",
      ownerId);
00679
00680             return UpdateOwnerResult.Success;
00681     }
00682
00688     public bool RemoveOwner(int ownerId)
00689     {
00690         _logger.LogInformation("Attempting to remove owner with ID: {OwnerId}.", ownerId);
00691
00692         var owner = _owners.FindOwnerById(ownerId);
00693         if (owner != null)
00694         {
00695             _owners.Remove(owner);
00696             _logger.LogInformation("Successfully removed owner with ID: {OwnerId}.", ownerId);
00697             return true;
00698         }
00699         else
00700         {
00701             _logger.LogWarning("Owner with ID: {OwnerId} not found, unable to remove.", ownerId);
00702             return false;
00703         }
00704     }
00705
00706 #endregion
00707
00708 #region Reservation Management
00709
00748     public Reservation CreateReservation(int clientId, int accommodationId, int roomId, DateTime
      checkIn,
00749                                          DateTime checkOut)
00750     {
00751         // Log the attempt to create a reservation
00752         _logger.LogInformation("Attempting to create reservation for client {ClientId} at
      accommodation " +
00753                                "{AccommodationId}, room {RoomId}, from {CheckIn} to {CheckOut}.",
00754                         clientId, accommodationId, roomId, checkIn, checkOut);
00755
00756         // Find the accommodation
00757         var accommodation = _accommodations.FindAccommodationById(accommodationId);
00758         if (accommodation == null)
00759         {
00760             _logger.LogError("Accommodation with ID {AccommodationId} not found.", accommodationId);
00761             throw new ArgumentException("Accommodation not found.");
00762         }
00763
00764         // Find the room in the accommodation
00765         var room = accommodation.FindRoomById(roomId);
00766         if (room == null)
00767         {
00768             _logger.LogError("Room with ID {RoomId} not found in accommodation {AccommodationId}.",
      roomId,
00769                             accommodationId);
00770             throw new ArgumentException("Room not found.");
00771         }
00772
00773         // Check if room is available upfront for a quick exit. AddReservation will validate again to
      ensure
00774         // consistency.
00775         var available = room.IsAvailable(checkIn, checkOut);
00776         if (!available)
00777         {
00778             _logger.LogError("Room {RoomId} in accommodation {AccommodationId} is not available for
      the dates " +
00779                             "{CheckIn} to {CheckOut}.",
```

```
00780                                    roomId, accommodationId, checkIn, checkOut);
00781                     throw new ArgumentException("Accommodation is not available for the selected dates.");
00782             }
00783
00784         // Calculate the total cost of the reservation
00785         decimal totalCost;
00786         try
00787         {
00788             totalCost = room.CalculateTotalCost(checkIn, checkOut);
00789         }
00790         catch (ArgumentException ex)
00791         {
00792             _logger.LogError(ex, "Error calculating total cost for reservation: {Message}",
      ex.Message);
00793                 throw new TotalCostException("An error occurred while calculating the total cost for the
      reservation.", ex);
00794         }
00795
00796         // Log the total cost calculation
00797         _logger.LogInformation(
00798             "Calculated total cost for reservation: {TotalCost} for {ClientId} from {CheckIn} to
      {CheckOut}.",
00799             totalCost, clientId, checkIn, checkOut);
00800
00801         // Create the reservation
00802         Reservation reservation;
00803         try
00804         {
00805             reservation =
00806                 new Reservation(clientId, accommodationId, roomId, accommodation.Type, checkIn,
      checkOut, totalCost);
00807         }
00808         catch (ValidationException ex)
00809         {
00810             _logger.LogError(ex,
00811                             "Validation failed when creating reservation for client {ClientId} at
      accommodation " +
00812                             "{AccommodationId}, room {RoomId}. Error: {ErrorCode}",
00813                             clientId, accommodationId, roomId, ex.ErrorCode);
00814             throw new ReservationCreationException(
00815                 $"An error occurred while creating the reservation due to invalid input.", ex);
00816         }
00817
00818         // Attempt to add the reservation to the room (this also checks for availability)
00819         bool success = room.AddReservation(checkIn, checkOut);
00820         if (!success)
00821         {
00822             _logger.LogError("Room {RoomId} in accommodation {AccommodationId} is not available for
      the dates " +
00823                             "{CheckIn} to {CheckOut}.",
00824                             roomId, accommodationId, checkIn, checkOut);
00825             throw new ArgumentException("Accommodation is not available for the selected dates.");
00826         }
00827
00828         // Add the reservation to the reservation list
00829         bool reservationAdded = _reservations.Add(reservation);
00830         if (!reservationAdded)
00831         {
00832             _logger.LogError("Failed to add reservation for client {ClientId} at accommodation " +
00833                             "{AccommodationId}, room {RoomId}. Reservation may already exist.",
00834                             clientId, accommodationId, roomId);
00835             throw new ArgumentException("Failed to add reservation.");
00836         }
00837
00838         // Log successful reservation creation
00839         _logger.LogInformation("Successfully created reservation {ReservationId} for client {ClientId}
      at " +
00840                                "accommodation {AccommodationId}, room {RoomId}. Total cost:
      {TotalCost}.",
00841                                reservation.Id, clientId, accommodationId, roomId, totalCost);
00842
00843         // Return the newly created reservation
00844         return reservation;
00845     }
00846
00867     public UpdateReservationResult UpdateReservation(int reservationId, DateTime? newCheckIn = null,
00868                                                     DateTime? newCheckOut = null)
00869     {
00870         _logger.LogInformation("Attempting to update reservation {ReservationId}.", reservationId);
00871
00872         try
00873         {
00874             // Use helper to find the reservation
00875             var reservation = FindReservation(reservationId);
00876
00877             // Use helper to find accommodation and room
00878             var (accommodation, room) = FindAssociatedEntities(reservation);
```

```
00879
00880            // Determine effective check-in and check-out dates
00881            var (effectiveCheckIn, effectiveCheckOut) = GetEffectiveDates(reservation, newCheckIn,
     newCheckOut);
00882
00883            _logger.LogInformation(
00884                "Effective dates for reservation {ReservationId}: Check-In: {CheckIn}, Check-Out:
     {CheckOut}.",
00885                reservationId, effectiveCheckIn, effectiveCheckOut);
00886
00887            // Check availability using helper
00888            if (!IsAvailableForUpdate(room, reservation.CheckInDate, reservation.CheckOutDate,
     effectiveCheckIn,
00889                                      effectiveCheckOut))
00890            {
00891                _logger.LogWarning(
00892                    "Room {RoomId} in accommodation {AccommodationId} is unavailable for the new
     dates: " +
00893                    "Check-In: {CheckIn}, Check-Out: {CheckOut}.",
00894                    room.Id, accommodation.Id, effectiveCheckIn, effectiveCheckOut);
00895                return UpdateReservationResult.DatesUnavailable;
00896            }
00897
00898            // Update room reservation (handles removing and adding)
00899            UpdateRoomReservation(room, reservation.CheckInDate, reservation.CheckOutDate,
     effectiveCheckIn,
00900                                  effectiveCheckOut);
00901
00902            // Update reservation dates
00903            reservation.CheckInDate = effectiveCheckIn;
00904            reservation.CheckOutDate = effectiveCheckOut;
00905
00906            _logger.LogInformation("Successfully updated reservation {ReservationId}.",
     reservationId);
00907            return UpdateReservationResult.Success;
00908        }
00909        catch (EntityNotFoundException ex) when (ex.EntityType == nameof(Reservation))
00910        {
00911            _logger.LogWarning(ex, "Reservation with ID {ReservationId} not found.", reservationId);
00912            return UpdateReservationResult.ReservationNotFound;
00913        }
00914        catch (EntityNotFoundException ex) when (ex.EntityType == nameof(Accommodation))
00915        {
00916            _logger.LogWarning(ex, "Accommodation for reservation {ReservationId} not found.",
     reservationId);
00917            return UpdateReservationResult.AccommodationNotFound;
00918        }
00919        catch (EntityNotFoundException ex) when (ex.EntityType == nameof(Room))
00920        {
00921            _logger.LogWarning(ex, "Room for reservation {ReservationId} not found.", reservationId);
00922            return UpdateReservationResult.RoomNotFound;
00923        }
00924        catch (ArgumentNullException ex)
00925        {
00926            _logger.LogWarning(ex, "Room was null for reservation {ReservationId}.", reservationId);
00927            return UpdateReservationResult.RoomNotFound;
00928        }
00929        catch (ArgumentException ex)
00930        {
00931            _logger.LogWarning(ex, "Invalid check-in/check-out dates for reservation
     {ReservationId}.", reservationId);
00932            return UpdateReservationResult.InvalidDates;
00933        }
00934        catch (Exception ex)
00935        {
00936            _logger.LogError(ex, "An error occurred while updating reservation {ReservationId}.",
     reservationId);
00937            return UpdateReservationResult.Error;
00938        }
00939    }
00940
00972    public CancellationResult CancelReservation(int reservationId)
00973    {
00974        _logger.LogInformation("Attempting to cancel reservation {ReservationId}.", reservationId);
00975
00976        try
00977        {
00978            // Use helper to find the reservation
00979            var reservation = FindReservation(reservationId);
00980
00981            // Use helper to find the associated accommodation and room
00982            var (accommodation, room) = FindAssociatedEntities(reservation);
00983
00984            // Attempt to remove the reservation from the room's reserved dates
00985            bool removeResult = room.RemoveReservation(reservation.CheckInDate,
     reservation.CheckOutDate);
00986            if (!removeResult)
```

```
00987                       {
00988                           _logger.LogError("Failed to remove reservation {ReservationId} from room {RoomId} in
          accommodation " +
00989                                             "{AccommodationId}.",
00990                               reservationId, room.Id, accommodation.Id);
00991                           return CancellationResult.Error;
00992                       }
00993
00994                       // Mark the reservation as cancelled
00995                       reservation.Status = ReservationStatus.Cancelled;
00996                       _logger.LogInformation("Successfully cancelled reservation {ReservationId}.",
          reservationId);
00997
00998                       return CancellationResult.Success;
00999                   }
01000               catch (EntityNotFoundException ex) when (ex.EntityType == nameof(Reservation))
01001               {
01002                   _logger.LogWarning(ex, "Reservation with ID {ReservationId} not found.", reservationId);
01003                   return CancellationResult.ReservationNotFound;
01004               }
01005               catch (EntityNotFoundException ex) when (ex.EntityType == nameof(Accommodation))
01006               {
01007                   _logger.LogWarning(ex, "Accommodation for reservation {ReservationId} not found.",
          reservationId);
01008                   return CancellationResult.AccommodationNotFound;
01009               }
01010               catch (EntityNotFoundException ex) when (ex.EntityType == nameof(Room))
01011               {
01012                   _logger.LogWarning(ex, "Room for reservation {ReservationId} not found.", reservationId);
01013                   return CancellationResult.RoomNotFound;
01014               }
01015               catch (Exception ex)
01016               {
01017                   _logger.LogError(ex, "An error occurred while cancelling reservation {ReservationId}.",
          reservationId);
01018                   return CancellationResult.Error;
01019               }
01020           }
01021
01022   #region Reservation Helper Functions
01023
01035       private bool IsAvailableForUpdate(Room room, DateTime currentStart, DateTime currentEnd, DateTime
          newStart,
01036                                        DateTime newEnd)
01037       {
01038           if (room == null)
01039               throw new ArgumentNullException(nameof(room), "Room cannot be null.");
01040
01041           if (newEnd <= newStart)
01042               throw new ArgumentException("New check-out date must be after the new check-in date.");
01043
01044           DateRange existingReservation = new(currentStart, currentEnd);
01045
01046           _logger?.LogDebug("Checking availability for update: Current [{CurrentStart} - {CurrentEnd}],
          Proposed " +
01047                             "[{NewStart} - {NewEnd}].",
01048                             currentStart, currentEnd, newStart, newEnd);
01049
01050           // Exclude the current reservation's dates from the availability check
01051           return room.IsAvailable(newStart, newEnd, existingReservation);
01052       }
01053
01060       private Reservation FindReservation(int reservationId)
01061       {
01062           _logger.LogInformation("Looking for reservation with ID {ReservationId}.", reservationId);
01063
01064           var reservation = _reservations.FindReservationById(reservationId);
01065           if (reservation == null)
01066           {
01067               _logger.LogWarning("Reservation with ID {ReservationId} not found.", reservationId);
01068               throw new EntityNotFoundException(nameof(Reservation), reservationId);
01069           }
01070
01071           _logger.LogInformation("Successfully found reservation with ID {ReservationId}.",
          reservationId);
01072           return reservation;
01073       }
01074
01083       private (Accommodation, Room) FindAssociatedEntities(Reservation reservation)
01084       {
01085           _logger.LogInformation("Finding associated accommodation and room for reservation ID
          {ReservationId}.",
01086                                  reservation.Id);
01087
01088           // Find accommodation
01089           var accommodation = _accommodations.FindAccommodationById(reservation.AccommodationId);
01090           if (accommodation == null)
```

```
01091            {
01092                _logger.LogWarning("Accommodation with ID {AccommodationId} not found for reservation
       {ReservationId}.",
01093                                 reservation.AccommodationId, reservation.Id);
01094                throw new EntityNotFoundException(nameof(Accommodation), reservation.AccommodationId);
01095            }
01096
01097            _logger.LogInformation(
01098                "Successfully found accommodation with ID {AccommodationId} for reservation
       {ReservationId}.",
01099                accommodation.Id, reservation.Id);
01100
01101            // Find room
01102            var room = accommodation.FindRoomById(reservation.RoomId);
01103            if (room == null)
01104            {
01105                _logger.LogWarning(
01106                    "Room with ID {RoomId} not found in accommodation {AccommodationId} for reservation
       {ReservationId}.",
01107                    reservation.RoomId, accommodation.Id, reservation.Id);
01108                throw new EntityNotFoundException(nameof(Room), reservation.RoomId);
01109            }
01110
01111            _logger.LogInformation("Successfully found room with ID {RoomId} in accommodation
       {AccommodationId} for " +
01112                                  "reservation {ReservationId}.",
01113                                  room.Id, accommodation.Id, reservation.Id);
01114
01115            return (accommodation, room);
01116        }
01117
01128        private (DateTime, DateTime) GetEffectiveDates(Reservation reservation, DateTime? newCheckIn,
       DateTime? newCheckOut)
01129        {
01130            _logger.LogInformation("Determining effective dates for reservation ID {ReservationId}.",
       reservation.Id);
01131
01132            DateTime effectiveCheckIn = newCheckIn ?? reservation.CheckInDate;
01133            DateTime effectiveCheckOut = newCheckOut ?? reservation.CheckOutDate;
01134
01135            if (effectiveCheckOut <= effectiveCheckIn)
01136            {
01137                _logger.LogError(
01138                    "Invalid dates for reservation {ReservationId}: Check-In: {CheckIn}, Check-Out:
       {CheckOut}.",
01139                    reservation.Id, effectiveCheckIn, effectiveCheckOut);
01140                throw new ArgumentException("Check-out date must be later than check-in date.");
01141            }
01142
01143            _logger.LogInformation(
01144                "Effective dates for reservation {ReservationId}: Check-In: {CheckIn}, Check-Out:
       {CheckOut}.",
01145                reservation.Id, effectiveCheckIn, effectiveCheckOut);
01146
01147            return (effectiveCheckIn, effectiveCheckOut);
01148        }
01149
01159        private void UpdateRoomReservation(Room room, DateTime currentStart, DateTime currentEnd, DateTime
       newStart,
01160                                          DateTime newEnd)
01161        {
01162            _logger.LogInformation(
01163                "Updating room reservation for Room {RoomId}: Removing dates {CurrentStart} to
       {CurrentEnd}, " +
01164                "and adding dates {NewStart} to {NewEnd}.",
01165                room.Id, currentStart, currentEnd, newStart, newEnd);
01166
01167            // Safely remove current reservation dates
01168            room.RemoveReservation(currentStart, currentEnd);
01169
01170            // Add the new reservation dates
01171            room.AddReservation(newStart, newEnd);
01172
01173            _logger.LogInformation(
01174                "Successfully updated room reservation for Room {RoomId} with new dates {NewStart} to
       {NewEnd}.", room.Id,
01175                newStart, newEnd);
01176        }
01177
01178 #endregion
01179
01180 #endregion
01181
01182 #region Accommodation Management
01183
01220        public Accommodation CreateAccommodation(int ownerId, AccommodationType type, string name, string
       address)
```

```
01221     {
01222         try
01223         {
01224             // Log information before creating the accommodation
01225             _logger.LogInformation("Attempting to create a new accommodation for owner ID {OwnerId}:
      Type: {Type}, " +
01226                                     "Name: {Name}, Address: {Address}.",
01227                                 ownerId, type, name, address);
01228
01229             // Find the owner by ID
01230             var owner = _owners.FindOwnerById(ownerId);
01231             if (owner == null)
01232             {
01233                 _logger.LogError("Owner with ID {OwnerId} not found.", ownerId);
01234                 throw new EntityNotFoundException(nameof(Owner), ownerId);
01235             }
01236
01237             // Create a new accommodation
01238             var accommodation = new Accommodation(ownerId, type, name, address); // May throw
      exception
01239
01240             // Attempt to add the accommodation to the owner's list of accommodations
01241             bool addSuccess = owner.AddAccommodation(accommodation);
01242             if (!addSuccess)
01243             {
01244                 _logger.LogError("Failed to add accommodation {Name} to owner ID {OwnerId}.", name,
      ownerId);
01245                 throw new OwnerAddAccommodationException("Failed to add accommodation to owner's
      list.");
01246             }
01247
01248             // Attempt to add the accommodation to the system
01249             bool addToSystemSuccess = _accommodations.Add(accommodation);
01250             if (!addToSystemSuccess)
01251             {
01252                 _logger.LogError("Failed to add accommodation {Name} to the system.", name);
01253                 throw new AddAccommodationSystemException(
01254                     "An error occurred while adding the accommodation to the system.");
01255             }
01256
01257             // Log success information
01258             _logger.LogInformation(
01259                 "Successfully created accommodation: {Type}, {Name}, Address: {Address} for owner ID
      {OwnerId}.", type,
01260                 name, address, ownerId);
01261
01262             // Return the created accommodation
01263             return accommodation;
01264         }
01265         catch (EntityNotFoundException ex)
01266         {
01267             // Log the exception with details
01268             _logger.LogError(ex,
01269                             "Error while creating accommodation for owner ID {OwnerId}: Type: {Type},
      Name: {Name}, " +
01270                             "Address: {Address}.",
01271                             ownerId, type, name, address);
01272
01273             // Rethrow the exception as a more specific one
01274             throw new AccommodationCreationException(
01275                 "An error occurred while creating the accommodation due to missing owner.", ex);
01276         }
01277         catch (OwnerAddAccommodationException ex)
01278         {
01279             // Log the exception if adding accommodation fails
01280             _logger.LogError(ex, "Error while adding accommodation {Name} for owner ID {OwnerId}.",
      name, ownerId);
01281             throw new AccommodationCreationException(
01282                 "An error occurred while trying to add accommodation to the owner accommodation
      list.");
01283         }
01284         catch (AddAccommodationSystemException ex)
01285         {
01286             // Log the exception if adding accommodation fails
01287             _logger.LogError(ex, "Error while adding accommodation {Name} to the system.", name);
01288             throw new AccommodationCreationException(
01289                 "An error occurred while trying to add accommodation to the system.");
01290         }
01291     }
01292
01305     public UpdateAccommodationResult UpdateAccommodation(int accommodationId,
01306                                                         AccommodationType type =
      AccommodationType.None,
01307                                                         string? name = null, string? address = null)
01308     {
01309         _logger.LogInformation("Attempting to update accommodation with ID: {AccommodationId}.",
      accommodationId);
```

```
01310
01311          // Find the accommodation by ID
01312          var accommodation = _accommodations.FindAccommodationById(accommodationId);
01313          if (accommodation == null)
01314          {
01315              _logger.LogWarning("Accommodation with ID: {AccommodationId} not found.",
       accommodationId);
01316              return UpdateAccommodationResult.AccommodationNotFound;
01317          }
01318
01319          // Validate information before updating (only if not default value)
01320          if (type != AccommodationType.None &&
01321              !Validation.Validators.AccommodationValidator.IsValidAccommodationType(type))
01322          {
01323              _logger.LogError("Invalid type provided for accommodation ID: {AccommodationId}.",
       accommodationId);
01324              return UpdateAccommodationResult.InvalidType;
01325          }
01326
01327          if (name != null && !Validation.Validators.NameValidator.IsValidAccommodationName(name))
01328          {
01329              _logger.LogError("Invalid name provided for accommodation ID: {AccommodationId}.",
       accommodationId);
01330              return UpdateAccommodationResult.InvalidName;
01331          }
01332
01333          if (address != null && !Validation.Validators.AddressValidator.IsValidAddress(address))
01334          {
01335              _logger.LogError("Invalid address provided for accommodation ID: {AccommodationId}.",
       accommodationId);
01336              return UpdateAccommodationResult.InvalidAddress;
01337          }
01338
01339          // Log success before updating
01340          _logger.LogInformation("Validations passed. Updating details for accommodation ID:
       {AccommodationId}.",
01341                                  accommodationId);
01342
01343          // Update accommodation information with given fields, only if not default
01344          SetField(type, AccommodationType.None,
01345                  value =>
01346                  {
01347                      if (type != AccommodationType.None)
01348                          accommodation.Type = value;
01349                  });
01350          SetField(name, null,
01351                  value =>
01352                  {
01353                      if (name != null && value != null)
01354                          accommodation.Name = value;
01355                  });
01356          SetField(address, null,
01357                  value =>
01358                  {
01359                      if (address != null && value != null)
01360                          accommodation.Address = value;
01361                  });
01362
01363          // Log success after updating
01364          _logger.LogInformation("Successfully updated accommodation details for accommodation ID:
       {AccommodationId}.",
01365                                  accommodationId);
01366
01367          return UpdateAccommodationResult.Success;
01368      }
01369
01382      public RemoveAccommodationResult RemoveAccommodation(int accommodationId)
01383      {
01384          _logger.LogInformation("Attempting to remove accommodation with ID: {AccommodationId}.",
       accommodationId);
01385
01386          try
01387          {
01388              // Find the accommodation by ID
01389              var accommodation = _accommodations.FindAccommodationById(accommodationId);
01390              if (accommodation == null)
01391              {
01392                  _logger.LogWarning("Accommodation with ID: {AccommodationId} not found, unable to
       remove.",
01393                                      accommodationId);
01394                  return RemoveAccommodationResult.AccommodationNotFound;
01395              }
01396
01397              // Find the owner by ID
01398              var owner = _owners.FindOwnerById(accommodation.OwnerId);
01399              if (owner == null)
01400              {
```

```
01401                    _logger.LogWarning("Owner with ID: {OwnerId} not found for accommodation ID:
       {AccommodationId}.",
01402                                        accommodation.OwnerId, accommodationId);
01403                    return RemoveAccommodationResult.OwnerNotFound;
01404                }
01405
01406                // Log before removal
01407                _logger.LogInformation("Owner with ID: {OwnerId} found. Removing accommodation ID:
       {AccommodationId}.",
01408                                        accommodation.OwnerId, accommodationId);
01409
01410                // Attempt to remove accommodation from the system
01411                bool removeSuccess = _accommodations.Remove(accommodation);
01412                if (!removeSuccess)
01413                {
01414                    _logger.LogError("Failed to remove accommodation ID: {AccommodationId} from the
       system.",
01415                                        accommodationId);
01416                    return RemoveAccommodationResult.AccommodationRemovalFailed;
01417                }
01418
01419                // Attempt to remove accommodation from the owner's list
01420                bool ownerRemoveSuccess = owner.RemoveAccommodation(accommodation);
01421                if (!ownerRemoveSuccess)
01422                {
01423                    _logger.LogError("Failed to disassociate accommodation ID: {AccommodationId} from
       owner ID: {OwnerId}.",
01424                                        accommodationId, accommodation.OwnerId);
01425                    return RemoveAccommodationResult.AccommodationDisassociationFailed;
01426                }
01427
01428                // Log success after removal
01429                _logger.LogInformation("Successfully removed accommodation ID: {AccommodationId} and
       disassociated it " +
01430                                        "from owner ID: {OwnerId}.",
01431                                        accommodationId, accommodation.OwnerId);
01432
01433                return RemoveAccommodationResult.Success;
01434            }
01435            catch (Exception ex)
01436            {
01437                // Log any unexpected errors
01438                _logger.LogError(ex, "Unexpected error occurred while removing accommodation with ID:
       {AccommodationId}.",
01439                                     accommodationId);
01440                return RemoveAccommodationResult.Error;
01441            }
01442        }
01443
01444 #endregion
01445 }
01446 }
```

7.83 DateRange.cs File Reference

Data Structures

- class SmartStay.Core.Utilities.DateRange

  Represents a range of dates with a start and end date. Implements IComparable<DateRange> to allow sorting and comparisons.

Namespaces

- namespace SmartStay
- namespace SmartStay.Core
- namespace SmartStay.Core.Utilities

  The `SmartStay.Utilities` *namespace provides helper functions and utility classes used throughout the* *SmartStay application. These utilities support common operations and enhance reusability across different components of the application.*

7.84   DateRange.cs

```
00001
00002 using ProtoBuf;
00003
00012
00018 namespace SmartStay.Core.Utilities
00019 {
00024 [ProtoContract]
00025 public class DateRange : IComparable<DateRange>
00026 {
00030     [ProtoMember(1)]
00031     public DateTime Start { get; set; }
00032
00036     [ProtoMember(2)]
00037     public DateTime End {
00038         get; set;
00039     }
00040
00046     public DateRange(DateTime start, DateTime end)
00047     {
00048         Start = start;
00049         End = end;
00050     }
00051
00062     public int CompareTo(DateRange? other)
00063     {
00064         if (other is null)
00065             return 1; // If other is null, this object is considered greater.
00066
00067         // First, compare by the Start date.
00068         int startComparison = Start.CompareTo(other.Start);
00069         if (startComparison != 0)
00070             return startComparison;
00071
00072         // If Start dates are the same, compare by End date.
00073         return End.CompareTo(other.End);
00074     }
00075
00082     public override bool Equals(object? obj)
00083     {
00084         if (obj is DateRange other)
00085         {
00086             // Check if both Start and End dates are equal.
00087             return Start == other.Start && End == other.End;
00088         }
00089         return false;
00090     }
00091
00096     public override int GetHashCode()
00097     {
00098         // Combine the hash codes of Start and End dates to get a unique hash code.
00099         return HashCode.Combine(Start, End);
00100     }
00101
00108     public static bool operator ==(DateRange left, DateRange right)
00109     {
00110         return left.Equals(right);
00111     }
00112
00119     public static bool operator !=(DateRange left, DateRange right)
00120     {
00121         return !(left == right);
00122     }
00123
00131     public static bool operator<(DateRange left, DateRange right)
00132     {
00133         return left.CompareTo(right) < 0;
00134     }
00135
00143     public static bool operator <=(DateRange left, DateRange right)
00144     {
00145         return left.CompareTo(right) <= 0;
00146     }
00147
00154     public static bool operator>(DateRange left, DateRange right)
00155     {
00156         return left.CompareTo(right) > 0;
00157     }
00158
00166     public static bool operator >=(DateRange left, DateRange right)
00167     {
00168         return left.CompareTo(right) >= 0;
00169     }
```

```
00170
00175    public DateRange Clone()
00176    {
00177        // Use the constructor to initialize the new DateRange
00178        return new DateRange(Start, End);
00179    }
00180 }
00181 }
```

7.85 JsonHelper.cs File Reference

Data Structures

- class **SmartStay.Core.Utilities.JsonHelper**

  Provides static methods to serialize and deserialize objects to and from JSON format.

Namespaces

- namespace SmartStay
- namespace SmartStay.Core
- namespace SmartStay.Core.Utilities

  The `SmartStay.Utilities` *namespace provides helper functions and utility classes used throughout the* *SmartStay* *application. These utilities support common operations and enhance reusability across different components of the application.*

7.86 JsonHelper.cs

Go to the documentation of this file.
```
00001
00009 using System.Collections.Generic;
00010 using System.Text.Json;
00011
00017 namespace SmartStay.Core.Utilities
00018 {
00022 public static class JsonHelper
00023 {
00027    private static readonly JsonSerializerOptions _jsonOptions = new() { WriteIndented = true };
00028
00039    public static string SerializeToJson<T>(IEnumerable<T> items)
00040    {
00041        return JsonSerializer.Serialize(items, _jsonOptions);
00042    }
00043
00054    public static List<T> DeserializeFromJson<T>(string json)
00055    {
00056        return JsonSerializer.Deserialize<List<T»(json) ?? new List<T>();
00057    }
00058 }
00059 }
```

7.87 FileExtensions.cs File Reference

Data Structures

- class **SmartStay.IO.Extensions.FileExtensions**

  Provides extension methods for file-related operations.

Namespaces

- namespace SmartStay
- namespace SmartStay.IO
- namespace SmartStay.IO.Extensions

  This namespace contains File Extension functions, such as ensuring a directory exists, used within the SmartStay application.

7.88 FileExtensions.cs

Go to the documentation of this file.
```
00001
00009
00014 namespace SmartStay.IO.Extensions
00015 {
00019 public static class FileExtensions
00020 {
00025     public static void EnsureDirectoryExists(this string filePath)
00026     {
00027         var directory = Path.GetDirectoryName(filePath);
00028         if (!string.IsNullOrEmpty(directory) && !Directory.Exists(directory))
00029         {
00030             Directory.CreateDirectory(directory);
00031         }
00032     }
00033 }
00034 }
```

7.89 FileHandler.cs File Reference

Data Structures

- class **SmartStay.IO.FileOperations.FileHandler**

  Provides static methods for file operations such as reading from and writing to files.

Namespaces

- namespace SmartStay
- namespace SmartStay.IO
- namespace SmartStay.IO.FileOperations

  Provides file handling operations such as reading from and writing to files.

7.90 FileHandler.cs

Go to the documentation of this file.
```
00001
00013
00017 namespace SmartStay.IO.FileOperations
00018 {
00022 public static class FileHandler
00023 {
00029     public static string ReadFile(string filePath)
00030     {
00031         if (string.IsNullOrWhiteSpace(filePath))
00032             throw new ArgumentException("File path cannot be null or empty.");
00033
00034         if (!File.Exists(filePath))
00035             throw new FileNotFoundException($"File not found: {filePath}");
```

```
00036
00037        return File.ReadAllText(filePath);
00038    }
00039
00045    public static void WriteFile(string filePath, string content)
00046    {
00047        if (string.IsNullOrWhiteSpace(filePath))
00048            throw new ArgumentException("File path cannot be null or empty.");
00049
00050        var directory = Path.GetDirectoryName(filePath);
00051        if (!string.IsNullOrEmpty(directory) && !Directory.Exists(directory))
00052        {
00053            Directory.CreateDirectory(directory);
00054        }
00055
00056        File.WriteAllText(filePath, content);
00057    }
00058 }
00059 }
```

7.91 PathValidator.cs File Reference

Data Structures

- class **SmartStay.IO.FileOperations.PathValidator**

    Provides utility methods for validating file paths and extensions.

Namespaces

- namespace SmartStay
- namespace SmartStay.IO
- namespace SmartStay.IO.FileOperations

    Provides file handling operations such as reading from and writing to files.

7.92 PathValidator.cs

Go to the documentation of this file.

```
00001
00009
00013 namespace SmartStay.IO.FileOperations
00014 {
00018 public static class PathValidator
00019 {
00025    public static bool FileExists(string filePath)
00026    {
00027        return File.Exists(filePath);
00028    }
00029
00036    public static bool IsValidFileType(string filePath, string extension)
00037    {
00038        if (string.IsNullOrWhiteSpace(filePath))
00039            throw new ArgumentException("File path cannot be null or empty.");
00040
00041        return Path.GetExtension(filePath).Equals(extension, StringComparison.OrdinalIgnoreCase);
00042    }
00043 }
00044 }
```

7.93 SmartStay.IO.AssemblyInfo.cs File Reference

7.94 SmartStay.IO.AssemblyInfo.cs

Go to the documentation of this file.
```
00001 //------------------------------------------------------------------------------
00002 // <auto-generated>
00003 //     This code was generated by a tool.
00004 //     Runtime Version:4.0.30319.42000
00005 //
00006 //     Changes to this file may cause incorrect behavior and will be lost if
00007 //     the code is regenerated.
00008 // </auto-generated>
00009 //------------------------------------------------------------------------------
00010
00011 using System;
00012 using System.Reflection;
00013
00014 [assembly: System.Reflection.AssemblyCompanyAttribute("SmartStay.IO")]
00015 [assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]
00016 [assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]
00017 [assembly:
      System.Reflection.AssemblyInformationalVersionAttribute("1.0.0+c366ac03947932e5126b804e73253b4d5f5e0e8d")]
00018 [assembly: System.Reflection.AssemblyProductAttribute("SmartStay.IO")]
00019 [assembly: System.Reflection.AssemblyTitleAttribute("SmartStay.IO")]
00020 [assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]
00021
00022 // Generated by the MSBuild WriteCodeFragment class.
00023
```

7.95 SmartStay.IO.GlobalUsings.g.cs File Reference

7.96 SmartStay.IO.GlobalUsings.g.cs

Go to the documentation of this file.
```
00001 // <auto-generated/>
00002 global using global::System;
00003 global using global::System.Collections.Generic;
00004 global using global::System.IO;
00005 global using global::System.Linq;
00006 global using global::System.Net.Http;
00007 global using global::System.Threading;
00008 global using global::System.Threading.Tasks;
```

7.97 SmartStay.Validation.AssemblyInfo.cs File Reference

7.98 SmartStay.Validation.AssemblyInfo.cs

Go to the documentation of this file.
```
00001 //------------------------------------------------------------------------------
00002 // <auto-generated>
00003 //     This code was generated by a tool.
00004 //     Runtime Version:4.0.30319.42000
00005 //
00006 //     Changes to this file may cause incorrect behavior and will be lost if
00007 //     the code is regenerated.
00008 // </auto-generated>
00009 //------------------------------------------------------------------------------
00010
00011 using System;
00012 using System.Reflection;
00013
00014 [assembly: System.Reflection.AssemblyCompanyAttribute("SmartStay.Validation")]
00015 [assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]
```

```
00016 [assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]
00017 [assembly:
     System.Reflection.AssemblyInformationalVersionAttribute("1.0.0+c366ac03947932e5126b804e73253b4d5f5e0e8d")]
00018 [assembly: System.Reflection.AssemblyProductAttribute("SmartStay.Validation")]
00019 [assembly: System.Reflection.AssemblyTitleAttribute("SmartStay.Validation")]
00020 [assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]
00021
00022 // Generated by the MSBuild WriteCodeFragment class.
00023
```

7.99 SmartStay.Validation.GlobalUsings.g.cs File Reference

7.100 SmartStay.Validation.GlobalUsings.g.cs

Go to the documentation of this file.
```
00001 // <auto-generated/>
00002 global using global::System;
00003 global using global::System.Collections.Generic;
00004 global using global::System.IO;
00005 global using global::System.Linq;
00006 global using global::System.Net.Http;
00007 global using global::System.Threading;
00008 global using global::System.Threading.Tasks;
```

7.101 ValidationMessages.Designer.cs File Reference

Data Structures

- class **SmartStay.Validation.Resources.ValidationMessages**

  A strongly-typed resource class, for looking up localized strings, etc.

Namespaces

- namespace SmartStay
- namespace SmartStay.Validation

  The SmartStay.Validation namespace contains classes and methods for validating data and enforcing business rules within the SmartStay application. These validations help ensure data integrity and compliance with application requirements.
- namespace SmartStay.Validation.Resources

7.102 ValidationMessages.Designer.cs

Go to the documentation of this file.
```
00001 //------------------------------------------------------------------------------
00002 // <auto-generated>
00003 //     This code was generated by a tool.
00004 //     Runtime Version:4.0.30319.42000
00005 //
00006 //     Changes to this file may cause incorrect behavior and will be lost if
00007 //     the code is regenerated.
00008 // </auto-generated>
00009 //------------------------------------------------------------------------------
00010
00011 namespace SmartStay.Validation.Resources {
00012     using System;
00013
00014
00018     // This class was auto-generated by the StronglyTypedResourceBuilder
```

```
00019      // class via a tool like ResGen or Visual Studio.
00020      // To add or remove a member, edit your .ResX file then rerun ResGen
00021      // with the /str option, or rebuild your VS project.
00022
     [global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyTypedResourceBuilder",
     "17.0.0.0")]
00023      [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
00024      [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
00025      internal class ValidationMessages {
00026
00027          private static global::System.Resources.ResourceManager resourceMan;
00028
00029          private static global::System.Globalization.CultureInfo resourceCulture;
00030
00031          [global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
     "CA1811:AvoidUncalledPrivateCode")]
00032          internal ValidationMessages() {
00033          }
00034
00038
     [global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
00039          internal static global::System.Resources.ResourceManager ResourceManager {
00040              get {
00041                  if (object.ReferenceEquals(resourceMan, null)) {
00042                      global::System.Resources.ResourceManager temp = new
     global::System.Resources.ResourceManager("SmartStay.Validation.Resources.ValidationMessages",
     typeof(ValidationMessages).Assembly);
00043                      resourceMan = temp;
00044                  }
00045                  return resourceMan;
00046              }
00047          }
00048
00053
     [global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
00054          internal static global::System.Globalization.CultureInfo Culture {
00055              get {
00056                  return resourceCulture;
00057              }
00058              set {
00059                  resourceCulture = value;
00060              }
00061          }
00062
00066          internal static string InvalidAccommodationName {
00067              get {
00068                  return ResourceManager.GetString("InvalidAccommodationName", resourceCulture);
00069              }
00070          }
00071
00075          internal static string InvalidAccommodationType {
00076              get {
00077                  return ResourceManager.GetString("InvalidAccommodationType", resourceCulture);
00078              }
00079          }
00080
00084          internal static string InvalidAddress {
00085              get {
00086                  return ResourceManager.GetString("InvalidAddress", resourceCulture);
00087              }
00088          }
00089
00093          internal static string InvalidAvailabilityStatus {
00094              get {
00095                  return ResourceManager.GetString("InvalidAvailabilityStatus", resourceCulture);
00096              }
00097          }
00098
00102          internal static string InvalidDate {
00103              get {
00104                  return ResourceManager.GetString("InvalidDate", resourceCulture);
00105              }
00106          }
00107
00111          internal static string InvalidDateRange {
00112              get {
00113                  return ResourceManager.GetString("InvalidDateRange", resourceCulture);
00114              }
00115          }
00116
00120          internal static string InvalidEmail {
00121              get {
00122                  return ResourceManager.GetString("InvalidEmail", resourceCulture);
00123              }
00124          }
00125
00129          internal static string InvalidId {
```

```
00130                get {
00131                    return ResourceManager.GetString("InvalidId", resourceCulture);
00132                }
00133            }
00134
00138        internal static string InvalidName {
00139            get {
00140                return ResourceManager.GetString("InvalidName", resourceCulture);
00141            }
00142        }
00143
00147        internal static string InvalidPaymentMethod {
00148            get {
00149                return ResourceManager.GetString("InvalidPaymentMethod", resourceCulture);
00150            }
00151        }
00152
00156        internal static string InvalidPaymentStatus {
00157            get {
00158                return ResourceManager.GetString("InvalidPaymentStatus", resourceCulture);
00159            }
00160        }
00161
00165        internal static string InvalidPaymentValue {
00166            get {
00167                return ResourceManager.GetString("InvalidPaymentValue", resourceCulture);
00168            }
00169        }
00170
00174        internal static string InvalidPhoneNumber {
00175            get {
00176                return ResourceManager.GetString("InvalidPhoneNumber", resourceCulture);
00177            }
00178        }
00179
00183        internal static string InvalidPrice {
00184            get {
00185                return ResourceManager.GetString("InvalidPrice", resourceCulture);
00186            }
00187        }
00188
00192        internal static string InvalidReservationStatus {
00193            get {
00194                return ResourceManager.GetString("InvalidReservationStatus", resourceCulture);
00195            }
00196        }
00197
00201        internal static string InvalidRoomType {
00202            get {
00203                return ResourceManager.GetString("InvalidRoomType", resourceCulture);
00204            }
00205        }
00206
00210        internal static string InvalidTotalCost {
00211            get {
00212                return ResourceManager.GetString("InvalidTotalCost", resourceCulture);
00213            }
00214        }
00215    }
00216 }
```

7.103 ValidationErrorCodes.cs File Reference

Namespaces

- namespace SmartStay
- namespace SmartStay.Validation

    The SmartStay.Validation namespace contains classes and methods for validating data and enforcing business rules within the SmartStay application. These validations help ensure data integrity and compliance with application requirements.

Enumerations

- enum SmartStay.Validation.ValidationErrorCode {
  SmartStay.Validation.InvalidName = 1001 , SmartStay.Validation.InvalidEmail = 1002 , SmartStay.Validation.InvalidPhoneNumbe

= 1003 , SmartStay.Validation.InvalidAddress = 1004 ,
SmartStay.Validation.InvalidPaymentMethod = 1005 , SmartStay.Validation.InvalidAccommodationType =
1006 , SmartStay.Validation.InvalidId = 1007 , SmartStay.Validation.InvalidDateRange = 1008 ,
SmartStay.Validation.InvalidDate = 1009 , SmartStay.Validation.InvalidTotalCost = 1010 , SmartStay.Validation.InvalidPaymentVa
= 1011 , SmartStay.Validation.InvalidReservationStatus = 1012 ,
SmartStay.Validation.InvalidAccommodationName = 1013 , SmartStay.Validation.InvalidPrice = 1014 ,
SmartStay.Validation.InvalidPaymentStatus = 1015 , SmartStay.Validation.InvalidAvailabilityStatus = 1016

,
SmartStay.Validation.InvalidRoomType = 1017 }

> *Defines error codes for validation failures within the SmartStay application.*

7.104 ValidationErrorCodes.cs

Go to the documentation of this file.

```
00001
00010
00016 namespace SmartStay.Validation
00017 {
00021 public enum ValidationErrorCode
00022 {
00026     InvalidName = 1001,
00027
00031     InvalidEmail = 1002,
00032
00036     InvalidPhoneNumber = 1003,
00037
00041     InvalidAddress = 1004,
00042
00046     InvalidPaymentMethod = 1005,
00047
00051     InvalidAccommodationType = 1006,
00052
00056     InvalidId = 1007,
00057
00062     InvalidDateRange = 1008,
00063
00068     InvalidDate = 1009,
00069
00073     InvalidTotalCost = 1010,
00074
00079     InvalidPaymentValue = 1011,
00080
00085     InvalidReservationStatus = 1012,
00086
00090     InvalidAccommodationName = 1013,
00091
00095     InvalidPrice = 1014,
00096
00100     InvalidPaymentStatus = 1015,
00101
00105     InvalidAvailabilityStatus = 1016,
00106
00110     InvalidRoomType = 1017
00111 }
00112 }
```

7.105 ValidationErrorMessages.cs File Reference

Data Structures

- class **SmartStay.Validation.ValidationErrorMessages**

    Provides a mechanism to retrieve localized validation error messages based on the given ValidationErrorCode. Messages are retrieved from resource files depending on the current culture of the application.

Namespaces

- namespace SmartStay
- namespace SmartStay.Validation

    The SmartStay.Validation namespace contains classes and methods for validating data and enforcing business rules within the SmartStay application. These validations help ensure data integrity and compliance with application requirements.

7.106 ValidationErrorMessages.cs

Go to the documentation of this file.
```
00001
00011 using System.Globalization;
00012 using System.Resources;
00013
00019 namespace SmartStay.Validation
00020 {
00026 public static class ValidationErrorMessages
00027 {
00031     private static readonly ResourceManager _resourceManager = new ResourceManager(
00032         "SmartStay.Validation.Resources.ValidationMessages",
     typeof(ValidationErrorMessages).Assembly);
00033
00040     public static string GetErrorMessage(ValidationErrorCode errorCode)
00041     {
00042         // Get the string based on the current culture
00043         return _resourceManager.GetString(errorCode.ToString(), CultureInfo.CurrentCulture) ??
00044             "Unknown validation error.";
00045     }
00046 }
00047 }
```

7.107 ValidationException.cs File Reference

Data Structures

- class SmartStay.Validation.ValidationException

    Represents an exception that is thrown when a validation error occurs. This exception contains an error code that corresponds to a specific validation failure. The error message is retrieved from the resource files based on the error code and the current culture.

Namespaces

- namespace SmartStay
- namespace SmartStay.Validation

    The SmartStay.Validation namespace contains classes and methods for validating data and enforcing business rules within the SmartStay application. These validations help ensure data integrity and compliance with application requirements.

7.108 ValidationException.cs

Go to the documentation of this file.
```
00001
00011
00017 namespace SmartStay.Validation
00018 {
00024 public class ValidationException : Exception
00025 {
00029     public ValidationErrorCode ErrorCode { get; }
00030
00037     public ValidationException(ValidationErrorCode errorCode) :
     base(ValidationErrorMessages.GetErrorMessage(errorCode))
00038     {
00039         ErrorCode = errorCode;
00040     }
00041 }
00042 }
```

7.109 AccommodationValidator.cs File Reference

Data Structures

- class **SmartStay.Validation.Validators.AccommodationValidator**

    Defines the AccommodationValidator class, which provides functionality for validating accommodation types in the SmartStay *application.*

Namespaces

- namespace SmartStay
- namespace SmartStay.Validation

    The `SmartStay.Validation` *namespace contains classes and methods for validating data and enforcing business rules within the* SmartStay *application. These validations help ensure data integrity and compliance with application requirements.*

- namespace SmartStay.Validation.Validators

    The `SmartStay.Validation.Validators` *namespace contains classes and methods for validating various types of input data in the* SmartStay *application. These validations enforce data integrity and compliance with application-specific requirements.*

7.110 AccommodationValidator.cs

Go to the documentation of this file.
```
00001
00015 using SmartStay.Common.Enums;
00016
00017 namespace SmartStay.Validation.Validators
00018 {
00023 public static class AccommodationValidator
00024 {
00031     public static AccommodationType ValidateAccommodationType(AccommodationType accommodationType)
00032     {
00033         if (!IsValidAccommodationType(accommodationType))
00034         {
00035             throw new ValidationException(ValidationErrorCode.InvalidAccommodationType);
00036         }
00037         return accommodationType;
00038     }
00039
00046     public static bool IsValidAccommodationType(AccommodationType accommodationType)
00047     {
00048         return Enum.IsDefined(typeof(AccommodationType), accommodationType);
00049     }
00050
00057     public static int ValidateAccommodationId(int id)
00058     {
00059         if (!IsValidAccommodationId(id))
00060         {
00061             throw new ValidationException(ValidationErrorCode.InvalidId);
00062         }
00063         return id;
00064     }
00065
00071     public static bool IsValidAccommodationId(int id) => id > 0;
00072 }
00073 }
```

7.111 AddressValidator.cs File Reference

Data Structures

- class **SmartStay.Validation.Validators.AddressValidator**

    Defines the AddressValidator class, which provides functionality for validating addresses used in the SmartStay *application.*

Namespaces

- namespace SmartStay
- namespace SmartStay.Validation

    The *SmartStay.Validation namespace contains classes and methods for validating data and enforcing busi-ness rules within the SmartStay application. These validations help ensure data integrity and compliance with appli-cation requirements.*

- namespace SmartStay.Validation.Validators

    The *SmartStay.Validation.Validators namespace contains classes and methods for validating vari-ous types of input data in the SmartStay application. These validations enforce data integrity and compliance with application-specific requirements.*

7.112 AddressValidator.cs

Go to the documentation of this file.
```
00001
00010
00016 namespace SmartStay.Validation.Validators
00017 {
00022     public static class AddressValidator
00023     {
00030         public static string ValidateAddress(string address)
00031         {
00032             if (!IsValidAddress(address))
00033             {
00034                 throw new ValidationException(ValidationErrorCode.InvalidAddress);
00035             }
00036             return address;
00037         }
00038
00044         public static bool IsValidAddress(string address) => !string.IsNullOrWhiteSpace(address);
00045 }
00046 }
```

7.113 ClientValidator.cs File Reference

Data Structures

- class **SmartStay.Validation.Validators.ClientValidator**

    Defines the ClientValidator class, which provides functionality for validating client-related data in the SmartStay appli-cation.

Namespaces

- namespace SmartStay
- namespace SmartStay.Validation

    The *SmartStay.Validation namespace contains classes and methods for validating data and enforcing busi-ness rules within the SmartStay application. These validations help ensure data integrity and compliance with appli-cation requirements.*

- namespace SmartStay.Validation.Validators

    The *SmartStay.Validation.Validators namespace contains classes and methods for validating vari-ous types of input data in the SmartStay application. These validations enforce data integrity and compliance with application-specific requirements.*

7.114 ClientValidator.cs

Go to the documentation of this file.
```
00001
00010
00016 namespace SmartStay.Validation.Validators
00017 {
00022 public static class ClientValidator
00023 {
00030     public static int ValidateClientId(int id)
00031     {
00032         if (!IsValidClientId(id))
00033         {
00034             throw new ValidationException(ValidationErrorCode.InvalidId);
00035         }
00036         return id;
00037     }
00038
00044     public static bool IsValidClientId(int id) => id > 0;
00045 }
00046 }
```

7.115 DateValidator.cs File Reference

Data Structures

- class **SmartStay.Validation.Validators.DateValidator**

  Defines the DateValidator class, which provides functionality for validating dates related to reservations, ensuring they adhere to application-specific rules.

Namespaces

- namespace SmartStay
- namespace SmartStay.Validation

  The `SmartStay.Validation` namespace contains classes and methods for validating data and enforcing business rules within the SmartStay application. These validations help ensure data integrity and compliance with application requirements.

- namespace SmartStay.Validation.Validators

  The `SmartStay.Validation.Validators` namespace contains classes and methods for validating various types of input data in the SmartStay application. These validations enforce data integrity and compliance with application-specific requirements.

7.116 DateValidator.cs

Go to the documentation of this file.
```
00001
00011
00017 namespace SmartStay.Validation.Validators
00018 {
00023 public static class DateValidator
00024 {
00031     public static DateTime ValidateCheckInDate(DateTime checkInDate)
00032     {
00033         if (!IsValidFutureDate(checkInDate))
00034         {
00035             throw new ValidationException(ValidationErrorCode.InvalidDate);
00036         }
00037         return checkInDate;
00038     }
00039
00047     public static DateTime ValidateCheckOutDate(DateTime checkOutDate, DateTime checkInDate)
00048     {
```

```
00049          if (!IsValidDateRange(checkInDate, checkOutDate))
00050          {
00051              throw new ValidationException(ValidationErrorCode.InvalidDateRange);
00052          }
00053          return checkOutDate;
00054      }
00055
00061      public static bool IsValidFutureDate(DateTime date)
00062      {
00063          return date >= DateTime.Today;
00064      }
00065
00072      public static bool IsValidDateRange(DateTime checkInDate, DateTime checkOutDate)
00073      {
00074          return checkInDate < checkOutDate;
00075      }
00076 }
00077 }
```

7.117 EmailValidator.cs File Reference

Data Structures

- class **SmartStay.Validation.Validators.EmailValidator**

  Defines the EmailValidator class, which provides functionality for validating email addresses within the SmartStay application.

Namespaces

- namespace SmartStay
- namespace SmartStay.Validation

  The `SmartStay.Validation` namespace contains classes and methods for validating data and enforcing business rules within the SmartStay application. These validations help ensure data integrity and compliance with application requirements.

- namespace SmartStay.Validation.Validators

  The `SmartStay.Validation.Validators` namespace contains classes and methods for validating various types of input data in the SmartStay application. These validations enforce data integrity and compliance with application-specific requirements.

7.118 EmailValidator.cs

Go to the documentation of this file.
```
00001
00010 using System.Text.RegularExpressions;
00011
00017 namespace SmartStay.Validation.Validators
00018 {
00023 public static class EmailValidator
00024 {
00025     // TODO: add a way to check if an email is already being used
00026
00030     static readonly string EmailPattern = @"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$";
00031
00038     public static string ValidateEmail(string email)
00039     {
00040         if (!IsValidEmail(email))
00041         {
00042             throw new ValidationException(ValidationErrorCode.InvalidEmail);
00043         }
00044         return email;
00045     }
00046
00052     public static bool IsValidEmail(string email) => !string.IsNullOrWhiteSpace(email) &&
     Regex.IsMatch(email,
00053
     EmailPattern);
00054 }
00055 }
```

7.119 NameValidator.cs File Reference

Data Structures

- class **SmartStay.Validation.Validators.NameValidator**

    Defines the NameValidator class, which provides functionality for validating various types of names within the SmartStay application.

Namespaces

- namespace SmartStay
- namespace SmartStay.Validation

    The `SmartStay.Validation` *namespace contains classes and methods for validating data and enforcing business rules within the SmartStay application. These validations help ensure data integrity and compliance with application requirements.*

- namespace SmartStay.Validation.Validators

    The `SmartStay.Validation.Validators` *namespace contains classes and methods for validating various types of input data in the SmartStay application. These validations enforce data integrity and compliance with application-specific requirements.*

7.120 NameValidator.cs

Go to the documentation of this file.
```
00001
00011
00017 namespace SmartStay.Validation.Validators
00018 {
00023   public static class NameValidator
00024 {
00031     public static string ValidateName(string name)
00032     {
00033         if (!IsValidName(name))
00034         {
00035             throw new ValidationException(ValidationErrorCode.InvalidName);
00036         }
00037         return name;
00038     }
00039
00045     public static bool IsValidName(string name) => !string.IsNullOrWhiteSpace(name) && name.Length <=
      50;
00046
00053     public static string ValidateAccommodationName(string name)
00054     {
00055         if (!IsValidAccommodationName(name))
00056         {
00057             throw new ValidationException(ValidationErrorCode.InvalidAccommodationName);
00058         }
00059         return name;
00060     }
00061
00067     public static bool IsValidAccommodationName(string name) => !string.IsNullOrWhiteSpace(name) &&
      name.Length <= 100;
00068 }
00069 }
```

7.121 OwnerValidator.cs File Reference

Data Structures

- class **SmartStay.Validation.Validators.OwnerValidator**

    Defines the OwnerValidator class, which provides functionality for validating owner-related data in the SmartStay application.

Namespaces

- namespace SmartStay
- namespace SmartStay.Validation

  The SmartStay.Validation namespace contains classes and methods for validating data and enforcing business rules within the SmartStay application. These validations help ensure data integrity and compliance with application requirements.

- namespace SmartStay.Validation.Validators

  The SmartStay.Validation.Validators namespace contains classes and methods for validating various types of input data in the SmartStay application. These validations enforce data integrity and compliance with application-specific requirements.

7.122   OwnerValidator.cs

Go to the documentation of this file.
```
00001
00010
00016 namespace SmartStay.Validation.Validators
00017 {
00022 public static class OwnerValidator
00023 {
00030     public static int ValidateOwnerId(int id)
00031     {
00032         if (!IsValidOwnerId(id))
00033         {
00034             throw new ValidationException(ValidationErrorCode.InvalidId);
00035         }
00036         return id;
00037     }
00038
00044     public static bool IsValidOwnerId(int id) => id > 0;
00045
00052     public static string ValidateOwnerName(string name)
00053     {
00054         if (string.IsNullOrWhiteSpace(name) || name.Length < 2)
00055         {
00056             throw new ValidationException(ValidationErrorCode.InvalidName);
00057         }
00058         return name;
00059     }
00060
00067     public static string ValidateOwnerEmail(string email)
00068     {
00069         if (string.IsNullOrWhiteSpace(email) || !email.Contains('@') || !email.Contains('.'))
00070         {
00071             throw new ValidationException(ValidationErrorCode.InvalidEmail);
00072         }
00073         return email;
00074     }
00075
00082     public static string ValidateOwnerPhoneNumber(string phoneNumber)
00083     {
00084         if (string.IsNullOrWhiteSpace(phoneNumber) || phoneNumber.Length < 10)
00085         {
00086             throw new ValidationException(ValidationErrorCode.InvalidPhoneNumber);
00087         }
00088         return phoneNumber;
00089     }
00090 }
00091 }
```

7.123   PaymentValidator.cs File Reference

Data Structures

- class **SmartStay.Validation.Validators.PaymentValidator**

  The SmartStay.Validation.Validators namespace contains classes and methods for validating various types of input data in the SmartStay application. These validations enforce data integrity and compliance with application-specific requirements.

Namespaces

- namespace SmartStay

- namespace SmartStay.Validation

    The *SmartStay.Validation* namespace contains classes and methods for validating data and enforcing business rules within the *SmartStay* application. These validations help ensure data integrity and compliance with application requirements.

- namespace SmartStay.Validation.Validators

    The *SmartStay.Validation.Validators* namespace contains classes and methods for validating various types of input data in the *SmartStay* application. These validations enforce data integrity and compliance with application-specific requirements.

7.124 PaymentValidator.cs

Go to the documentation of this file.
```
00001
00011 using SmartStay.Common.Enums;
00012
00018 namespace SmartStay.Validation.Validators
00019 {
00025 public static class PaymentValidator
00026 {
00033     public static decimal ValidatePrice(decimal price)
00034     {
00035         if (!IsValidPrice(price))
00036         {
00037             throw new ValidationException(ValidationErrorCode.InvalidPrice);
00038         }
00039         return price;
00040     }
00041
00047     public static bool IsValidPrice(decimal price) => price > 0;
00048
00055     public static decimal ValidateTotalCost(decimal totalCost)
00056     {
00057         if (!IsValidTotalCost(totalCost))
00058         {
00059             throw new ValidationException(ValidationErrorCode.InvalidTotalCost);
00060         }
00061         return totalCost;
00062     }
00063
00069     public static bool IsValidTotalCost(decimal totalCost) => totalCost > 0;
00070
00077     public static decimal ValidatePaymentAmount(decimal amount)
00078     {
00079         if (!IsValidPaymentAmount(amount))
00080         {
00081             throw new ValidationException(ValidationErrorCode.InvalidPaymentValue);
00082         }
00083         return amount;
00084     }
00085
00091     public static bool IsValidPaymentAmount(decimal amount) => amount > 0;
00092
00099     public static PaymentStatus ValidatePaymentStatus(PaymentStatus status)
00100     {
00101         if (!IsValidPaymentStatus(status))
00102         {
00103             throw new ValidationException(ValidationErrorCode.InvalidPaymentStatus);
00104         }
00105         return status;
00106     }
00107
00113     public static bool IsValidPaymentStatus(PaymentStatus paymentStatus) =>
      Enum.IsDefined(typeof(PaymentStatus),
00114
      paymentStatus);
00115
00122     public static PaymentMethod ValidatePaymentMethod(PaymentMethod paymentMethod)
00123     {
00124         if (!IsValidPaymentMethod(paymentMethod))
00125         {
00126             throw new ValidationException(ValidationErrorCode.InvalidPaymentMethod);
00127         }
00128         return paymentMethod;
```

```
00129     }
00130
00136     public static bool IsValidPaymentMethod(PaymentMethod paymentMethod) =>
     Enum.IsDefined(typeof(PaymentMethod),
00137
     paymentMethod);
00138
00145     public static decimal ValidatePayment(decimal paymentValue)
00146     {
00147         if (paymentValue < 0)
00148         {
00149             throw new ValidationException(ValidationErrorCode.InvalidPaymentValue);
00150         }
00151         return paymentValue;
00152     }
00153 }
00154 }
```

7.125 PhoneNumberValidator.cs File Reference

Data Structures

- class **SmartStay.Validation.Validators.PhoneNumberValidator**

    The SmartStay.Validation.Validators namespace contains classes and methods for validating various types of input data in the SmartStay application. These validations enforce data integrity and compliance with application-specific requirements.

Namespaces

- namespace SmartStay
- namespace SmartStay.Validation

    The SmartStay.Validation namespace contains classes and methods for validating data and enforcing business rules within the SmartStay application. These validations help ensure data integrity and compliance with application requirements.

- namespace SmartStay.Validation.Validators

    The SmartStay.Validation.Validators namespace contains classes and methods for validating various types of input data in the SmartStay application. These validations enforce data integrity and compliance with application-specific requirements.

7.126 PhoneNumberValidator.cs

Go to the documentation of this file.
```
00001
00011 using System.Text.RegularExpressions;
00012
00018 namespace SmartStay.Validation.Validators
00019 {
00025 public static class PhoneNumberValidator
00026 {
00031     private static readonly string PhoneNumberPattern = @"^\+(\d{1,3})\d{7,15}$";
00032
00039     public static string ValidatePhoneNumber(string phoneNumber)
00040     {
00041         if (!IsValidPhoneNumber(phoneNumber))
00042         {
00043             throw new ValidationException(ValidationErrorCode.InvalidPhoneNumber);
00044         }
00045         return phoneNumber;
00046     }
00047
00054     public static bool IsValidPhoneNumber(string phoneNumber) =>
00055         !string.IsNullOrWhiteSpace(phoneNumber) && Regex.IsMatch(phoneNumber, PhoneNumberPattern);
00056 }
00057 }
```

7.127 ReservationValidator.cs File Reference

Data Structures

- class **SmartStay.Validation.Validators.ReservationValidator**

    Provides validation methods for reservation-related data.

Namespaces

- namespace SmartStay
- namespace SmartStay.Validation

    The SmartStay.Validation namespace contains classes and methods for validating data and enforcing business rules within the SmartStay application. These validations help ensure data integrity and compliance with application requirements.

- namespace SmartStay.Validation.Validators

    The SmartStay.Validation.Validators namespace contains classes and methods for validating various types of input data in the SmartStay application. These validations enforce data integrity and compliance with application-specific requirements.

7.128 ReservationValidator.cs

[Go to the documentation of this file.](#)
```
00001
00010 using SmartStay.Common.Enums;
00011
00017 namespace SmartStay.Validation.Validators
00018 {
00022 public static class ReservationValidator
00023 {
00032     public static ReservationStatus ValidateReservationStatus(ReservationStatus status)
00033     {
00034         if (!IsValidReservationStatus(status))
00035         {
00036             throw new ValidationException(ValidationErrorCode.InvalidReservationStatus);
00037         }
00038         return status;
00039     }
00040
00050     public static bool IsValidReservationStatus(ReservationStatus status)
00051     {
00052         return Enum.IsDefined(typeof(ReservationStatus), status);
00053     }
00054
00061     public static int ValidateReservationId(int id)
00062     {
00063         if (!IsValidReservationId(id))
00064         {
00065             throw new ValidationException(ValidationErrorCode.InvalidId);
00066         }
00067         return id;
00068     }
00069
00075     public static bool IsValidReservationId(int id) => id > 0;
00076 }
00077 }
```

7.129 RoomValidator.cs File Reference

Data Structures

- class **SmartStay.Validation.Validators.RoomValidator**

    The RoomValidator class provides methods for validating room-related data within the SmartStay application. It ensures integrity and compliance with business rules.

Namespaces

- namespace SmartStay
- namespace SmartStay.Validation

  The *SmartStay.Validation* namespace contains classes and methods for validating data and enforcing business rules within the *SmartStay* application. These validations help ensure data integrity and compliance with application requirements.

- namespace SmartStay.Validation.Validators

  The *SmartStay.Validation.Validators* namespace contains classes and methods for validating various types of input data in the *SmartStay* application. These validations enforce data integrity and compliance with application-specific requirements.

7.130 RoomValidator.cs

Go to the documentation of this file.
```
00001
00011 using SmartStay.Common.Enums;
00012
00018 namespace SmartStay.Validation.Validators
00019 {
00024 public static class RoomValidator
00025 {
00032     public static RoomType ValidateRoomType(RoomType roomType)
00033     {
00034         if (!IsValidRoomType(roomType))
00035         {
00036             throw new ValidationException(ValidationErrorCode.InvalidRoomType);
00037         }
00038         return roomType;
00039     }
00040
00046     public static bool IsValidRoomType(RoomType roomType) => Enum.IsDefined(typeof(RoomType),
    roomType);
00047
00054     public static bool ValidateAvailability(bool isAvailable)
00055     {
00056         if (!IsValidAvailability(isAvailable))
00057         {
00058             throw new ValidationException(ValidationErrorCode.InvalidAvailabilityStatus);
00059         }
00060         return isAvailable;
00061     }
00062
00068     public static bool IsValidAvailability(bool isAvailable) => isAvailable || !isAvailable;
00069
00076     public static int ValidateRoomId(int id)
00077     {
00078         if (!IsValidRoomId(id))
00079         {
00080             throw new ValidationException(ValidationErrorCode.InvalidId);
00081         }
00082         return id;
00083     }
00084
00090     public static bool IsValidRoomId(int id) => id > 0;
00091 }
00092 }
```

Index