Smart Stay Backend

Generated by Doxygen 1.10.0

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

roomId	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.

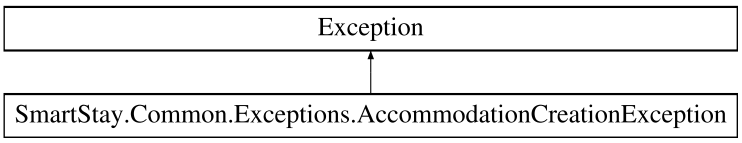
The documentation for this class was generated from the following file:

- 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:



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.

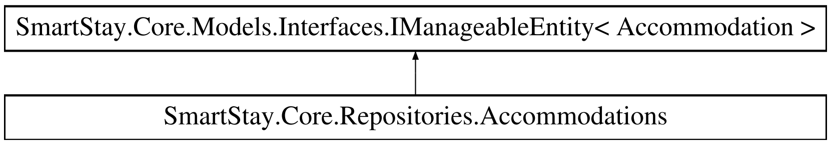
The documentation for this class was generated from the following file:

- 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 291 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 279 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.

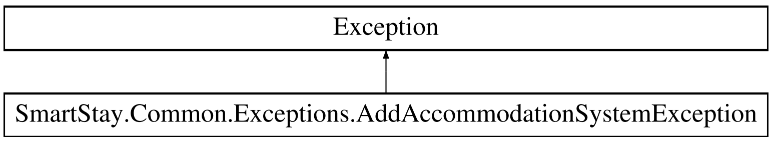
The documentation for this class was generated from the following file:

- 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:



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

clientId	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

ownerId	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.

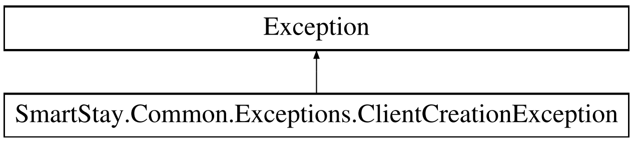
The documentation for this class was generated from the following file:

- 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:



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.

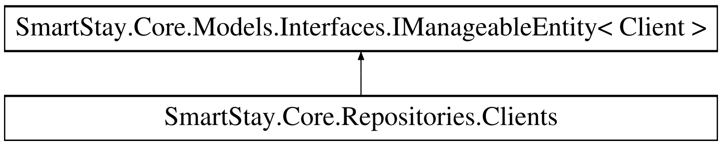
The documentation for this class was generated from the following file:

- 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 282 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 270 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.

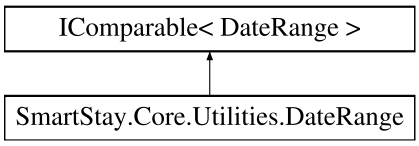
The documentation for this class was generated from the following file:

- 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.

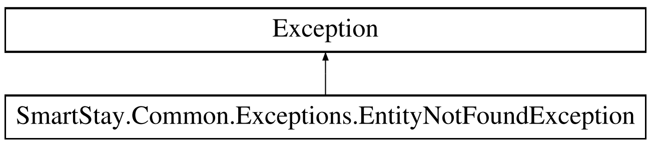
The documentation for this class was generated from the following file:

- 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.

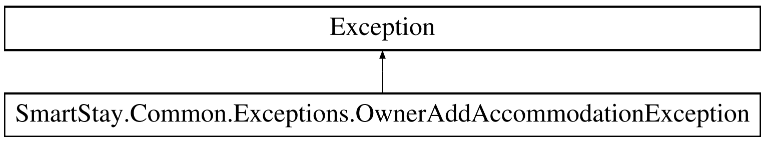
The documentation for this class was generated from the following file:

- 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.

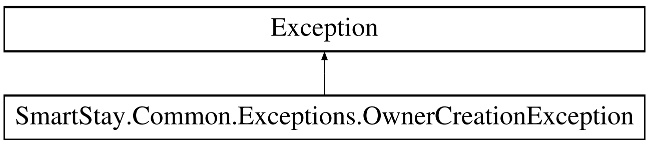
The documentation for this class was generated from the following file:

- 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:



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.

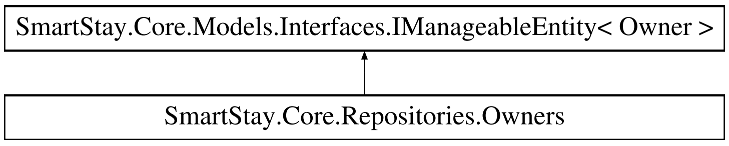
The documentation for this class was generated from the following file:

- 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:


SmartStay.Core.Models.Interfaces.IManageableEntity< Owner >
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 290 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 278 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.

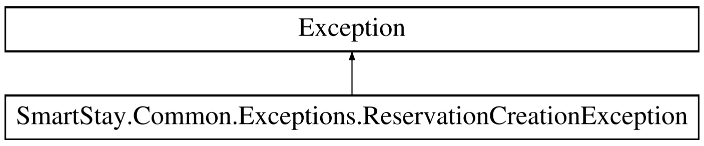
The documentation for this class was generated from the following file:

- 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:



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.

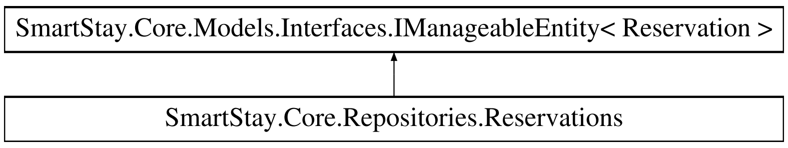
The documentation for this class was generated from the following file:

- 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 330 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 281 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 305 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 292 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 315 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.

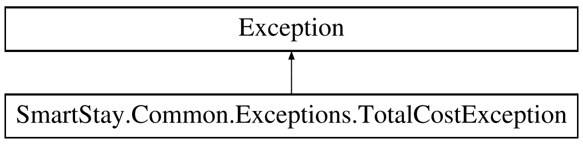
The documentation for this class was generated from the following file:

- 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.

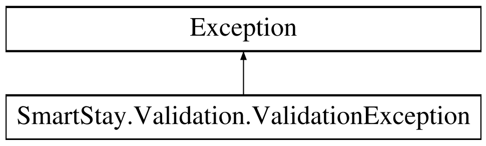
The documentation for this class was generated from the following file:

- 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.

```
namespace SmartStay.Common.Enums
{
public enum AccommodationType
{
    None,

    Hotel,

    House,

    Apartment,

    Villa,

    BedAndBreakfast,
    Hostel,

    Resort,

    Cottage,

    Cabin,

    Guesthouse,

    Chalet,

    Lodge
}
}
```

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.

```


namespace SmartStay.Common.Enums
{
public enum CancellationResult
{
    Success,

    ReservationNotFound,

    AccommodationNotFound,

    RoomNotFound,

    Error
}
}
```

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.

```
namespace SmartStay.Common.Enums
{
public enum PaymentMethod
{
    Unchanged,

    None,

    PayPal,

    MultiBanco,

    BankTransfer
}
}
```

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.

```


namespace SmartStay.Common.Enums
{
public enum PaymentResult
{
    Success,

    InvalidAmount,

    AlreadyFullyPaid,

    AmountExceedsTotal,

    InvalidPaymentMethod,

    Error
}
}
```

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.

```


namespace SmartStay.Common.Enums
{
public enum PaymentStatus
{
    Unpaid,

    Pending,

    Completed,

    PartiallyPaid,

    Rejected,

    Refunded,

    Cancelled
}
}
```

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.

```
namespace SmartStay.Common.Enums
{
public enum RemoveAccommodationResult
{
    Success,

    AccommodationNotFound,

    OwnerNotFound,

    AccommodationRemovalFailed,

    AccommodationDisassociationFailed,

    Error
}
}
```

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.

```
namespace SmartStay.Common.Enums
{
public enum ReservationStatus
{
    Pending,

    CheckedIn,

    CheckedOut,

    Cancelled,

    NoShow,

    Confirmed,

    Declined
}
}
```

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.

```
namespace SmartStay.Common.Enums
{
public enum RoomType
{
    None,

    Single,

    Double,

    Twin,

    Suite,

    Family,

    Studio,

    Deluxe,

    Penthouse,

    Dormitory,

    Accessible,

    PresidentialSuite
}
}
```

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.

```


namespace SmartStay.Common.Enums
{
public enum UpdateAccommodationResult
{
    Success,

    AccommodationNotFound,

    InvalidType,

    InvalidName,

    InvalidAddress,

    Error
}
}
```

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.

```

namespace SmartStay.Common.Enums
{
public enum UpdateClientResult
{
    Success,

    ClientNotFound,

    InvalidFirstName,

    InvalidLastName,

    InvalidEmail,

    InvalidPhoneNumber,

    InvalidAddress,

    InvalidPaymentMethod,

    Error
}
}
```

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.

```
namespace SmartStay.Common.Enums
{
public enum UpdateOwnerResult
{
    Success,

    OwnerNotFound,

    InvalidFirstName,

    InvalidLastName,

    InvalidEmail,

    InvalidPhoneNumber,

    InvalidAddress
}

}
```

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.InvalidD
  , 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.

```
namespace SmartStay.Common.Enums
{
public enum UpdateReservationResult
{
    Success,

    ReservationNotFound,

    AccommodationNotFound,

    RoomNotFound,

    RoomIsNull,

    DatesUnavailable,

    InvalidDates,

    Error
}
}
```

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.

```
namespace SmartStay.Common.Exceptions
{
public class AccommodationCreationException : Exception
{
    public AccommodationCreationException(string message) : base(message)
    {
    }

    public AccommodationCreationException(string message, Exception innerException) : base(message,
innerException)
    {
    }

    public override string Message => base.Message;

    public override string ToString()
    {
        return $"{base.ToString()}";
    }
}
}
```

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.

```
namespace SmartStay.Common.Exceptions
{
public class AddAccommodationSystemException : Exception
{
    public AddAccommodationSystemException(string message) : base(message)
    {
    }

    public AddAccommodationSystemException(string message, Exception innerException) : base(message,
innerException)
    {
    }

    public override string Message => base.Message;

    public override string ToString()
    {
        return $"{base.ToString()}";
    }
}
}
```

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.

```
namespace SmartStay.Common.Exceptions
{
public class ClientCreationException : Exception
{
    public ClientCreationException(string message) : base(message)
    {
    }

    public ClientCreationException(string message, Exception innerException) : base(message,
innerException)
    {
    }

    public override string Message => base.Message;

    public override string ToString()
    {
        return $"{base.ToString()}";
    }
}
}
```

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.

```
namespace SmartStay.Common.Exceptions
{
public class EntityNotFoundException : Exception
{
    public string EntityType { get; }

    public int EntityId { get; }

    public EntityNotFoundException(string entityType, int entityId)
        : base($"{entityType} with ID {entityId} was not found.")
    {
        EntityType = entityType;
        EntityId = entityId;
    }

    public EntityNotFoundException(string entityType, int entityId, string message)
        : base($"{entityType} with ID {entityId} was not found. {message}")
    {
        EntityType = entityType;
        EntityId = entityId;
    }

    public EntityNotFoundException(string entityType, int entityId, string message, Exception
innerException)
        : base($"{entityType} with ID {entityId} was not found. {message}", innerException)
    {
        EntityType = entityType;
        EntityId = entityId;
    }

    public override string ToString()
    {
        return $"{base.ToString()}";
    }
}
}
```

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.

```


namespace SmartStay.Common.Exceptions
{
public class OwnerAddAccommodationException : Exception
{
    public OwnerAddAccommodationException(string message) : base(message)
    {
    }

    public OwnerAddAccommodationException(string message, Exception innerException) : base(message,
innerException)
    {
    }

    public override string Message => base.Message;

    public override string ToString()
    {
        return $"{base.ToString()}";
    }
}
}
```

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.

```
namespace SmartStay.Common.Exceptions
{
public class OwnerCreationException : Exception
{
    public OwnerCreationException(string message) : base(message)
    {
    }

    public OwnerCreationException(string message, Exception innerException) : base(message,
innerException)
    {
    }

    public override string Message => base.Message;

    public override string ToString()
    {
        return $"{base.ToString()}";
    }
}
}
```

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.

```
namespace SmartStay.Common.Exceptions
{
public class ReservationCreationException : Exception
{
    public ReservationCreationException(string message) : base(message)
    {
    }

    public ReservationCreationException(string message, Exception innerException) : base(message,
innerException)
    {
    }

    public override string Message => base.Message;

    public override string ToString()
    {
        return $"{base.ToString()}";
    }
}
}
```

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.

```
namespace SmartStay.Common.Exceptions
{
public class TotalCostException : Exception
{
    public TotalCostException(string message) : base(message)
    {
    }

    public TotalCostException(string message, Exception innerException) : base(message,
innerException)
    {
    }

    public override string Message => base.Message;

    public override string ToString()
    {
        return $"{base.ToString()}";
    }
}
}
```

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.

```
namespace SmartStay.Common.Models
{
public class ImportResult
{
    public int ImportedCount { get; set; }

    public int ReplacedCount { get; set; }

    public int TotalCount => ImportedCount + ReplacedCount;

    public override string ToString()
    {
        return $"Imported: {ImportedCount}, Replaced: {ReplacedCount}, Total: {TotalCount}";
    }
}
}
```

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.

```
// <autogenerated />
using System;
using System.Reflection;
[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.

```
// <autogenerated />
using System;
using System.Reflection;
[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.

```
// <autogenerated />
using System;
using System.Reflection;
[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.

```
// <autogenerated />
using System;
using System.Reflection;
[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.

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//      Runtime Version:4.0.30319.42000
//
//      Changes to this file may cause incorrect behavior and will be lost if
//      the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

using System;
using System.Reflection;

[assembly: System.Reflection.AssemblyCompanyAttribute("SmartStay.Common")]
[assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]
[assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]
[assembly:
System.Reflection.AssemblyInformationalVersionAttribute("1.0.0+c366ac03947932e5126b804e73253b4d5f5e0e8d")]
[assembly: System.Reflection.AssemblyProductAttribute("SmartStay.Common")]
[assembly: System.Reflection.AssemblyTitleAttribute("SmartStay.Common")]
[assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]

// Generated by the MSBuild WriteCodeFragment class.

```

7.53 SmartStay.Common.GlobalUsings.g.cs File Reference

7.54 SmartStay.Common.GlobalUsings.g.cs

Go to the documentation of this file.

```
// <auto-generated/>
global using global::System;
global using global::System.Collections.Generic;
global using global::System.IO;
global using global::System.Linq;
global using global::System.Net.Http;
global using global::System.Threading;
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.

```

using System.Text.Encodings.Web;
using System.Text.Json;
using System.Text.Json.Serialization;
using ProtoBuf;
using SmartStay.Common.Enums;
using SmartStay.Validation;
using SmartStay.Validation.Validators;

namespace SmartStay.Core.Models
{
[ProtoContract]
public class Accommodation
{
    static int _lastAccommodationId = 0; // Last assigned accommodation ID

    static readonly JsonSerializerOptions _jsonOptions =
        new() { WriteIndented = true, Encoder = JavaScriptEncoder.UnsafeRelaxedJsonEscaping,
                Converters = { new JsonStringEnumConverter() } };

    [ProtoMember(1)]
    readonly int _id; // ID of the accommodation

    [ProtoMember(2)]
    int _ownerId; // ID of the owner

    [ProtoMember(3)]
    AccommodationType _type; // Type of accommodation (Hotel, House, etc.)

    [ProtoMember(4)]
    string _name; // Name of the accommodation

    [ProtoMember(5)]
    string _address; // Address of the accommodation

    [ProtoMember(6)]
    readonly List<Room> _rooms = []; // List of rooms

#pragma warning disable CS8618
    public Accommodation()
#pragma warning restore CS8618
    {
        // This constructor is intentionally empty and only needed for Protobuf-net deserialization.
    }

    public Accommodation(int ownerId, AccommodationType type, string name, string address)
    {
        AccommodationValidator.ValidateAccommodationType(type);
        NameValidator.ValidateAccommodationName(name);
        AddressValidator.ValidateAddress(address);

        _id = GenerateAccommodationId();
        _ownerId = ownerId;
        _type = type;
        _name = name;
        _address = address;
    }

    [JsonConstructor]
    public Accommodation(int id, int ownerId, AccommodationType type, string name, string address,
List<Room> rooms)
    {
        _id = id;
        UpdateLastAccommodationId(id);
        _ownerId = ownerId;
        _type = type;
        _name = name;
        _address = address;
        _rooms = rooms;
    }

    public static int LastAssignedId
    {
        get => _lastAccommodationId;
        set {
            if (_lastAccommodationId < value)
                _lastAccommodationId = value;
        }
    }

    public int Id => _id;

    public int OwnerId
    {
        get => _ownerId;
        set => _ownerId = OwnerValidator.ValidateOwnerId(value);
    }

    public AccommodationType Type
    {
        get => _type;
        set => _type = AccommodationValidator.ValidateAccommodationType(value);
    }

    public string Name
    {
        get => _name;
        set => _name = NameValidator.ValidateAccommodationName(value);
    }

    public string Address
    {
        get => _address;
        set => _address = AddressValidator.ValidateAddress(value);
    }

    public List<Room> Rooms => GetRoomsCopy();

    public Room? FindRoomById(int roomId)
    {
        return _rooms.Find(room => room.Id == roomId);
    }

    public bool AddRoom(Room room)
    {
        if (room == null)
        {
            return false; // If the room is null, return false
        }

        _rooms.Add(room);
        return true; // Successfully added the room
    }

    public bool DeleteRoom(int roomId)
    {
        var roomToDelete = _rooms.Find(r => r.Id == roomId);

        if (roomToDelete == null)
        {
            return false; // Return false if the room with the given ID is not found
        }

        _rooms.Remove(roomToDelete);
        return true; // Successfully removed the room
    }

    private static int GenerateAccommodationId()
    {
        // Check if the current value exceeds the max limit of int (2,147,483,647)
        if (_lastAccommodationId >= int.MaxValue)
        {
            throw new InvalidOperationException("Accommodation ID limit exceeded.");
        }

        return Interlocked.Increment(ref _lastAccommodationId);
    }

    private static void UpdateLastAccommodationId(int id)
    {
        if (id > _lastAccommodationId)
        {
            _lastAccommodationId = id; // Update the last assigned client ID if the new ID is larger
        }
    }

    private List<Room> GetRoomsCopy()
    {
        // Deep copy each room
        return _rooms.Select(room => room.Clone()).ToList();
    }

    public Accommodation Clone()
    {
        // Create a new instance of Accommodation and deep copy the fields
        return new Accommodation(_id,                                                  // Immutable
                                 _ownerId,                                             // Immutable
                                 _type,                                                // Enum, can be
 directly copied
```

```
                                  _name,                                      // String, can be
directly copied
                                  _address,                                   // String, can be
directly copied
                                  new List<Room>(_rooms.Select(room => room.Clone())) // Deep copy of
Room objects
                );
        }

        public override string ToString()
        {
            // Create a dictionary for the properties you want to serialize
            var accommodationData = new { Id = _id, Type = _type.ToString(), Name = _name, Address =
_address,
                                      Rooms = _rooms.Select(room => new {
                                          room.Id,
                                          room.PricePerNight,
                                          room.Type,
                                      }) };

            // Serialize the dictionary into a JSON string, which will include Rooms as an array
            return JsonSerializer.Serialize(accommodationData, _jsonOptions);
        }
    }
}
```

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 application. These models represent core entities and structures essential for managing application data.

7.58 Client.cs

Go to the documentation of this file.

```

using System.Text.Encodings.Web;
using System.Text.Json;
using System.Text.Json.Serialization;
using ProtoBuf;
using SmartStay.Common.Enums;
using SmartStay.Core.Repositories;
using SmartStay.Validation.Validators;

namespace SmartStay.Core.Models
{
[ProtoContract]
public class Client
{
    static int _lastClientId = 0; // Last assigned client ID

    static readonly JsonSerializerOptions _jsonOptions =
        new() { WriteIndented = true, Encoder = JavaScriptEncoder.UnsafeRelaxedJsonEscaping,
                Converters = { new JsonStringEnumConverter() } };

    [ProtoMember(1)]
    readonly int _id; // ID of the client

    [ProtoMember(2)]
    string _firstName; // First name of the client

    [ProtoMember(3)]
    string _lastName; // Last name of the client

    [ProtoMember(4)]
    string _email; // Email address of the client

    [ProtoMember(5)]
    string _phoneNumber = string.Empty; // Phone number of the client

    [ProtoMember(6)]
    string _address = string.Empty; // Address of the client

    [ProtoMember(7)]
    PaymentMethod _preferredPaymentMethod = PaymentMethod.None; // Preferred payment method of the
client

#pragma warning disable CS8618
    public Client()
#pragma warning restore CS8618
    {
        // This constructor is intentionally empty and only needed for Protobuf-net deserialization.
    }

    public Client(string firstName, string lastName, string email)
    {
        NameValidator.ValidateName(firstName);
        NameValidator.ValidateName(lastName);
        EmailValidator.ValidateEmail(email);

        _id = GenerateClientId();
        _firstName = firstName;
        _lastName = lastName;
        _email = email;
    }

    public Client(string firstName, string lastName, string email, string phoneNumber, string address)
        : this(firstName, lastName, email)
    {
        PhoneNumberValidator.ValidatePhoneNumber(phoneNumber);
        AddressValidator.ValidateAddress(address);

        _phoneNumber = phoneNumber;
        _address = address;
    }

    public Client(string firstName, string lastName, string email, string phoneNumber, string address,
                  PaymentMethod preferredPaymentMethod)
        : this(firstName, lastName, email, phoneNumber, address)
    {
        PaymentValidator.ValidatePaymentMethod(preferredPaymentMethod);

        _preferredPaymentMethod = preferredPaymentMethod;
    }

    [JsonConstructor]
    public Client(int id, string firstName, string lastName, string email, string phoneNumber, string
address,
                  PaymentMethod preferredPaymentMethod)
    {
        _id = id;
        UpdateLastClientId(id);
        _firstName = firstName;
        _lastName = lastName;
        _email = email;
        _phoneNumber = phoneNumber;
        _address = address;
        _preferredPaymentMethod = preferredPaymentMethod;
    }

    public static int LastAssignedId
    {
        get => _lastClientId;
        set {
            if (_lastClientId < value)
                _lastClientId = value;
        }
    }

    public int Id => _id;

    public string FirstName
    {
        get => _firstName;
        set => _firstName = NameValidator.ValidateName(value);
    }

    public string LastName
    {
        get => _lastName;
        set => _lastName = NameValidator.ValidateName(value);
    }

    public string Email
    {
        get => _email;
        set => _email = EmailValidator.ValidateEmail(value);
    }

    public string PhoneNumber
    {
        get => _phoneNumber;
        set => _phoneNumber = PhoneNumberValidator.ValidatePhoneNumber(value);
    }

    public string Address
    {
        get => _address;
        set => _address = AddressValidator.ValidateAddress(value);
    }

    public PaymentMethod PreferredPaymentMethod
    {
        get => _preferredPaymentMethod;
        set => _preferredPaymentMethod = PaymentValidator.ValidatePaymentMethod(value);
    }

    private static int GenerateClientId()
    {
        // Check if the current value exceeds the max limit of int (2,147,483,647)
        if (_lastClientId >= int.MaxValue)
        {
            throw new InvalidOperationException("Client ID limit exceeded.");
        }

        return Interlocked.Increment(ref _lastClientId);
    }

    private static void UpdateLastClientId(int id)
    {
        if (id > _lastClientId)
        {
            _lastClientId = id; // Update the last assigned client ID if the new ID is larger
        }
    }

    public override string ToString()
    {
        return JsonSerializer.Serialize(this, _jsonOptions);
    }
}
}
```

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 application. 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.

```
namespace SmartStay.Core.Models.Interfaces
{
using SmartStay.Common.Models;

public interface IManageableEntity<in T>
{
    bool Add(T item);

    bool Remove(T item);

    ImportResult Import(string data);

    string Export();

    void Save(string filePath);

    void Load(string filePath);
}
}
```

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 application. These models represent core entities and structures essential for managing application data.

7.62 Owner.cs

Go to the documentation of this file.

```
using System.Text.Encodings.Web;
using System.Text.Json;
using System.Text.Json.Serialization;
using ProtoBuf;
using SmartStay.Validation.Validators;

```

```
namespace SmartStay.Core.Models
{
[ProtoContract]
public class Owner
{
    static int _lastOwnerId = 0;

    static readonly JsonSerializerOptions _jsonOptions =
        new() { WriteIndented = true, Encoder = JavaScriptEncoder.UnsafeRelaxedJsonEscaping,
                Converters = { new JsonStringEnumConverter() } };

    [ProtoMember(1)]
    readonly int _id; // ID of the owner

    [ProtoMember(2)]
    string _firstName; // First name of the owner

    [ProtoMember(3)]
    string _lastName; // Last name of the owner

    [ProtoMember(4)]
    string _email; // Email address of the owner

    [ProtoMember(5)]
    string _phoneNumber = string.Empty; // Phone number of the owner

    [ProtoMember(6)]
    string _address = string.Empty; // Address of the owner

    [ProtoMember(7)]
    readonly List<Accommodation> _accommodationsOwned = []; // List of accommodations owned by the
owner

#pragma warning disable CS8618
    public Owner()
#pragma warning restore CS8618
    {
        // This constructor is intentionally empty and only needed for Protobuf-net deserialization.
    }

    public Owner(string firstName, string lastName, string email)
    {
        NameValidator.ValidateName(firstName);
        NameValidator.ValidateName(lastName);
        EmailValidator.ValidateEmail(email);

        _id = GenerateOwnerId();
        _firstName = firstName;
        _lastName = lastName;
        _email = email;
    }

    public Owner(string firstName, string lastName, string email, string phoneNumber, string address)
        : this(firstName, lastName, email)
    {
        PhoneNumberValidator.ValidatePhoneNumber(phoneNumber);
        AddressValidator.ValidateAddress(address);

        _phoneNumber = phoneNumber;
        _address = address;
    }

    [JsonConstructor]
    public Owner(int id, string firstName, string lastName, string email, string phoneNumber, string
address,
                 List<Accommodation> accommodationsOwned)
    {
        _id = id;
        UpdateLastOwnerId(id);
        _firstName = firstName;
        _lastName = lastName;
        _email = email;
        _phoneNumber = phoneNumber;
        _address = address;
        _accommodationsOwned = accommodationsOwned;
    }

    public static int LastAssignedId
    {
        get => _lastOwnerId;
        set {
            if (_lastOwnerId < value)
                _lastOwnerId = value;
        }
    }

    public int Id => _id;
    public string FirstName
    {
        get => _firstName;
        set => _firstName = NameValidator.ValidateName(value);
    }

    public string LastName
    {
        get => _lastName;
        set => _lastName = NameValidator.ValidateName(value);
    }

    public string Email
    {
        get => _email;
        set => _email = EmailValidator.ValidateEmail(value);
    }

    public string PhoneNumber
    {
        get => _phoneNumber;
        set => _phoneNumber = PhoneNumberValidator.ValidatePhoneNumber(value);
    }

    public string Address
    {
        get => _address;
        set => _address = AddressValidator.ValidateAddress(value);
    }

    public List<Accommodation> AccommodationsOwned => GetAccommodationsCopy();

    public bool AddAccommodation(Accommodation accommodation)
    {
        if (accommodation == null)
            return false;

        _accommodationsOwned.Add(accommodation);
        return true;
    }

    public bool RemoveAccommodation(Accommodation accommodation)
    {
        if (accommodation == null)
            return false;

        _accommodationsOwned.Remove(accommodation);
        return true;
    }

    private static int GenerateOwnerId()
    {
        // Check if the current value exceeds the max limit of int (2,147,483,647)
        if (_lastOwnerId >= int.MaxValue)
        {
            throw new InvalidOperationException("Owner ID limit exceeded.");
        }

        return Interlocked.Increment(ref _lastOwnerId);
    }

    private static void UpdateLastOwnerId(int id)
    {
        if (id > _lastOwnerId)
        {
            _lastOwnerId = id; // Update the last assigned owner ID if the new ID is larger
        }
    }

    private List<Accommodation> GetAccommodationsCopy()
    {
        // Deep copy each room
        return _accommodationsOwned.Select(accommodation => accommodation.Clone()).ToList();
    }

    public override string ToString()
    {
        return JsonSerializer.Serialize(this, _jsonOptions);
    }
}
}
```

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.

```

using System.ComponentModel.DataAnnotations;
using System.Text.Encodings.Web;
using System.Text.Json;
using System.Text.Json.Serialization;
using ProtoBuf;
using SmartStay.Common.Enums;
using SmartStay.Validation;
using SmartStay.Validation.Validators;

namespace SmartStay.Core.Models
{
[ProtoContract]
public class Payment
{
    static int _lastPaymentId = 0;

    static readonly JsonSerializerOptions _jsonOptions =
        new() { WriteIndented = true, Encoder = JavaScriptEncoder.UnsafeRelaxedJsonEscaping,
                Converters = { new JsonStringEnumConverter() } };

    [ProtoMember(1)]
    readonly int _id; // ID of the payment

    [ProtoMember(2)]
    int _reservationId; // ID of the reservation being paid

    [ProtoMember(3)]
    decimal _amount; // Amount of the payment

    [ProtoMember(4)]
    DateTime _date; // Date the payment was made

    [ProtoMember(5)]
    PaymentMethod _method; // Payment Method used

    [ProtoMember(6)]
    PaymentStatus _status; // Status of the payment

#pragma warning disable CS8618
    public Payment()
#pragma warning restore CS8618
    {
        // This constructor is intentionally empty and only needed for Protobuf-net deserialization.
    }

    public Payment(int reservationId, decimal amount, DateTime paymentDate, PaymentMethod
paymentMethod,
                   PaymentStatus paymentStatus)
    {
        ReservationValidator.ValidateReservationId(reservationId);
        PaymentValidator.ValidatePayment(amount);
        PaymentValidator.ValidatePaymentMethod(paymentMethod);
        PaymentValidator.ValidatePaymentStatus(paymentStatus);

        _id = GeneratePaymentId();
        _reservationId = reservationId;
        _amount = amount;
        _date = paymentDate;
        _method = paymentMethod;
        _status = paymentStatus;
    }

    [JsonConstructor]
    public Payment(int id, int reservationId, decimal amount, DateTime date, PaymentMethod method,
PaymentStatus status)
    {
        _id = id;
        UpdateLastPaymentId(id);
        _reservationId = reservationId;
        _amount = amount;
        _date = date;
        _method = method;
        _status = status;
    }

    public static int LastAssignedId
    {
        get => _lastPaymentId;
        set {
            if (_lastPaymentId < value)
                _lastPaymentId = value;
        }
    }

    public int Id => _id;

    public int ReservationId => _reservationId;

    public decimal Amount => _amount;

    public DateTime Date => _date;

    public PaymentMethod Method => _method;

    public PaymentStatus Status
    {

        get => _status;
        set => _status = PaymentValidator.ValidatePaymentStatus(value);
    }

    private static int GeneratePaymentId()
    {
        // Check if the current value exceeds the max limit of int (2,147,483,647)
        if (_lastPaymentId >= int.MaxValue)
        {
            throw new InvalidOperationException("Payment ID limit exceeded.");
        }

        return Interlocked.Increment(ref _lastPaymentId);
    }

    private static void UpdateLastPaymentId(int id)
    {
        if (id > _lastPaymentId)
        {
            _lastPaymentId = id; // Update the last assigned owner ID if the new ID is larger
        }
    }

    public Payment Clone()
    {
        // Create a new instance of Payment and deep copy the fields
        return new Payment(_id,            // Immutable, directly copy
                           _reservationId, // Value type, directly copy
                           _amount,        // Value type, directly copy
                           _date,          // Value type (DateTime), directly copy
                           _method,        // Enum type, directly copy
                           _status         // Enum type, directly copy
        );
    }

    public override string ToString()
    {
        return JsonSerializer.Serialize(this, _jsonOptions);
    }
}
}
```

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.

```
using System.Text.Encodings.Web;
using System.Text.Json;
using System.Text.Json.Serialization;
using ProtoBuf;
using SmartStay.Common.Enums;
using SmartStay.Validation;
using SmartStay.Validation.Validators;

namespace SmartStay.Core.Models
{
[ProtoContract]
public class Reservation
{
    static int _lastReservationId = 0;

    static readonly JsonSerializerOptions _jsonOptions =
        new() { WriteIndented = true, Encoder = JavaScriptEncoder.UnsafeRelaxedJsonEscaping,
                Converters = { new JsonStringEnumConverter() } };

    [ProtoMember(1)]
    readonly int _reservationId; // ID of the reservation

    [ProtoMember(2)]
    readonly int _clientId; // ID of the client making the reservation

    [ProtoMember(3)]
    readonly int _accommodationId; // ID of the accommodation

    [ProtoMember(4)]
    readonly int _roomId; // ID of the room

    [ProtoMember(5)]
    AccommodationType _accommodationType; // Type of accommodation (e.g., Room, Suite, etc.)

    [ProtoMember(6)]
    DateTime _checkInDate; // Check-in date for the reservation

    [ProtoMember(7)]
    DateTime _checkOutDate; // Check-out date for the reservation

    [ProtoMember(8)]
    ReservationStatus _status = ReservationStatus.Pending; // Current reservation status

    [ProtoMember(9)]
    decimal _totalCost; // Total cost of the reservation

    [ProtoMember(10)]
    decimal _amountPaid = 0; // Amount paid towards the reservation

    [ProtoMember(11)]
    readonly List<Payment> _payments = []; // List of payments made for the reservation

#pragma warning disable CS8618
    public Reservation()
#pragma warning restore CS8618
    {
        // This constructor is intentionally empty and only needed for Protobuf-net deserialization.
    }

    public Reservation(int clientId, int accommodationId, int roomId, AccommodationType accommodationType,
                       DateTime checkInDate, DateTime checkOutDate, decimal totalCost)
    {
        ClientValidator.ValidateClientId(clientId);
        AccommodationValidator.ValidateAccommodationId(accommodationId);
        RoomValidator.ValidateRoomId(roomId);
        PaymentValidator.ValidateTotalCost(totalCost);
        if (!DateValidator.IsValidDateRange(checkInDate, checkOutDate))
            throw new ValidationException(ValidationErrorCode.InvalidDateRange);

        _reservationId = GenerateReservationId();
        _clientId = clientId;
        _accommodationId = accommodationId;
        _roomId = roomId;
        _accommodationType = accommodationType;
        _checkInDate = checkInDate;
        _checkOutDate = checkOutDate;
        _totalCost = totalCost;
    }

    [JsonConstructor]
    public Reservation(int id, int clientId, int accommodationId, int roomId, AccommodationType accommodationType,
                       DateTime checkInDate, DateTime checkOutDate, ReservationStatus status, decimal totalCost,
                       decimal amountPaid, List<Payment> payments)
    {
        _reservationId = id;
        UpdateLastReservationId(id);
        _clientId = clientId;
        _accommodationId = accommodationId;
        _roomId = roomId;
        _accommodationType = accommodationType;
        _checkInDate = checkInDate;
        _checkOutDate = checkOutDate;
        _status = status;
        _totalCost = totalCost;
        _amountPaid = amountPaid;
        _payments = payments;
    }

    public static int LastAssignedId
    {
        get => _lastReservationId;
        set {
            if (_lastReservationId < value)
                _lastReservationId = value;
        }
    }

    public int Id => _reservationId;

    public int ClientId => _clientId;

    public int AccommodationId => _accommodationId;

    public int RoomId => _roomId;

    public AccommodationType AccommodationType
    {
        get => _accommodationType;
        set => _accommodationType = AccommodationValidator.ValidateAccommodationType(value);
    }

    public DateTime CheckInDate
    {
        get => _checkInDate;
        set => _checkInDate = DateValidator.ValidateCheckInDate(value);
    }

    public DateTime CheckOutDate
    {
        get => _checkOutDate;
        set => _checkOutDate = DateValidator.ValidateCheckOutDate(value, _checkInDate);
    }

    public ReservationStatus Status
    {
        get => _status;
        set => _status = ReservationValidator.ValidateReservationStatus(value);
    }

    public decimal TotalCost
    {
        get => _totalCost;
        set => _totalCost = PaymentValidator.ValidateTotalCost(value);
    }

    public decimal AmountPaid
    {
        get => _amountPaid;
        set => _amountPaid = PaymentValidator.ValidatePayment(value);
    }

    public List<Payment> Payments => GetPaymentsCopy();

    public bool CheckIn()
    {
        if (_status != ReservationStatus.Pending)
        {
            return false;
        }
        _status = ReservationStatus.CheckedIn;
        return true;
    }

    public bool CheckOut()
    {
        if (_status != ReservationStatus.CheckedIn)
        {
            return false;
        }
        _status = ReservationStatus.CheckedOut;
        return true;
    }

    public PaymentResult MakePayment(decimal paymentAmount, PaymentMethod paymentMethod)
    {
        if (paymentAmount <= 0)
            return PaymentResult.InvalidAmount;
        if (IsFullyPaid())
            return PaymentResult.AlreadyFullyPaid;
        if (paymentAmount > _totalCost - _amountPaid)
            return PaymentResult.AmountExceedsTotal;
        if (!PaymentValidator.IsValidPaymentMethod(paymentMethod))
            return PaymentResult.InvalidPaymentMethod;

        // Create a new Payment instance and add it to the list
        var payment = new Payment(_reservationId, paymentAmount, DateTime.Now, paymentMethod,
PaymentStatus.Completed);
        _payments.Add(payment);

        // Update the amount paid
        _amountPaid += paymentAmount;

        return PaymentResult.Success;
    }

    public bool IsFullyPaid()
    {
        return _amountPaid >= _totalCost;
    }

    private static int GenerateReservationId()
    {
        if (_lastReservationId >= int.MaxValue)
        {
            throw new InvalidOperationException("Reservation ID limit exceeded.");
        }
        return Interlocked.Increment(ref _lastReservationId);
    }

    private static void UpdateLastReservationId(int id)
    {
        if (id > _lastReservationId)
        {
            _lastReservationId = id; // Update the last assigned owner ID if the new ID is larger
        }
    }

    private List<Payment> GetPaymentsCopy()
    {
        // Deep copy each payment
        return _payments.Select(payment => payment.Clone()).ToList();
    }

    public override string ToString()
    {
        return JsonSerializer.Serialize(this, _jsonOptions);
    }
}
}
```

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.

```

using System.Text.Encodings.Web;
using System.Text.Json;
using System.Text.Json.Serialization;
using ProtoBuf;
using SmartStay.Common.Enums;
using SmartStay.Core.Utilities;
using SmartStay.Validation;
using SmartStay.Validation.Validators;

namespace SmartStay.Core.Models
{
[ProtoContract]
public class Room
{
    static int _lastRoomId = 0; // Last assigned room ID

    static readonly JsonSerializerOptions _jsonOptions =
        new() { WriteIndented = true, Encoder = JavaScriptEncoder.UnsafeRelaxedJsonEscaping,
                Converters = { new JsonStringEnumConverter() } };

    [ProtoMember(1)]
    readonly int _id; // ID of the room

    [ProtoMember(2)]
    RoomType _type; // Type of room (Single, Double, Suite, etc.)

    [ProtoMember(3)]
    decimal _pricePerNight; // Price per night for the room

    [ProtoMember(4)]
    readonly SortedSet<DateRange> _reservationDates = []; // Sorted set for efficient availability
check

    public Room()
    {
        // This constructor is intentionally empty and only needed for Protobuf-net deserialization.
    }

    public Room(RoomType type, decimal pricePerNight)
    {
        RoomValidator.ValidateRoomType(type);
        PaymentValidator.ValidatePrice(pricePerNight);

        _id = GenerateRoomId();
        _type = type;
        _pricePerNight = pricePerNight;
    }

    [JsonConstructor]
    public Room(int id, RoomType type, decimal pricePerNight, SortedSet<DateRange> reservationDates)
    {
        _id = id;
        UpdateLastRoomId(id);
        _type = type;
        _pricePerNight = pricePerNight;
        _reservationDates = reservationDates;
    }

    public static int LastAssignedId
    {
        get => _lastRoomId;
        set {
            if (_lastRoomId < value)
                _lastRoomId = value;
        }
    }

    public int Id => _id;

    public RoomType Type
    {
        get => _type;
        set => _type = RoomValidator.ValidateRoomType(value);
    }

    public decimal PricePerNight
    {
        get => _pricePerNight;
        set => _pricePerNight = PaymentValidator.ValidatePrice(value);
    }

    public SortedSet<DateRange> ReservationDates => GetReservationDatesCopy();

    public bool IsAvailable(DateTime startDate, DateTime endDate, DateRange? existingReservationRange
= null)
    {
        if (endDate <= startDate)
            throw new ArgumentException("End date must be after the start date.");

        var newReservation = new DateRange(startDate, endDate);

        // Get potential conflicting reservations within the requested range
        var potentialConflicts = _reservationDates.GetViewBetween(
            new DateRange(DateTime.MinValue, startDate), // All reservations that end before the start
            new DateRange(DateTime.MaxValue, endDate)    // All reservations that start after the end
        );

        // Check if there are any overlapping reservations
        foreach (var existingReservation in potentialConflicts)
        {
            // Skip the existing reservation if it's the one we're trying to modify
            if (existingReservation.Equals(existingReservationRange))
            {
                continue; // Skip this reservation as it's the one we're modifying
            }

            // An overlap occurs if the start date is before the end date, and the end date is after
the start date
            if ((newReservation.Start < existingReservation.End) && (newReservation.End >
existingReservation.Start))
            {
                return false; // There's an overlap, so the accommodation is not available
            }
        }

        return true; // No overlap found, accommodation is available
    }

    public bool AddReservation(DateTime startDate, DateTime endDate)
    {
        var newReservation = new DateRange(startDate, endDate);

        // Attempt to add the new reservation
        bool addedSuccessfully = _reservationDates.Add(newReservation);

        // Return the result of the Add operation (true if added, false if it already exists due to overlap)
        return addedSuccessfully;
    }

    public bool RemoveReservation(DateTime startDate, DateTime endDate)
    {
        // Create the date range object to remove
        var reservationToRemove = new DateRange(startDate, endDate);

        // Remove the reservation from the SortedSet
        bool removed = _reservationDates.Remove(reservationToRemove);

        // Return whether the reservation was successfully removed
        return removed;
    }

    public decimal CalculateTotalCost(DateTime startDate, DateTime endDate)
    {
        if (endDate <= startDate)
        {
            throw new ArgumentException("End date must be after the start date.");
        }

        int nights = (endDate - startDate).Days;
        return nights * _pricePerNight;
    }

    private static int GenerateRoomId()
    {
        if (_lastRoomId >= int.MaxValue)
            throw new InvalidOperationException("Room ID limit exceeded.");

        return Interlocked.Increment(ref _lastRoomId);
    }

    private static void UpdateLastRoomId(int id)
    {
        if (id > _lastRoomId)
        {
            _lastRoomId = id; // Update the last assigned owner ID if the new ID is larger
        }
    }

    private SortedSet<DateRange> GetReservationDatesCopy()
    {
        // Deep copy each DateRange in the collection
        return new SortedSet<DateRange>(_reservationDates.Select(dateRange => dateRange.Clone()));
    }

    public Room Clone()
    {
        // Create a new instance of Room and deep copy the fields
        return new Room(
            _id,                                                             // Immutable
            _type,                                                           // Immutable
            _pricePerNight,                                                  // Value type, directly copy
            new SortedSet<DateRange>(_reservationDates.Select(dr => dr.Clone())) // Deep copy of DateRange objects
        );
    }

    public override string ToString()
    {
        return JsonSerializer.Serialize(this, _jsonOptions);
    }
}
}
```

7.69 SmartStay.Core.AssemblyInfo.cs File Reference

7.70 SmartStay.Core.AssemblyInfo.cs

Go to the documentation of this file.

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:4.0.30319.42000
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

using System;
using System.Reflection;

[assembly: System.Reflection.AssemblyCompanyAttribute("SmartStay.Core")]
[assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]
[assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]
[assembly:
System.Reflection.AssemblyInformationalVersionAttribute("1.0.0+55fef41297156f945c89b29ec6d959f5a875a2ea")]
[assembly: System.Reflection.AssemblyProductAttribute("SmartStay.Core")]
[assembly: System.Reflection.AssemblyTitleAttribute("SmartStay.Core")]
[assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]

// Generated by the MSBuild WriteCodeFragment class.

```

7.71 SmartStay.Core.GlobalUsings.g.cs File Reference

7.72 SmartStay.Core.GlobalUsings.g.cs

Go to the documentation of this file.

```
// <auto-generated/>
global using global::System;
global using global::System.Collections.Generic;
global using global::System.IO;
global using global::System.Linq;
global using global::System.Net.Http;
global using global::System.Threading;
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.

```
#nullable enable
using System.Runtime.Serialization;
using ProtoBuf;
using SmartStay.Common.Models;
using SmartStay.Core.Models;
using SmartStay.Core.Models.Interfaces;
using SmartStay.Core.Utilities;

namespace SmartStay.Core.Repositories
{
[ProtoContract]
public class Accommodations : IManageableEntity<Accommodation>
{
    readonly Dictionary<int, Accommodation> _accommodationDictionary = new();

    [ProtoMember(1)] // Serialize the list of accommodations
    List<Accommodation> _accommodationList = new();

    public bool Add(Accommodation accommodation)
    {
        if (accommodation == null)
        {
            throw new ArgumentNullException(nameof(accommodation), "Accommodation cannot be null");
        }

        if (_accommodationDictionary.ContainsKey(accommodation.Id))
        {
            return false; // Accommodation already exists
        }

        _accommodationDictionary[accommodation.Id] = accommodation;
        return true; // Accommodation added successfully
    }

    public bool Remove(Accommodation accommodation)
    {
        if (accommodation == null)
        {
            throw new ArgumentNullException(nameof(accommodation), "Accommodation cannot be null");
        }

        return _accommodationDictionary.Remove(accommodation.Id); // Remove using accommodation ID
    }

    public ImportResult Import(string data)
    {
        if (string.IsNullOrEmpty(data))
        {
            throw new ArgumentException("Data cannot be null or empty", nameof(data));
        }

        // Deserialize the data into a List<Accommodation> instead of a single Accommodation
        var accommodations =
            JsonHelper.DeserializeFromJson<Accommodation>(data) ??
            throw new ArgumentException("Deserialized accommodation data cannot be null", nameof(data));

        int replacedCount = 0;
        int importedCount = 0;

        foreach (var accommodation in accommodations)
        {
            if (_accommodationDictionary.ContainsKey(accommodation.Id))
            {
                replacedCount++;
            }
            else
            {
                importedCount++;
            }
            _accommodationDictionary[accommodation.Id] = accommodation; // Direct insertion for efficiency
        }

        return new ImportResult { ImportedCount = importedCount, ReplacedCount = replacedCount };
    }

    public string Export()
    {
        return JsonHelper.SerializeToJson(_accommodationDictionary.Values);
    }
    [ProtoBeforeSerialization]
    private void PrepareForSerialization()
    {
        // Clear the temporary list to ensure no leftover data
        _accommodationList.Clear();

        // Add all accommodations from the dictionary to the temporary list
        foreach (var accommodation in _accommodationDictionary.Values)
        {
            _accommodationList.Add(accommodation);
        }
    }

    [ProtoAfterDeserialization]
    [System.Diagnostics.CodeAnalysis.SuppressMessage("CodeQuality", "IDE0051:Remove unused private members",
                                                     Justification =
                                                         "IDE Error, this is called automatically by protobuf-net.")]
    private void AfterDeserialization()
    {
        // Clear the dictionary before rebuilding
        _accommodationDictionary.Clear();

        // Rebuild the dictionary using the data from the list
        foreach (var accommodation in _accommodationList)
        {
            _accommodationDictionary[accommodation.Id] = accommodation;
        }

        // Clear the temporary list once the dictionary is rebuilt
        _accommodationList.Clear();

        // Set _lastAccommodationId to the highest ID in the deserialized data
        if (_accommodationDictionary.Count > 0)
        {
            // Find the highest ID from the loaded accommodations
            Accommodation.LastAssignedId = _accommodationDictionary.Values.Max(a => a.Id);
        }
        else
        {
            // If no accommodations, reset to 0
            Accommodation.LastAssignedId = 0;
        }
    }

    public void Save(string filePath)
    {
        try
        {
            // Prepare for serialization by copying the dictionary contents to the temporary list
            PrepareForSerialization();

            // Open the file stream for saving the data to the specified file
            using (var fileStream = File.Create(filePath))
            {
                // Serialize the accommodations object and write it to the file stream
                Serializer.Serialize(fileStream, this);
            }
        }
        catch (IOException ioEx)
        {
            throw new IOException("An error occurred while saving the accommodations data.", ioEx);
        }
        catch (SerializationException serEx)
        {
            throw new SerializationException(
                "An error occurred during serialization while saving the accommodations data.", serEx);
        }
    }

    public void Load(string filePath)
    {
        try
        {
            // Open the file stream for reading
            using (var fileStream = File.OpenRead(filePath))
            {
                // Deserialize the accommodations object from the file
                var accommodations = Serializer.Deserialize<Accommodations>(fileStream);

                // Clear the current dictionary
                _accommodationDictionary.Clear();

                // Reset LastAssignedId to ensure consistency
                Accommodation.LastAssignedId = 0;

                // Iterate over the deserialized data and copy it
                foreach (var accommodation in accommodations._accommodationDictionary)
                {
                    _accommodationDictionary[accommodation.Key] = accommodation.Value;
                    Accommodation.LastAssignedId = Math.Max(Accommodation.LastAssignedId, accommodation.Value.Id);
                }
            }
        }
        catch (IOException ioEx)
        {
            throw new IOException("An error occurred while loading the accommodations data.", ioEx);
        }
        catch (SerializationException serEx)
        {
            throw new SerializationException(
                "An error occurred during deserialization while loading the accommodations data.", serEx);
        }
    }

    public Accommodation? FindAccommodationById(int accommodationId)
    {
      _accommodationDictionary.TryGetValue(accommodationId, out Accommodation? accommodation);
      return accommodation;
    }

    public int CountAccommodations()
    {
        return _accommodationDictionary.Count;
    }
}
}
```

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.

```

#nullable enable
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using ProtoBuf;
using SmartStay.Common.Models;
using SmartStay.Core.Models;
using SmartStay.Core.Models.Interfaces;
using SmartStay.Core.Utilities;

```

```
namespace SmartStay.Core.Repositories
{
[ProtoContract]
public class Clients : IManageableEntity<Client>
{
    readonly Dictionary<int, Client> _clientDictionary = new();

    [ProtoMember(1)] // Serialize the list of clients
    List<Client> _clientList = new();

    public bool Add(Client client)
    {
        if (client == null)
        {
            throw new ArgumentNullException(nameof(client), "Client cannot be null");
        }

        if (_clientDictionary.ContainsKey(client.Id))
        {
            return false; // Client already exists
        }

        _clientDictionary[client.Id] = client;
        return true; // Client added successfully
    }

    public bool Remove(Client client)
    {
        if (client == null)
        {
            throw new ArgumentNullException(nameof(client), "Client cannot be null");
        }

        return _clientDictionary.Remove(client.Id); // Remove by client ID
    }

    public ImportResult Import(string data)
    {
        if (string.IsNullOrEmpty(data))
        {
            throw new ArgumentException("Data cannot be null or empty", nameof(data));
        }

        // Deserialize the data into a List<Client> instead of a single Client
        var clients = JsonHelper.DeserializeFromJson<Client>(data) ??
                      throw new ArgumentException("Deserialized client data cannot be null",
nameof(data));

        int replacedCount = 0;
        int importedCount = 0;

        foreach (var client in clients)
        {
            if (_clientDictionary.ContainsKey(client.Id))
            {
                replacedCount++;
            }
            else
            {
                importedCount++;
            }
            _clientDictionary[client.Id] = client; // Direct insertion for efficiency
        }

        return new ImportResult { ImportedCount = importedCount, ReplacedCount = replacedCount };
    }

    public string Export()
    {
        return JsonHelper.SerializeToJson(_clientDictionary.Values ?? Enumerable.Empty<Client>());
    }

    [ProtoBeforeSerialization]
    private void PrepareForSerialization()
    {
        // Clear the temporary list to ensure no leftover data
        _clientList.Clear();

        // Add all clients from the dictionary to the temporary list
        foreach (var client in _clientDictionary.Values)
        {
            _clientList.Add(client);
        }
    }

    [ProtoAfterDeserialization]
    [System.Diagnostics.CodeAnalysis.SuppressMessage("CodeQuality", "IDE0051:Remove unused private
members",
                                                     Justification =
                                                         "IDE Error, this is called automatically by
protobuf-net.")]
    private void AfterDeserialization()
    {
        // Clear the dictionary before rebuilding
        _clientDictionary.Clear();

        // Rebuild the dictionary using the data from the list
        foreach (var client in _clientList)
        {
            _clientDictionary[client.Id] = client;
        }

        // Clear the temporary list once the dictionary is rebuilt
        _clientList.Clear();
    }

    public void Save(string filePath)
    {
        try
        {
            // Prepare for serialization by copying the dictionary contents to the temporary list
            PrepareForSerialization();

            // Open the file stream for saving the data to the specified file
            using (var fileStream = File.Create(filePath))
            {
                // Serialize the clients object and write it to the file stream
                Serializer.Serialize(fileStream, this);
            }
        }
        catch (IOException ioEx)
        {
            throw new IOException("An error occurred while saving the clients data.", ioEx);
        }
        catch (SerializationException serEx)
        {
            throw new SerializationException("An error occurred during serialization while saving the
clients data.",
                                             serEx);
        }
    }

    public void Load(string filePath)
    {
        try
        {
            // Open the file stream for reading
            using (var fileStream = File.OpenRead(filePath))
            {
                // Deserialize the clients object from the file
                var clients = Serializer.Deserialize<Clients>(fileStream);

                // Clear the current dictionary
                _clientDictionary.Clear();

                // Reset LastAssignedId to ensure consistency
                Client.LastAssignedId = 0;

                // Iterate over the deserialized data and copy it
                foreach (var client in clients._clientDictionary)
                {
                    _clientDictionary[client.Key] = client.Value;

                    // Update LastAssignedId to the maximum ID found so far
                    Client.LastAssignedId = Math.Max(Client.LastAssignedId, client.Value.Id);
                }
            }
        }
        catch (IOException ioEx)
        {
            throw new IOException("An error occurred while loading the clients data.", ioEx);
        }
        catch (SerializationException serEx)
        {
            throw new SerializationException("An error occurred during deserialization while loading
the clients data.",
                                             serEx);
        }
    }

    public Client? FindClientById(int id)
    {
        _clientDictionary.TryGetValue(id, out Client? client);
        return client;
    }

    public int CountClients()
    {
        return _clientDictionary.Count;
    }
}
}
```

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.

```

#nullable enable
using System.Runtime.Serialization;
using ProtoBuf;
using SmartStay.Common.Models;
using SmartStay.Core.Models;
using SmartStay.Core.Models.Interfaces;
using SmartStay.Core.Utilities;

namespace SmartStay.Core.Repositories
{
[ProtoContract]
public class Owners : IManageableEntity<Owner>
{
    readonly Dictionary<int, Owner> _ownerDictionary = new();

    [ProtoMember(1)] // Serialize the list of owners
    List<Owner> _ownerList = new();

    public bool Add(Owner owner)
    {
        if (owner == null)
        {
            throw new ArgumentNullException(nameof(owner), "Owner cannot be null");
        }

        if (_ownerDictionary.ContainsKey(owner.Id))
        {
            return false; // Owner already exists
        }

        _ownerDictionary[owner.Id] = owner;
        return true; // Owner added successfully
    }

    public bool Remove(Owner owner)
    {
        if (owner == null)
        {
            throw new ArgumentNullException(nameof(owner), "Owner cannot be null");
        }

        return _ownerDictionary.Remove(owner.Id); // Remove by owner ID
    }

    public ImportResult Import(string data)
    {
        if (string.IsNullOrEmpty(data))
        {
            throw new ArgumentException("Data cannot be null or empty", nameof(data));
        }

        // Deserialize the data into a List<Owner> instead of a single Owner
        var owners = JsonHelper.DeserializeFromJson<Owner>(data) ??
                     throw new ArgumentException("Deserialized owner data cannot be null",
nameof(data));

        int replacedCount = 0;
        int importedCount = 0;

        foreach (var owner in owners)
        {
            if (_ownerDictionary.ContainsKey(owner.Id))
            {
                replacedCount++;
            }
            else
            {
                importedCount++;
            }
            _ownerDictionary[owner.Id] = owner; // Direct insertion for efficiency
        }

        return new ImportResult { ImportedCount = importedCount, ReplacedCount = replacedCount };
    }

    public string Export()
    {
        return JsonHelper.SerializeToJson(_ownerDictionary.Values ?? Enumerable.Empty<Owner>());
    }

    [ProtoBeforeSerialization]
    private void PrepareForSerialization()
    {
        // Clear the temporary list to ensure no leftover data
        _ownerList.Clear();

        // Add all owners from the dictionary to the temporary list
        foreach (var owner in _ownerDictionary.Values)
        {
            _ownerList.Add(owner);
        }
    }

    [ProtoAfterDeserialization]
    [System.Diagnostics.CodeAnalysis.SuppressMessage("CodeQuality", "IDE0051:Remove unused private
members",
                                                     Justification =
                                                         "IDE Error, this is called automatically by
protobuf-net.")]
    private void AfterDeserialization()
    {
        // Clear the dictionary before rebuilding
        _ownerDictionary.Clear();

        // Rebuild the dictionary using the data from the list
        foreach (var owner in _ownerList)
        {
            _ownerDictionary[owner.Id] = owner;
        }

        // Clear the temporary list once the dictionary is rebuilt
        _ownerList.Clear();

        // Set _lastOwnerId to the highest ID in the deserialized data
        if (_ownerDictionary.Count > 0)
        {
            // Find the highest ID from the loaded owners
            Owner.LastAssignedId = _ownerDictionary.Values.Max(o => o.Id);
        }
        else
        {
            // If no owners, reset to 0
            Owner.LastAssignedId = 0;
        }
    }

    public void Save(string filePath)
    {
        try
        {
            // Prepare for serialization by copying the dictionary contents to the temporary list
            PrepareForSerialization();

            // Open the file stream for saving the data to the specified file
            using (var fileStream = File.Create(filePath))
            {
                // Serialize the owners object and write it to the file stream
                Serializer.Serialize(fileStream, this);
            }
        }
        catch (IOException ioEx)
        {
            throw new IOException("An error occurred while saving the owners data.", ioEx);
        }
        catch (SerializationException serEx)
        {
            throw new SerializationException("An error occurred during serialization while saving the
owners data.",
                                             serEx);
        }
    }

    public void Load(string filePath)
    {
        try
        {
            // Open the file stream for reading
            using (var fileStream = File.OpenRead(filePath))
            {
                // Deserialize the owners object from the file
                var owners = Serializer.Deserialize<Owners>(fileStream);

                // Clear the current dictionary
                _ownerDictionary.Clear();

                // Reset LastAssignedId to ensure consistency
                Owner.LastAssignedId = 0;

                // Iterate over the deserialized data and copy it
                foreach (var owner in owners._ownerDictionary)
                {
                    _ownerDictionary[owner.Key] = owner.Value;

                    // Update LastAssignedId to the maximum ID found so far
                    Owner.LastAssignedId = Math.Max(Owner.LastAssignedId, owner.Value.Id);
                }
            }
        }
        catch (IOException ioEx)
        {
            throw new IOException("An error occurred while loading the owners data.", ioEx);
        }
        catch (SerializationException serEx)
        {
            throw new SerializationException("An error occurred during deserialization while loading
the owners data.",
                                             serEx);
        }
    }

    public Owner? FindOwnerById(int id)
    {
      _ownerDictionary.TryGetValue(id, out Owner? owner);
      return owner;
    }

    public int CountOwners()
    {
        return _ownerDictionary.Count;
    }
}
}
```

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.

```
#nullable enable
using System.Runtime.Serialization;
using ProtoBuf;
using SmartStay.Common.Models;
using SmartStay.Core.Models;
using SmartStay.Core.Models.Interfaces;
using SmartStay.Core.Utilities;

namespace SmartStay.Core.Repositories
{
[ProtoContract]
public class Reservations : IManageableEntity<Reservation>
{
    readonly Dictionary<int, Reservation> _reservationDictionary = new Dictionary<int, Reservation>();

    [ProtoMember(1)] // Serialize the list of accommodations
    List<Reservation> _reservationList = new();

    public bool Add(Reservation reservation)
    {
        if (reservation == null)
        {
            throw new ArgumentNullException(nameof(reservation), "Reservation cannot be null");
        }

        if (_reservationDictionary.ContainsKey(reservation.Id))
        {
            return false; // Reservation already exists
        }

        _reservationDictionary[reservation.Id] = reservation;
        return true; // Reservation added successfully
    }

    public bool Remove(Reservation reservation)
    {
        if (reservation == null)
        {
            throw new ArgumentNullException(nameof(reservation), "Reservation cannot be null");
        }

        // Remove the reservation using its ID
        return _reservationDictionary.Remove(reservation.Id);
    }

    public ImportResult Import(string data)
    {
        if (string.IsNullOrEmpty(data))
        {
            throw new ArgumentException("Data cannot be null or empty", nameof(data));
        }

        // Deserialize the data into a List<Reservation> instead of a single Reservation
        var reservations = JsonHelper.DeserializeFromJson<Reservation>(data) ??
                           throw new ArgumentException("Deserialized reservation data cannot be null",
nameof(data));

        int replacedCount = 0;
        int importedCount = 0;

        foreach (var reservation in reservations)
        {
            if (_reservationDictionary.ContainsKey(reservation.Id))
            {
                replacedCount++;
            }
            else
            {
                importedCount++;
            }
            _reservationDictionary[reservation.Id] = reservation; // Direct insertion for efficiency
        }

        return new ImportResult { ImportedCount = importedCount, ReplacedCount = replacedCount };
    }

    public string Export()
    {
        return JsonHelper.SerializeToJson<Reservation>(_reservationDictionary.Values);
    }

    [ProtoBeforeSerialization]
    private void PrepareForSerialization()
    {
        // Clear the temporary list to ensure no leftover data
        _reservationList.Clear();

        // Add all reservations from the dictionary to the temporary list
        foreach (var reservation in _reservationDictionary.Values)
        {
            _reservationList.Add(reservation);
        }
    }

    [ProtoAfterDeserialization]
    [System.Diagnostics.CodeAnalysis.SuppressMessage("CodeQuality", "IDE0051:Remove unused private
members",
                                                     Justification =
                                                         "IDE Error, this is called automatically by
protobuf-net.")]
    private void AfterDeserialization()
    {
        // Clear the dictionary before rebuilding
        _reservationDictionary.Clear();

        // Rebuild the dictionary using the data from the list
        foreach (var reservation in _reservationList)
        {
            _reservationDictionary[reservation.Id] = reservation;
        }

        // Clear the temporary list once the dictionary is rebuilt
        _reservationList.Clear();

        // Set _lastReservationId to the highest ID in the deserialized data
        if (_reservationDictionary.Count > 0)
        {
            // Find the highest ID from the loaded reservations
            Reservation.LastAssignedId = _reservationDictionary.Values.Max(r => r.Id);
        }
        else
        {
            // If no reservations, reset to 0
            Reservation.LastAssignedId = 0;
        }
    }

    public void Save(string filePath)
    {
        try
        {
            // Prepare for serialization by copying the dictionary contents to the temporary list
            PrepareForSerialization();

            // Open the file stream for saving the data to the specified file
            using (var fileStream = File.Create(filePath))
            {
                // Serialize the reservations object and write it to the file stream
                Serializer.Serialize(fileStream, this);
            }
        }
        catch (IOException ioEx)
        {
            throw new IOException("An error occurred while saving the reservations data.", ioEx);
        }
        catch (SerializationException serEx)
        {
            throw new SerializationException(
                "An error occurred during serialization while saving the reservations data.", serEx);
        }
    }

    public void Load(string filePath)
    {
        try
        {
            // Open the file stream for reading
            using (var fileStream = File.OpenRead(filePath))
            {
                // Deserialize the reservations object from the file
                var reservations = Serializer.Deserialize<Reservations>(fileStream);

                // Clear the current dictionary
                _reservationDictionary.Clear();

                // Reset LastAssignedId to ensure consistency
                Reservation.LastAssignedId = 0;

                // Iterate over the deserialized data and copy it
                foreach (var reservation in reservations._reservationDictionary)
                {
                    _reservationDictionary[reservation.Key] = reservation.Value;

                    // Update LastAssignedId to the maximum ID found so far
                    Reservation.LastAssignedId = Math.Max(Reservation.LastAssignedId,
reservation.Value.Id);
                }
            }
        }
        catch (IOException ioEx)
        {
            throw new IOException("An error occurred while loading the reservations data.", ioEx);
        }
        catch (SerializationException serEx)
        {
            throw new SerializationException(
                "An error occurred during deserialization while loading the reservations data.",
serEx);
        }
    }

    public Reservation? FindReservationById(int reservationId)
    {
      _reservationDictionary.TryGetValue(reservationId, out Reservation? reservation);
      return reservation;
    }

    public IEnumerable<Reservation> FindReservationsByClientId(int clientId)
    {
        return _reservationDictionary.Values.Where(r => r.ClientId == clientId);
    }

    public IEnumerable<Reservation> FindReservationsByAccommodationId(int accommodationId)
    {
        return _reservationDictionary.Values.Where(r => r.AccommodationId == accommodationId);
    }

    public IEnumerable<Reservation> GetFutureReservations(int accommodationId)
    {
        // Use LINQ to filter the dictionary's values directly without copying to a list.
        return _reservationDictionary.Values
            .Where(reservation =>
                       reservation.AccommodationId == accommodationId && reservation.CheckInDate >=
DateTime.Now)
            .ToList();
    }

    public int CountReservations()
    {
        return _reservationDictionary.Count;
    }
}
}
```

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

Go to the documentation of this file.

```
#nullable enable
using Microsoft.Extensions.Logging;
using SmartStay.Common.Enums;
using SmartStay.Common.Exceptions;
using SmartStay.Core.Models;
using SmartStay.Core.Repositories;
using SmartStay.Core.Utilities;
using SmartStay.Validation;

namespace SmartStay.Core.Services
{
public class BookingManager
{
#region Fields and Properties

    readonly ILogger<BookingManager> _logger;

    internal readonly Owners _owners = new();

    internal readonly Clients _clients = new();

    internal readonly Reservations _reservations = new();

    internal readonly Accommodations _accommodations = new();

#endregion

#region Constructor

    public BookingManager(ILogger<BookingManager> logger)
    {
        _logger = logger ?? throw new ArgumentNullException(nameof(logger));
        _logger.LogInformation("BookingManager initialized.");
    }

#endregion

#region Accessors for Repositories

    public Owners Owners => _owners;

    public Clients Clients => _clients;

    public Reservations Reservations => _reservations;

    public Accommodations Accommodations => _accommodations;

#endregion

#region Helper Functions

    private void SetField<T>(T fieldValue, T defaultValue, Action<T> setterAction)
    {
        // Only set the field if the value is not the default value
        if (!EqualityComparer<T>.Default.Equals(fieldValue, defaultValue))
        {
            setterAction(fieldValue); // Call the setter action with the field value
            _logger.LogInformation("Field set to {FieldValue}.", fieldValue); // Log the field change
        }
        else
        {
            _logger.LogInformation($"Field value is unchanged, remaining as default.");
        }
    }

#endregion

#region System Management

    public void SaveAll(string dataFolder)
    {
        try
        {
            // Define file paths for each repository
            string clientsFilePath = Path.Combine(dataFolder, "clients.dat");
            string accommodationsFilePath = Path.Combine(dataFolder, "accommodations.dat");
            string reservationsFilePath = Path.Combine(dataFolder, "reservations.dat");
            string ownersFilePath = Path.Combine(dataFolder, "owners.dat");

            // Save each repository to its corresponding file
            _clients.Save(clientsFilePath);
            _accommodations.Save(accommodationsFilePath);
            _reservations.Save(reservationsFilePath);
            _owners.Save(ownersFilePath);

            _logger.LogInformation("All repositories saved successfully.");
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "Error occurred while saving all repositories:");
            throw new InvalidOperationException("Error occurred while saving all repositories.", ex);
        }
    }

    public void LoadAll(string dataFolder)
    {
        try
        {
            // Define file paths for each repository
            string clientsFilePath = Path.Combine(dataFolder, "clients.dat");
            string accommodationsFilePath = Path.Combine(dataFolder, "accommodations.dat");
            string reservationsFilePath = Path.Combine(dataFolder, "reservations.dat");
            string ownersFilePath = Path.Combine(dataFolder, "owners.dat");

            // Load each repository from its corresponding file
            if (File.Exists(clientsFilePath))
                _clients.Load(clientsFilePath);
            if (File.Exists(accommodationsFilePath))
                _accommodations.Load(accommodationsFilePath);
            if (File.Exists(reservationsFilePath))
                _reservations.Load(reservationsFilePath);
            if (File.Exists(ownersFilePath))
                _owners.Load(ownersFilePath);

            _logger.LogInformation("All repositories loaded successfully.");
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "Error occurred while loading all repositories.");
            throw new InvalidOperationException("Error occurred while loading all repositories.", ex);
        }
    }

#endregion

#region Client Management

    public Client CreateBasicClient(string firstName, string lastName, string email)
    {
        try
        {
            // Log information before creating the client
            _logger.LogInformation(
                "Attempting to create a new client with Name: {FirstName} {LastName}, Email:
{Email}.", firstName,
                lastName, email);

            // Create a new client
            Client client = new(firstName, lastName, email); // May throw exception

            // Add client to the system
            _clients.Add(client);

            // Log success information
            _logger.LogInformation("Successfully created client: {FirstName} {LastName}, Email:
```

```
{Email}.", firstName,
                                   lastName, email);

            // Return the created client
            return client;
        }
        catch (ValidationException ex)
        {
            // Log the exception with details
            _logger.LogError(ex, "Error while creating client with Name: {FirstName} {LastName}, Email: {Email}.",
                             firstName, lastName, email);

            // Throw a new exception with more context
            throw new ClientCreationException("An error occurred while creating the client due to invalid input.", ex);
        }
    }

    public Client CreateCompleteClient(string firstName, string lastName, string email, string phoneNumber,
                                       string address)
    {
        try
        {
            // Log information before creating the client
            _logger.LogInformation(
                "Attempting to create a new client with Name: {FirstName} {LastName}, Email: {Email}, " +
                    "Phone Number: {PhoneNumber}, Address: {Address}.",
                firstName, lastName, email, phoneNumber, address);

            // Create a new client with the provided information
            Client client = new(firstName, lastName, email, phoneNumber, address); // May throw exception

            // Add client to the system
            _clients.Add(client);

            // Log success information
            _logger.LogInformation("Successfully created client: {FirstName} {LastName}, Email: {Email}, Phone: " +
                                       "{PhoneNumber}, Address: {Address}.",
                                   firstName, lastName, email, phoneNumber, address);

            // Return the created client
            return client;
        }
        catch (ValidationException ex)
        {
            // Log the exception with details
            _logger.LogError(ex,
                             "Error while creating client with Name: {FirstName} {LastName}, Email: {Email}, " +
                                 "Phone Number: {PhoneNumber}, Address: {Address}.",
                             firstName, lastName, email, phoneNumber, address);

            // Throw a new exception with more context
            throw new ClientCreationException("An error occurred while creating the client due to invalid input.", ex);
        }
    }

    public Client FindClientById(int clientId)
    {
        _logger.LogInformation("Attempting to find client with ID: {ClientId}.", clientId);

        var client = _clients.FindClientById(clientId);

        if (client != null)
        {
            _logger.LogInformation("Successfully found client with ID: {ClientId}.", clientId);
            return client;
        }
        else
        {
            _logger.LogError("Error finding client with ID: {ClientId}.", clientId);
            throw new ArgumentException($"Client with ID {clientId} not found.");
        }
    }

    public UpdateClientResult UpdateClient(int clientId, string? firstName = null, string? lastName = null,
                                           string? email = null, string? phoneNumber = null, string? address = null,
                                           PaymentMethod paymentMethod = PaymentMethod.Unchanged)
    {
        _logger.LogInformation("Attempting to update client with ID: {ClientId}.", clientId);

        // Find the client by ID
        var client = _clients.FindClientById(clientId);
        if (client == null)
        {
            _logger.LogWarning("Client with ID: {ClientId} not found.", clientId);
            return UpdateClientResult.ClientNotFound;
        }

        // Validate information before updating (only if not null or default value)
        if (firstName != null && !Validation.Validators.NameValidator.IsValidName(firstName))
        {
            _logger.LogError("Invalid first name provided for client ID: {ClientId}.", clientId);
            return UpdateClientResult.InvalidFirstName;
        }

        if (lastName != null && !Validation.Validators.NameValidator.IsValidName(lastName))
        {
            _logger.LogError("Invalid last name provided for client ID: {ClientId}.", clientId);
            return UpdateClientResult.InvalidLastName;
        }

        if (email != null && !Validation.Validators.EmailValidator.IsValidEmail(email))
        {
            _logger.LogError("Invalid email provided for client ID: {ClientId}.", clientId);
            return UpdateClientResult.InvalidEmail;
        }

        if (phoneNumber != null && !Validation.Validators.PhoneNumberValidator.IsValidPhoneNumber(phoneNumber))
        {
            _logger.LogError("Invalid phone number provided for client ID: {ClientId}.", clientId);
            return UpdateClientResult.InvalidPhoneNumber;
        }

        if (address != null && !Validation.Validators.AddressValidator.IsValidAddress(address))
        {
            _logger.LogError("Invalid address provided for client ID: {ClientId}.", clientId);
            return UpdateClientResult.InvalidAddress;
        }

        if (paymentMethod != PaymentMethod.Unchanged &&
            !Validation.Validators.PaymentValidator.IsValidPaymentMethod(paymentMethod))
        {
            _logger.LogError("Invalid payment method provided for client ID: {ClientId}.", clientId);
            return UpdateClientResult.InvalidPaymentMethod;
        }

        // Log success before updating
        _logger.LogInformation("Validations passed. Updating details for client ID: {ClientId}.", clientId);

        // Update client information with given fields, if not null or default
        SetField(firstName, null,
                 value =>
                 {
                     if (firstName != null && value != null)
                         client.FirstName = value;
                 });
        SetField(lastName, null,
                 value =>
                 {
                     if (lastName != null && value != null)
                         client.LastName = value;
                 });
        SetField(email, null,
                 value =>
                 {
                     if (email != null && value != null)
                         client.Email = value;
                 });
        SetField(phoneNumber, null,
                 value =>
                 {
                     if (phoneNumber != null && value != null)
                         client.PhoneNumber = value;
                 });
        SetField(address, null,
                 value =>
                 {
                     if (address != null && value != null)
                         client.Address = value;
                 });
        SetField(paymentMethod, PaymentMethod.Unchanged,
                 value =>
                 {
                    if (paymentMethod != PaymentMethod.Unchanged)
                        client.PreferredPaymentMethod = value;
                });

        _logger.LogInformation("Successfully updated client details for client ID: {ClientId}.",
clientId);

        return UpdateClientResult.Success;
    }

    public bool RemoveClient(int clientId)
    {
        _logger.LogInformation("Attempting to remove client with ID: {ClientId}.", clientId);

        // Find the client by ID
        var client = _clients.FindClientById(clientId);

        if (client == null)
        {
            _logger.LogWarning("Client with ID: {ClientId} not found. No removal performed.",
clientId);
            return false;
        }

        // Remove the client from the list
        _clients.Remove(client);
        _logger.LogInformation("Successfully removed client with ID: {ClientId}.", clientId);

        return true;
    }

#endregion

#region Owner Management

    public Owner CreateBasicOwner(string firstName, string lastName, string email)
    {
        try
        {
            // Log information before creating the owner
            _logger.LogInformation(
                "Attempting to create a new owner with Name: {FirstName} {LastName}, Email: {Email}.",
firstName,
                lastName, email);

            // Create a new owner
            Owner owner = new(firstName, lastName, email); // May throw exception

            // Add owner to the system
            _owners.Add(owner);

            // Log success information
            _logger.LogInformation("Successfully created owner: {FirstName} {LastName}, Email:
{Email}.", firstName,
                                   lastName, email);

            // Return the created owner
            return owner;
        }
        catch (ValidationException ex)
        {
            // Log the exception with details
            _logger.LogError(ex, "Error while creating owner with Name: {FirstName} {LastName}, Email:
{Email}.",
                             firstName, lastName, email);

            // Throw a new exception with more context
            throw new OwnerCreationException("An error occurred while creating the owner due to
invalid input.", ex);
        }
    }

    public Owner CreateCompleteOwner(string firstName, string lastName, string email, string
phoneNumber,
                                     string address)
    {
        try
        {
            // Log information before creating the owner
            _logger.LogInformation(
                "Attempting to create a new owner with Name: {FirstName} {LastName}, Email: {Email}, "
+
                    "Phone Number: {PhoneNumber}, Address: {Address}.",
                firstName, lastName, email, phoneNumber, address);

            // Create a new owner with the provided information
            Owner owner = new(firstName, lastName, email, phoneNumber, address); // May throw
```

```
exception

            // Add owner to the system
            _owners.Add(owner);

            // Log success information
            _logger.LogInformation("Successfully created owner: {FirstName} {LastName}, Email: {Email}.", firstName,
                                   lastName, email);

            // Return the created owner
            return owner;
        }
        catch (ValidationException ex)
        {
            // Log the exception with details
            _logger.LogError(ex,
                             "Error while creating owner with Name: {FirstName} {LastName}, Email: {Email}, " +
                                 "Phone Number: {PhoneNumber}, Address: {Address}.",
                             firstName, lastName, email, phoneNumber, address);

            // Throw a new exception with more context
            throw new OwnerCreationException("An error occurred while creating the owner due to invalid input.", ex);
        }
    }

    public Owner FindOwnerById(int ownerId)
    {
        // Log information before attempting to find the owner
        _logger.LogInformation("Attempting to find owner with ID: {OwnerId}.", ownerId);

        var owner = _owners.FindOwnerById(ownerId);

        if (owner != null)
        {
            // Log success information if the owner is found
            _logger.LogInformation("Successfully found owner with ID: {OwnerId}.", ownerId);
            return owner;
        }
        else
        {
            // Log error if the owner is not found
            _logger.LogError("Error finding owner with ID: {OwnerId}.", ownerId);
            throw new ArgumentException($"Owner with ID {ownerId} not found.");
        }
    }

    public UpdateOwnerResult UpdateOwner(int ownerId, string? firstName = null, string? lastName = null,
                                         string? email = null, string? phoneNumber = null, string? address = null)
    {
        _logger.LogInformation("Attempting to update owner with ID: {OwnerId}.", ownerId);

        // Find the owner by ID
        var owner = _owners.FindOwnerById(ownerId);
        if (owner == null)
        {
            _logger.LogWarning("Owner with ID: {OwnerId} not found.", ownerId);
            return UpdateOwnerResult.OwnerNotFound;
        }

        // Validate information before updating (only if not null)
        if (firstName != null && !Validation.Validators.NameValidator.IsValidName(firstName))
        {
            _logger.LogError("Invalid first name provided for owner ID: {OwnerId}.", ownerId);
            return UpdateOwnerResult.InvalidFirstName;
        }

        if (lastName != null && !Validation.Validators.NameValidator.IsValidName(lastName))
        {
            _logger.LogError("Invalid last name provided for owner ID: {OwnerId}.", ownerId);
            return UpdateOwnerResult.InvalidLastName;
        }

        if (email != null && !Validation.Validators.EmailValidator.IsValidEmail(email))
        {
            _logger.LogError("Invalid email provided for owner ID: {OwnerId}.", ownerId);
            return UpdateOwnerResult.InvalidEmail;
        }

        if (phoneNumber != null && !Validation.Validators.PhoneNumberValidator.IsValidPhoneNumber(phoneNumber))
        {
            _logger.LogError("Invalid phone number provided for owner ID: {OwnerId}.", ownerId);
            return UpdateOwnerResult.InvalidPhoneNumber;
        }

        if (address != null && !Validation.Validators.AddressValidator.IsValidAddress(address))
        {
            _logger.LogError("Invalid address provided for owner ID: {OwnerId}.", ownerId);
            return UpdateOwnerResult.InvalidAddress;
        }

        // Log success before updating
        _logger.LogInformation("Validations passed. Updating details for owner ID: {OwnerId}.", ownerId);

        // Update owner information with given fields, only if not null
        SetField(firstName, null,
                 value =>
                 {
                     if (firstName != null && value != null)
                         owner.FirstName = value;
                 });
        SetField(lastName, null,
                 value =>
                 {
                     if (lastName != null && value != null)
                         owner.LastName = value;
                 });
        SetField(email, null,
                 value =>
                 {
                     if (email != null && value != null)
                         owner.Email = value;
                 });
        SetField(phoneNumber, null,
                 value =>
                 {
                     if (phoneNumber != null && value != null)
                         owner.PhoneNumber = value;
                 });
        SetField(address, null,
                 value =>
                 {
                     if (address != null && value != null)
                         owner.Address = value;
                 });

        _logger.LogInformation("Successfully updated owner details for owner ID: {OwnerId}.", ownerId);

        return UpdateOwnerResult.Success;
    }

    public bool RemoveOwner(int ownerId)
    {
        _logger.LogInformation("Attempting to remove owner with ID: {OwnerId}.", ownerId);

        var owner = _owners.FindOwnerById(ownerId);
        if (owner != null)
        {
            _owners.Remove(owner);
            _logger.LogInformation("Successfully removed owner with ID: {OwnerId}.", ownerId);
            return true;
        }
        else
        {
            _logger.LogWarning("Owner with ID: {OwnerId} not found, unable to remove.", ownerId);
            return false;
        }
    }

#endregion

#region Reservation Management

    public Reservation CreateReservation(int clientId, int accommodationId, int roomId, DateTime checkIn,
                                         DateTime checkOut)
    {
        // Log the attempt to create a reservation
        _logger.LogInformation("Attempting to create reservation for client {ClientId} at accommodation " +
                                   "{AccommodationId}, room {RoomId}, from {CheckIn} to {CheckOut}.",
                               clientId, accommodationId, roomId, checkIn, checkOut);

        // Find the accommodation
        var accommodation = _accommodations.FindAccommodationById(accommodationId);
        if (accommodation == null)
        {
            _logger.LogError("Accommodation with ID {AccommodationId} not found.", accommodationId);
            throw new ArgumentException("Accommodation not found.");
        }

        // Find the room in the accommodation
        var room = accommodation.FindRoomById(roomId);
        if (room == null)
        {
            _logger.LogError("Room with ID {RoomId} not found in accommodation {AccommodationId}.",
roomId,
                             accommodationId);
            throw new ArgumentException("Room not found.");
        }

        // Check if room is available upfront for a quick exit. AddReservation will validate again to
ensure
        // consistency.
        var available = room.IsAvailable(checkIn, checkOut);
        if (!available)
        {
            _logger.LogError("Room {RoomId} in accommodation {AccommodationId} is not available for
the dates " +
                                 "{CheckIn} to {CheckOut}.",
                             roomId, accommodationId, checkIn, checkOut);
            throw new ArgumentException("Accommodation is not available for the selected dates.");
        }

        // Calculate the total cost of the reservation
        decimal totalCost;
        try
        {
            totalCost = room.CalculateTotalCost(checkIn, checkOut);
        }
        catch (ArgumentException ex)
        {
            _logger.LogError(ex, "Error calculating total cost for reservation: {Message}",
ex.Message);
            throw new TotalCostException("An error occurred while calculating the total cost for the
reservation.", ex);
        }

        // Log the total cost calculation
        _logger.LogInformation(
            "Calculated total cost for reservation: {TotalCost} for {ClientId} from {CheckIn} to
{CheckOut}.",
            totalCost, clientId, checkIn, checkOut);

        // Create the reservation
        Reservation reservation;
        try
        {
            reservation =
                new Reservation(clientId, accommodationId, roomId, accommodation.Type, checkIn,
checkOut, totalCost);
        }
        catch (ValidationException ex)
        {
            _logger.LogError(ex,
                             "Validation failed when creating reservation for client {ClientId} at
accommodation " +
                                 "{AccommodationId}, room {RoomId}. Error: {ErrorCode}",
                             clientId, accommodationId, roomId, ex.ErrorCode);
            throw new ReservationCreationException(
                $"An error occurred while creating the reservation due to invalid input.", ex);
        }

        // Attempt to add the reservation to the room (this also checks for availability)
        bool success = room.AddReservation(checkIn, checkOut);
        if (!success)
        {
            _logger.LogError("Room {RoomId} in accommodation {AccommodationId} is not available for
the dates " +
                                 "{CheckIn} to {CheckOut}.",
                             roomId, accommodationId, checkIn, checkOut);
            throw new ArgumentException("Accommodation is not available for the selected dates.");
        }

        // Add the reservation to the reservation list
        bool reservationAdded = _reservations.Add(reservation);
        if (!reservationAdded)
        {
            _logger.LogError("Failed to add reservation for client {ClientId} at accommodation " +
                                 "{AccommodationId}, room {RoomId}. Reservation may already exist.",
                             clientId, accommodationId, roomId);
            throw new ArgumentException("Failed to add reservation.");
        }

        // Log successful reservation creation
        _logger.LogInformation("Successfully created reservation {ReservationId} for client {ClientId}
at " +
                                   "accommodation {AccommodationId}, room {RoomId}. Total cost:
{TotalCost}.",
                               reservation.Id, clientId, accommodationId, roomId, totalCost);

        // Return the newly created reservation
        return reservation;
    }

    public UpdateReservationResult UpdateReservation(int reservationId, DateTime? newCheckIn = null,
                                                     DateTime? newCheckOut = null)
    {
        _logger.LogInformation("Attempting to update reservation {ReservationId}.", reservationId);

        try
        {
            // Use helper to find the reservation
            var reservation = FindReservation(reservationId);

            // Use helper to find accommodation and room
            var (accommodation, room) = FindAssociatedEntities(reservation);

            // Determine effective check-in and check-out dates
            var (effectiveCheckIn, effectiveCheckOut) = GetEffectiveDates(reservation, newCheckIn,
newCheckOut);

            _logger.LogInformation(
                "Effective dates for reservation {ReservationId}: Check-In: {CheckIn}, Check-Out:
{CheckOut}.",
                reservationId, effectiveCheckIn, effectiveCheckOut);

            // Check availability using helper
            if (!IsAvailableForUpdate(room, reservation.CheckInDate, reservation.CheckOutDate,
effectiveCheckIn,
                                      effectiveCheckOut))
            {
                _logger.LogWarning(
                    "Room {RoomId} in accommodation {AccommodationId} is unavailable for the new
dates: " +
                        "Check-In: {CheckIn}, Check-Out: {CheckOut}.",
                    room.Id, accommodation.Id, effectiveCheckIn, effectiveCheckOut);
                return UpdateReservationResult.DatesUnavailable;
            }

            // Update room reservation (handles removing and adding)
            UpdateRoomReservation(room, reservation.CheckInDate, reservation.CheckOutDate,
effectiveCheckIn,
                                  effectiveCheckOut);

            // Update reservation dates
            reservation.CheckInDate = effectiveCheckIn;
            reservation.CheckOutDate = effectiveCheckOut;

            _logger.LogInformation("Successfully updated reservation {ReservationId}.",
reservationId);
            return UpdateReservationResult.Success;
        }
        catch (EntityNotFoundException ex) when (ex.EntityType == nameof(Reservation))
        {
            _logger.LogWarning(ex, "Reservation with ID {ReservationId} not found.", reservationId);
            return UpdateReservationResult.ReservationNotFound;
        }
        catch (EntityNotFoundException ex) when (ex.EntityType == nameof(Accommodation))
        {
            _logger.LogWarning(ex, "Accommodation for reservation {ReservationId} not found.",
reservationId);
            return UpdateReservationResult.AccommodationNotFound;
        }
        catch (EntityNotFoundException ex) when (ex.EntityType == nameof(Room))
        {
            _logger.LogWarning(ex, "Room for reservation {ReservationId} not found.", reservationId);
            return UpdateReservationResult.RoomNotFound;
        }
        catch (ArgumentNullException ex)
        {
            _logger.LogWarning(ex, "Room was null for reservation {ReservationId}.", reservationId);
            return UpdateReservationResult.RoomNotFound;
        }
        catch (ArgumentException ex)
        {
            _logger.LogWarning(ex, "Invalid check-in/check-out dates for reservation
{ReservationId}.", reservationId);
            return UpdateReservationResult.InvalidDates;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "An error occurred while updating reservation {ReservationId}.", reservationId);
            return UpdateReservationResult.Error;
        }
    }

    public CancellationResult CancelReservation(int reservationId)
    {
        _logger.LogInformation("Attempting to cancel reservation {ReservationId}.", reservationId);

        try
        {
            // Use helper to find the reservation
            var reservation = FindReservation(reservationId);

            // Use helper to find the associated accommodation and room
            var (accommodation, room) = FindAssociatedEntities(reservation);

            // Attempt to remove the reservation from the room's reserved dates
            bool removeResult = room.RemoveReservation(reservation.CheckInDate, reservation.CheckOutDate);
            if (!removeResult)
            {
                _logger.LogError("Failed to remove reservation {ReservationId} from room {RoomId} in accommodation " +
                                     "{AccommodationId}.",
                                 reservationId, room.Id, accommodation.Id);
                return CancellationResult.Error;
            }

            // Mark the reservation as cancelled
            reservation.Status = ReservationStatus.Cancelled;
            _logger.LogInformation("Successfully cancelled reservation {ReservationId}.", reservationId);

            return CancellationResult.Success;
        }
        catch (EntityNotFoundException ex) when (ex.EntityType == nameof(Reservation))
        {
            _logger.LogWarning(ex, "Reservation with ID {ReservationId} not found.", reservationId);
            return CancellationResult.ReservationNotFound;
        }
        catch (EntityNotFoundException ex) when (ex.EntityType == nameof(Accommodation))
        {
            _logger.LogWarning(ex, "Accommodation for reservation {ReservationId} not found.", reservationId);
            return CancellationResult.AccommodationNotFound;
        }
        catch (EntityNotFoundException ex) when (ex.EntityType == nameof(Room))
        {
            _logger.LogWarning(ex, "Room for reservation {ReservationId} not found.", reservationId);
            return CancellationResult.RoomNotFound;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "An error occurred while cancelling reservation {ReservationId}.", reservationId);
            return CancellationResult.Error;
        }
    }

#region Reservation Helper Functions

    private bool IsAvailableForUpdate(Room room, DateTime currentStart, DateTime currentEnd, DateTime newStart,
                                      DateTime newEnd)
    {
        if (room == null)
            throw new ArgumentNullException(nameof(room), "Room cannot be null.");

        if (newEnd <= newStart)
            throw new ArgumentException("New check-out date must be after the new check-in date.");

        DateRange existingReservation = new(currentStart, currentEnd);

        _logger?.LogDebug("Checking availability for update: Current [{CurrentStart} - {CurrentEnd}], Proposed " +
                              "[{NewStart} - {NewEnd}].",
                          currentStart, currentEnd, newStart, newEnd);

        // Exclude the current reservation's dates from the availability check
        return room.IsAvailable(newStart, newEnd, existingReservation);
    }

    private Reservation FindReservation(int reservationId)
    {
        _logger.LogInformation("Looking for reservation with ID {ReservationId}.", reservationId);

        var reservation = _reservations.FindReservationById(reservationId);
        if (reservation == null)
        {
            _logger.LogWarning("Reservation with ID {ReservationId} not found.", reservationId);
            throw new EntityNotFoundException(nameof(Reservation), reservationId);
        }

        _logger.LogInformation("Successfully found reservation with ID {ReservationId}.", reservationId);
        return reservation;
    }

    private (Accommodation, Room) FindAssociatedEntities(Reservation reservation)
    {
        _logger.LogInformation("Finding associated accommodation and room for reservation ID {ReservationId}.",
                               reservation.Id);

        // Find accommodation
        var accommodation = _accommodations.FindAccommodationById(reservation.AccommodationId);
        if (accommodation == null)
        {
            _logger.LogWarning("Accommodation with ID {AccommodationId} not found for reservation {ReservationId}.",
                               reservation.AccommodationId, reservation.Id);
            throw new EntityNotFoundException(nameof(Accommodation), reservation.AccommodationId);
        }

        _logger.LogInformation(
            "Successfully found accommodation with ID {AccommodationId} for reservation {ReservationId}.",
            accommodation.Id, reservation.Id);

        // Find room
        var room = accommodation.FindRoomById(reservation.RoomId);
        if (room == null)
        {
            _logger.LogWarning(
                "Room with ID {RoomId} not found in accommodation {AccommodationId} for reservation {ReservationId}.",
                reservation.RoomId, accommodation.Id, reservation.Id);
            throw new EntityNotFoundException(nameof(Room), reservation.RoomId);
        }

        _logger.LogInformation("Successfully found room with ID {RoomId} in accommodation {AccommodationId} for " +
                                   "reservation {ReservationId}.",
                               room.Id, accommodation.Id, reservation.Id);

        return (accommodation, room);
    }

    private (DateTime, DateTime) GetEffectiveDates(Reservation reservation, DateTime? newCheckIn, DateTime? newCheckOut)
    {
        _logger.LogInformation("Determining effective dates for reservation ID {ReservationId}.", reservation.Id);

        DateTime effectiveCheckIn = newCheckIn ?? reservation.CheckInDate;
        DateTime effectiveCheckOut = newCheckOut ?? reservation.CheckOutDate;

        if (effectiveCheckOut <= effectiveCheckIn)
        {
            _logger.LogError(
                "Invalid dates for reservation {ReservationId}: Check-In: {CheckIn}, Check-Out: {CheckOut}.",
                reservation.Id, effectiveCheckIn, effectiveCheckOut);
            throw new ArgumentException("Check-out date must be later than check-in date.");
        }

        _logger.LogInformation(
            "Effective dates for reservation {ReservationId}: Check-In: {CheckIn}, Check-Out: {CheckOut}.",
            reservation.Id, effectiveCheckIn, effectiveCheckOut);

        return (effectiveCheckIn, effectiveCheckOut);
    }

    private void UpdateRoomReservation(Room room, DateTime currentStart, DateTime currentEnd, DateTime newStart,
                                       DateTime newEnd)
    {
        _logger.LogInformation(
            "Updating room reservation for Room {RoomId}: Removing dates {CurrentStart} to {CurrentEnd}, " +
                "and adding dates {NewStart} to {NewEnd}.",
            room.Id, currentStart, currentEnd, newStart, newEnd);

        // Safely remove current reservation dates
        room.RemoveReservation(currentStart, currentEnd);

        // Add the new reservation dates
        room.AddReservation(newStart, newEnd);

        _logger.LogInformation(
            "Successfully updated room reservation for Room {RoomId} with new dates {NewStart} to {NewEnd}.", room.Id,
            newStart, newEnd);
    }

#endregion

#endregion

#region Accommodation Management

    public Accommodation CreateAccommodation(int ownerId, AccommodationType type, string name, string address)
    {
        try
        {
            // Log information before creating the accommodation
            _logger.LogInformation("Attempting to create a new accommodation for owner ID {OwnerId}: Type: {Type}, " +
                                       "Name: {Name}, Address: {Address}.",
                                   ownerId, type, name, address);

            // Find the owner by ID
            var owner = _owners.FindOwnerById(ownerId);
            if (owner == null)
            {
                _logger.LogError("Owner with ID {OwnerId} not found.", ownerId);
                throw new EntityNotFoundException(nameof(Owner), ownerId);
            }

            // Create a new accommodation
            var accommodation = new Accommodation(ownerId, type, name, address); // May throw exception

            // Attempt to add the accommodation to the owner's list of accommodations
            bool addSuccess = owner.AddAccommodation(accommodation);
            if (!addSuccess)
            {
                _logger.LogError("Failed to add accommodation {Name} to owner ID {OwnerId}.", name, ownerId);
                throw new OwnerAddAccommodationException("Failed to add accommodation to owner's list.");
            }

            // Attempt to add the accommodation to the system
            bool addToSystemSuccess = _accommodations.Add(accommodation);
            if (!addToSystemSuccess)
            {
                _logger.LogError("Failed to add accommodation {Name} to the system.", name);
                throw new AddAccommodationSystemException(
                    "An error occurred while adding the accommodation to the system.");
            }

            // Log success information
            _logger.LogInformation(
                "Successfully created accommodation: {Type}, {Name}, Address: {Address} for owner ID {OwnerId}.", type,
                name, address, ownerId);

            // Return the created accommodation
            return accommodation;
        }
        catch (EntityNotFoundException ex)
        {
            // Log the exception with details
            _logger.LogError(ex,
                             "Error while creating accommodation for owner ID {OwnerId}: Type: {Type}, Name: {Name}, " +
                                 "Address: {Address}.",
                             ownerId, type, name, address);

            // Rethrow the exception as a more specific one
            throw new AccommodationCreationException(
                "An error occurred while creating the accommodation due to missing owner.", ex);
        }
        catch (OwnerAddAccommodationException ex)
        {
            // Log the exception if adding accommodation fails
            _logger.LogError(ex, "Error while adding accommodation {Name} for owner ID {OwnerId}.",
name, ownerId);
            throw new AccommodationCreationException(
                "An error occurred while trying to add accommodation to the owner accommodation
list.");
        }
        catch (AddAccommodationSystemException ex)
        {
            // Log the exception if adding accommodation fails
            _logger.LogError(ex, "Error while adding accommodation {Name} to the system.", name);
            throw new AccommodationCreationException(
                "An error occurred while trying to add accommodation to the system.");
        }
    }

    public UpdateAccommodationResult UpdateAccommodation(int accommodationId,
                                                         AccommodationType type =
AccommodationType.None,
                                                         string? name = null, string? address = null)
    {
        _logger.LogInformation("Attempting to update accommodation with ID: {AccommodationId}.",
accommodationId);

        // Find the accommodation by ID
        var accommodation = _accommodations.FindAccommodationById(accommodationId);
        if (accommodation == null)
        {
            _logger.LogWarning("Accommodation with ID: {AccommodationId} not found.",
accommodationId);
            return UpdateAccommodationResult.AccommodationNotFound;
        }

        // Validate information before updating (only if not default value)
        if (type != AccommodationType.None &&
            !Validation.Validators.AccommodationValidator.IsValidAccommodationType(type))
        {
            _logger.LogError("Invalid type provided for accommodation ID: {AccommodationId}.",
accommodationId);
            return UpdateAccommodationResult.InvalidType;
        }

        if (name != null && !Validation.Validators.NameValidator.IsValidAccommodationName(name))
        {
            _logger.LogError("Invalid name provided for accommodation ID: {AccommodationId}.",
accommodationId);
            return UpdateAccommodationResult.InvalidName;
        }

        if (address != null && !Validation.Validators.AddressValidator.IsValidAddress(address))
        {
            _logger.LogError("Invalid address provided for accommodation ID: {AccommodationId}.",
accommodationId);
            return UpdateAccommodationResult.InvalidAddress;
        }

        // Log success before updating
        _logger.LogInformation("Validations passed. Updating details for accommodation ID:
{AccommodationId}.",
                               accommodationId);

        // Update accommodation information with given fields, only if not default
        SetField(type, AccommodationType.None,
                 value =>
                 {
                     if (type != AccommodationType.None)
                         accommodation.Type = value;
                 });
        SetField(name, null,
                 value =>
                 {
                     if (name != null && value != null)
                         accommodation.Name = value;
                 });
        SetField(address, null,
                 value =>
                 {
                     if (address != null && value != null)
                         accommodation.Address = value;
                 });

        // Log success after updating
        _logger.LogInformation("Successfully updated accommodation details for accommodation ID:
{AccommodationId}.",
                               accommodationId);

        return UpdateAccommodationResult.Success;
    }

    public RemoveAccommodationResult RemoveAccommodation(int accommodationId)
    {
        _logger.LogInformation("Attempting to remove accommodation with ID: {AccommodationId}.",
accommodationId);

        try
        {
            // Find the accommodation by ID
            var accommodation = _accommodations.FindAccommodationById(accommodationId);
            if (accommodation == null)
            {
                _logger.LogWarning("Accommodation with ID: {AccommodationId} not found, unable to
remove.",
                                   accommodationId);
                return RemoveAccommodationResult.AccommodationNotFound;
            }

            // Find the owner by ID
            var owner = _owners.FindOwnerById(accommodation.OwnerId);
            if (owner == null)
            {
                _logger.LogWarning("Owner with ID: {OwnerId} not found for accommodation ID:
{AccommodationId}.",
                                   accommodation.OwnerId, accommodationId);
                return RemoveAccommodationResult.OwnerNotFound;
            }

            // Log before removal
            _logger.LogInformation("Owner with ID: {OwnerId} found. Removing accommodation ID:
{AccommodationId}.",
                                   accommodation.OwnerId, accommodationId);

            // Attempt to remove accommodation from the system
            bool removeSuccess = _accommodations.Remove(accommodation);
            if (!removeSuccess)
            {
                _logger.LogError("Failed to remove accommodation ID: {AccommodationId} from the
system.",
                                 accommodationId);
                return RemoveAccommodationResult.AccommodationRemovalFailed;
            }

            // Attempt to remove accommodation from the owner's list
            bool ownerRemoveSuccess = owner.RemoveAccommodation(accommodation);
            if (!ownerRemoveSuccess)
            {
                _logger.LogError("Failed to disassociate accommodation ID: {AccommodationId} from
owner ID: {OwnerId}.",
                                 accommodationId, accommodation.OwnerId);
                return RemoveAccommodationResult.AccommodationDisassociationFailed;
            }

            // Log success after removal
            _logger.LogInformation("Successfully removed accommodation ID: {AccommodationId} and
disassociated it " +
                                       "from owner ID: {OwnerId}.",
                                   accommodationId, accommodation.OwnerId);

            return RemoveAccommodationResult.Success;
        }
        catch (Exception ex)
        {
            // Log any unexpected errors
            _logger.LogError(ex, "Unexpected error occurred while removing accommodation with ID:
{AccommodationId}.",
                             accommodationId);
            return RemoveAccommodationResult.Error;
        }
    }

#endregion
}
}
```

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

Go to the documentation of this file.

```
using ProtoBuf;

namespace SmartStay.Core.Utilities
{
[ProtoContract]
public class DateRange : IComparable<DateRange>
{
    [ProtoMember(1)]
    public DateTime Start { get; set; }

    [ProtoMember(2)]
    public DateTime End {
        get; set;
    }

    public DateRange(DateTime start, DateTime end)
    {
        Start = start;
        End = end;
    }

    public int CompareTo(DateRange? other)
    {
        if (other is null)
            return 1; // If other is null, this object is considered greater.

        // First, compare by the Start date.
        int startComparison = Start.CompareTo(other.Start);
        if (startComparison != 0)
            return startComparison;

        // If Start dates are the same, compare by End date.
        return End.CompareTo(other.End);
    }

    public override bool Equals(object? obj)
    {
        if (obj is DateRange other)
        {
            // Check if both Start and End dates are equal.
            return Start == other.Start && End == other.End;
        }
        return false;
    }

    public override int GetHashCode()
    {
        // Combine the hash codes of Start and End dates to get a unique hash code.
        return HashCode.Combine(Start, End);
    }

    public static bool operator ==(DateRange left, DateRange right)
    {
        return left.Equals(right);
    }

    public static bool operator !=(DateRange left, DateRange right)
    {
        return !(left == right);
    }

    public static bool operator<(DateRange left, DateRange right)
    {
        return left.CompareTo(right) < 0;
    }

    public static bool operator <=(DateRange left, DateRange right)
    {
        return left.CompareTo(right) <= 0;
    }

    public static bool operator>(DateRange left, DateRange right)
    {
        return left.CompareTo(right) > 0;
    }

    public static bool operator >=(DateRange left, DateRange right)
    {
        return left.CompareTo(right) >= 0;
    }

    public DateRange Clone()
    {
        // Use the constructor to initialize the new DateRange
        return new DateRange(Start, End);
    }
}
}
```

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.

```

using System.Collections.Generic;
using System.Text.Json;

namespace SmartStay.Core.Utilities
{
public static class JsonHelper
{
    private static readonly JsonSerializerOptions _jsonOptions = new() { WriteIndented = true };

    public static string SerializeToJson<T>(IEnumerable<T> items)
    {
        return JsonSerializer.Serialize(items, _jsonOptions);
    }

    public static List<T> DeserializeFromJson<T>(string json)
    {
        return JsonSerializer.Deserialize<List<T»(json) ?? new List<T>();
    }
}
}
```

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.

```
namespace SmartStay.IO.Extensions
{
public static class FileExtensions
{
    public static void EnsureDirectoryExists(this string filePath)
    {
        var directory = Path.GetDirectoryName(filePath);
        if (!string.IsNullOrEmpty(directory) && !Directory.Exists(directory))
        {
            Directory.CreateDirectory(directory);
        }
    }
}
}
```

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.

```
namespace SmartStay.IO.FileOperations
{
public static class FileHandler
{
    public static string ReadFile(string filePath)
    {
        if (string.IsNullOrWhiteSpace(filePath))
            throw new ArgumentException("File path cannot be null or empty.");

        if (!File.Exists(filePath))
            throw new FileNotFoundException($"File not found: {filePath}");

        return File.ReadAllText(filePath);
    }

    public static void WriteFile(string filePath, string content)
    {
        if (string.IsNullOrWhiteSpace(filePath))
            throw new ArgumentException("File path cannot be null or empty.");

        var directory = Path.GetDirectoryName(filePath);
        if (!string.IsNullOrEmpty(directory) && !Directory.Exists(directory))
        {
            Directory.CreateDirectory(directory);
        }

        File.WriteAllText(filePath, content);
    }
}
}
```

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.

```
namespace SmartStay.IO.FileOperations
{
public static class PathValidator
{
    public static bool FileExists(string filePath)
    {
        return File.Exists(filePath);
    }

    public static bool IsValidFileType(string filePath, string extension)
    {
        if (string.IsNullOrWhiteSpace(filePath))
            throw new ArgumentException("File path cannot be null or empty.");

        return Path.GetExtension(filePath).Equals(extension, StringComparison.OrdinalIgnoreCase);
    }
}
}
```

7.93 SmartStay.IO.AssemblyInfo.cs File Reference

7.94 SmartStay.IO.AssemblyInfo.cs

Go to the documentation of this file.

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:4.0.30319.42000
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

using System;
using System.Reflection;

[assembly: System.Reflection.AssemblyCompanyAttribute("SmartStay.IO")]
[assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]
[assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]
[assembly:
System.Reflection.AssemblyInformationalVersionAttribute("1.0.0+c366ac03947932e5126b804e73253b4d5f5e0e8d")]
[assembly: System.Reflection.AssemblyProductAttribute("SmartStay.IO")]
[assembly: System.Reflection.AssemblyTitleAttribute("SmartStay.IO")]
[assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]

// Generated by the MSBuild WriteCodeFragment class.

```

7.95 SmartStay.IO.GlobalUsings.g.cs File Reference

7.96 SmartStay.IO.GlobalUsings.g.cs

Go to the documentation of this file.

```
// <auto-generated/>
global using global::System;
global using global::System.Collections.Generic;
global using global::System.IO;
global using global::System.Linq;
global using global::System.Net.Http;
global using global::System.Threading;
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.

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:4.0.30319.42000
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

using System;
using System.Reflection;

[assembly: System.Reflection.AssemblyCompanyAttribute("SmartStay.Validation")]
[assembly: System.Reflection.AssemblyConfigurationAttribute("Debug")]
```

```
[assembly: System.Reflection.AssemblyFileVersionAttribute("1.0.0.0")]
[assembly:
System.Reflection.AssemblyInformationalVersionAttribute("1.0.0+c366ac03947932e5126b804e73253b4d5f5e0e8d")]
[assembly: System.Reflection.AssemblyProductAttribute("SmartStay.Validation")]
[assembly: System.Reflection.AssemblyTitleAttribute("SmartStay.Validation")]
[assembly: System.Reflection.AssemblyVersionAttribute("1.0.0.0")]

// Generated by the MSBuild WriteCodeFragment class.

```

7.99 SmartStay.Validation.GlobalUsings.g.cs File Reference

7.100 SmartStay.Validation.GlobalUsings.g.cs

Go to the documentation of this file.

```
// <auto-generated/>
global using global::System;
global using global::System.Collections.Generic;
global using global::System.IO;
global using global::System.Linq;
global using global::System.Net.Http;
global using global::System.Threading;
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.

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:4.0.30319.42000
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace SmartStay.Validation.Resources {
    using System;


    // This class was auto-generated by the StronglyTypedResourceBuilder
    // class via a tool like ResGen or Visual Studio.
    // To add or remove a member, edit your .ResX file then rerun ResGen
    // with the /str option, or rebuild your VS project.

[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyTypedResourceBuilder",
"17.0.0.0")]
    [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
    [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
    internal class ValidationMessages {

        private static global::System.Resources.ResourceManager resourceMan;

        private static global::System.Globalization.CultureInfo resourceCulture;

        [global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1811:AvoidUncalledPrivateCode")]
        internal ValidationMessages() {
        }


[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
        internal static global::System.Resources.ResourceManager ResourceManager {
            get {
                if (object.ReferenceEquals(resourceMan, null)) {
                    global::System.Resources.ResourceManager temp = new
global::System.Resources.ResourceManager("SmartStay.Validation.Resources.ValidationMessages",
typeof(ValidationMessages).Assembly);
                    resourceMan = temp;
                }
                return resourceMan;
            }
        }


[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
        internal static global::System.Globalization.CultureInfo Culture {
            get {
                return resourceCulture;
            }
            set {
                resourceCulture = value;
            }
        }

        internal static string InvalidAccommodationName {
            get {
                return ResourceManager.GetString("InvalidAccommodationName", resourceCulture);
            }
        }

        internal static string InvalidAccommodationType {
            get {
                return ResourceManager.GetString("InvalidAccommodationType", resourceCulture);
            }
        }

        internal static string InvalidAddress {
            get {
                return ResourceManager.GetString("InvalidAddress", resourceCulture);
            }
        }

        internal static string InvalidAvailabilityStatus {
            get {
                return ResourceManager.GetString("InvalidAvailabilityStatus", resourceCulture);
            }
        }

        internal static string InvalidDate {
            get {
                return ResourceManager.GetString("InvalidDate", resourceCulture);
            }
        }

        internal static string InvalidDateRange {
            get {
                return ResourceManager.GetString("InvalidDateRange", resourceCulture);
            }
        }

        internal static string InvalidEmail {
            get {
                return ResourceManager.GetString("InvalidEmail", resourceCulture);
            }
        }

        internal static string InvalidId {
            get {
                return ResourceManager.GetString("InvalidId", resourceCulture);
            }
        }

        internal static string InvalidName {
            get {
                return ResourceManager.GetString("InvalidName", resourceCulture);
            }
        }

        internal static string InvalidPaymentMethod {
            get {
                return ResourceManager.GetString("InvalidPaymentMethod", resourceCulture);
            }
        }

        internal static string InvalidPaymentStatus {
            get {
                return ResourceManager.GetString("InvalidPaymentStatus", resourceCulture);
            }
        }

        internal static string InvalidPaymentValue {
            get {
                return ResourceManager.GetString("InvalidPaymentValue", resourceCulture);
            }
        }

        internal static string InvalidPhoneNumber {
            get {
                return ResourceManager.GetString("InvalidPhoneNumber", resourceCulture);
            }
        }

        internal static string InvalidPrice {
            get {
                return ResourceManager.GetString("InvalidPrice", resourceCulture);
            }
        }

        internal static string InvalidReservationStatus {
            get {
                return ResourceManager.GetString("InvalidReservationStatus", resourceCulture);
            }
        }

        internal static string InvalidRoomType {
            get {
                return ResourceManager.GetString("InvalidRoomType", resourceCulture);
            }
        }

        internal static string InvalidTotalCost {
            get {
                return ResourceManager.GetString("InvalidTotalCost", resourceCulture);
            }
        }
    }
}
```

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.

```


namespace SmartStay.Validation
{
public 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
}
}
```

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.

```
using System.Globalization;
using System.Resources;

namespace SmartStay.Validation
{
public static class ValidationErrorMessages
{
    private static readonly ResourceManager _resourceManager = new ResourceManager(
        "SmartStay.Validation.Resources.ValidationMessages",
typeof(ValidationErrorMessages).Assembly);

    public static string GetErrorMessage(ValidationErrorCode errorCode)
    {
        // Get the string based on the current culture
        return _resourceManager.GetString(errorCode.ToString(), CultureInfo.CurrentCulture) ??
               "Unknown validation error.";
    }
}
}
```

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.

```

namespace SmartStay.Validation
{
public class ValidationException : Exception
{
    public ValidationErrorCode ErrorCode { get; }

    public ValidationException(ValidationErrorCode errorCode) :
base(ValidationErrorMessages.GetErrorMessage(errorCode))
    {
        ErrorCode = errorCode;
    }
}
}
```

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.

```
using SmartStay.Common.Enums;

namespace SmartStay.Validation.Validators
{
public static class AccommodationValidator
{
    public static AccommodationType ValidateAccommodationType(AccommodationType accommodationType)
    {
        if (!IsValidAccommodationType(accommodationType))
        {
            throw new ValidationException(ValidationErrorCode.InvalidAccommodationType);
        }
        return accommodationType;
    }

    public static bool IsValidAccommodationType(AccommodationType accommodationType)
    {
        return Enum.IsDefined(typeof(AccommodationType), accommodationType);
    }

    public static int ValidateAccommodationId(int id)
    {
        if (!IsValidAccommodationId(id))
        {
            throw new ValidationException(ValidationErrorCode.InvalidId);
        }
        return id;
    }

    public static bool IsValidAccommodationId(int id) => id > 0;
}
}
```

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 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.

7.112 AddressValidator.cs

Go to the documentation of this file.

```
namespace SmartStay.Validation.Validators
{
  public static class AddressValidator
{
    public static string ValidateAddress(string address)
    {
        if (!IsValidAddress(address))
        {
            throw new ValidationException(ValidationErrorCode.InvalidAddress);
        }
        return address;
    }

    public static bool IsValidAddress(string address) => !string.IsNullOrWhiteSpace(address);
}
}
```

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 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.

7.114 ClientValidator.cs

Go to the documentation of this file.

```

namespace SmartStay.Validation.Validators
{
public static class ClientValidator
{
    public static int ValidateClientId(int id)
    {
        if (!IsValidClientId(id))
        {
            throw new ValidationException(ValidationErrorCode.InvalidId);
        }
        return id;
    }

    public static bool IsValidClientId(int id) => id > 0;
}
}
```

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.

```

namespace SmartStay.Validation.Validators
{
public static class DateValidator
{
    public static DateTime ValidateCheckInDate(DateTime checkInDate)
    {
        if (!IsValidFutureDate(checkInDate))
        {
            throw new ValidationException(ValidationErrorCode.InvalidDate);
        }
        return checkInDate;
    }

    public static DateTime ValidateCheckOutDate(DateTime checkOutDate, DateTime checkInDate)
    {
        if (!IsValidDateRange(checkInDate, checkOutDate))
        {
            throw new ValidationException(ValidationErrorCode.InvalidDateRange);
        }
        return checkOutDate;
    }

    public static bool IsValidFutureDate(DateTime date)
    {
        return date >= DateTime.Today;
    }

    public static bool IsValidDateRange(DateTime checkInDate, DateTime checkOutDate)
    {
        return checkInDate < checkOutDate;
    }
}
}
```

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.

```

using System.Text.RegularExpressions;

namespace SmartStay.Validation.Validators
{
public static class EmailValidator
{
    // TODO: add a way to check if an email is already being used

    static readonly string EmailPattern = @"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$";

    public static string ValidateEmail(string email)
    {
        if (!IsValidEmail(email))
        {
            throw new ValidationException(ValidationErrorCode.InvalidEmail);
        }
        return email;
    }

    public static bool IsValidEmail(string email) => !string.IsNullOrWhiteSpace(email) &&
Regex.IsMatch(email,

EmailPattern);
}
}
```

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.

```
namespace SmartStay.Validation.Validators
{
  public static class NameValidator
{
    public static string ValidateName(string name)
    {
        if (!IsValidName(name))
        {
            throw new ValidationException(ValidationErrorCode.InvalidName);
        }
        return name;
    }

    public static bool IsValidName(string name) => !string.IsNullOrWhiteSpace(name) && name.Length <=
50;

    public static string ValidateAccommodationName(string name)
    {
        if (!IsValidAccommodationName(name))
        {
            throw new ValidationException(ValidationErrorCode.InvalidAccommodationName);
        }
        return name;
    }

    public static bool IsValidAccommodationName(string name) => !string.IsNullOrWhiteSpace(name) &&
name.Length <= 100;
}
}
```

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.

```
namespace SmartStay.Validation.Validators
{
public static class OwnerValidator
{
    public static int ValidateOwnerId(int id)
    {
        if (!IsValidOwnerId(id))
        {
            throw new ValidationException(ValidationErrorCode.InvalidId);
        }
        return id;
    }

    public static bool IsValidOwnerId(int id) => id > 0;

    public static string ValidateOwnerName(string name)
    {
        if (string.IsNullOrWhiteSpace(name) || name.Length < 2)
        {
            throw new ValidationException(ValidationErrorCode.InvalidName);
        }
        return name;
    }

    public static string ValidateOwnerEmail(string email)
    {
        if (string.IsNullOrWhiteSpace(email) || !email.Contains('@') || !email.Contains('.'))
        {
            throw new ValidationException(ValidationErrorCode.InvalidEmail);
        }
        return email;
    }

    public static string ValidateOwnerPhoneNumber(string phoneNumber)
    {
        if (string.IsNullOrWhiteSpace(phoneNumber) || phoneNumber.Length < 10)
        {
            throw new ValidationException(ValidationErrorCode.InvalidPhoneNumber);
        }
        return phoneNumber;
    }
}
}
```

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.

```
using SmartStay.Common.Enums;

namespace SmartStay.Validation.Validators
{
public static class PaymentValidator
{
    public static decimal ValidatePrice(decimal price)
    {
        if (!IsValidPrice(price))
        {
            throw new ValidationException(ValidationErrorCode.InvalidPrice);
        }
        return price;
    }

    public static bool IsValidPrice(decimal price) => price > 0;

    public static decimal ValidateTotalCost(decimal totalCost)
    {
        if (!IsValidTotalCost(totalCost))
        {
            throw new ValidationException(ValidationErrorCode.InvalidTotalCost);
        }
        return totalCost;
    }

    public static bool IsValidTotalCost(decimal totalCost) => totalCost > 0;

    public static decimal ValidatePaymentAmount(decimal amount)
    {
        if (!IsValidPaymentAmount(amount))
        {
            throw new ValidationException(ValidationErrorCode.InvalidPaymentValue);
        }
        return amount;
    }

    public static bool IsValidPaymentAmount(decimal amount) => amount > 0;

    public static PaymentStatus ValidatePaymentStatus(PaymentStatus status)
    {
        if (!IsValidPaymentStatus(status))
        {
            throw new ValidationException(ValidationErrorCode.InvalidPaymentStatus);
        }
        return status;
    }

    public static bool IsValidPaymentStatus(PaymentStatus paymentStatus) =>
Enum.IsDefined(typeof(PaymentStatus),

paymentStatus);

    public static PaymentMethod ValidatePaymentMethod(PaymentMethod paymentMethod)
    {
        if (!IsValidPaymentMethod(paymentMethod))
        {
            throw new ValidationException(ValidationErrorCode.InvalidPaymentMethod);
        }
        return paymentMethod;
    }

    public static bool IsValidPaymentMethod(PaymentMethod paymentMethod) =>
Enum.IsDefined(typeof(PaymentMethod),

paymentMethod);

    public static decimal ValidatePayment(decimal paymentValue)
    {
        if (paymentValue < 0)
        {
            throw new ValidationException(ValidationErrorCode.InvalidPaymentValue);
        }
        return paymentValue;
    }
}
}
```

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.

```

using System.Text.RegularExpressions;

namespace SmartStay.Validation.Validators
{
public static class PhoneNumberValidator
{
    private static readonly string PhoneNumberPattern = @"^\+(\d{1,3})\d{7,15}$";

    public static string ValidatePhoneNumber(string phoneNumber)
    {
        if (!IsValidPhoneNumber(phoneNumber))
        {
            throw new ValidationException(ValidationErrorCode.InvalidPhoneNumber);
        }
        return phoneNumber;
    }

    public static bool IsValidPhoneNumber(string phoneNumber) =>
        !string.IsNullOrWhiteSpace(phoneNumber) && Regex.IsMatch(phoneNumber, PhoneNumberPattern);
}
}
```

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.

```
using SmartStay.Common.Enums;

namespace SmartStay.Validation.Validators
{
public static class ReservationValidator
{
    public static ReservationStatus ValidateReservationStatus(ReservationStatus status)
    {
        if (!IsValidReservationStatus(status))
        {
            throw new ValidationException(ValidationErrorCode.InvalidReservationStatus);
        }
        return status;
    }

    public static bool IsValidReservationStatus(ReservationStatus status)
    {
        return Enum.IsDefined(typeof(ReservationStatus), status);
    }

    public static int ValidateReservationId(int id)
    {
        if (!IsValidReservationId(id))
        {
            throw new ValidationException(ValidationErrorCode.InvalidId);
        }
        return id;
    }

    public static bool IsValidReservationId(int id) => id > 0;
}
}
```

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.

```

using SmartStay.Common.Enums;

namespace SmartStay.Validation.Validators
{
public static class RoomValidator
{
    public static RoomType ValidateRoomType(RoomType roomType)
    {
        if (!IsValidRoomType(roomType))
        {
            throw new ValidationException(ValidationErrorCode.InvalidRoomType);
        }
        return roomType;
    }

    public static bool IsValidRoomType(RoomType roomType) => Enum.IsDefined(typeof(RoomType),
roomType);

    public static bool ValidateAvailability(bool isAvailable)
    {
        if (!IsValidAvailability(isAvailable))
        {
            throw new ValidationException(ValidationErrorCode.InvalidAvailabilityStatus);
        }
        return isAvailable;
    }

    public static bool IsValidAvailability(bool isAvailable) => isAvailable || !isAvailable;

    public static int ValidateRoomId(int id)
    {
        if (!IsValidRoomId(id))
        {
            throw new ValidationException(ValidationErrorCode.InvalidId);
        }
        return id;
    }

    public static bool IsValidRoomId(int id) => id > 0;
}
}
```

Index