TAP Travel Components

Generated by Doxygen 1.8.11

Contents

Chapter 1

TAP 2017/18

1.1 Introduction

The project is a toy solution to the problem of planning trips between cities, and consists of two related components:

- the Travel Company Component, which is used to manage confederations of Travel Companies, where any Travel Company is a provider of point-to-point legs;
- the Planner Component, which uses Travel Companies to plan a multi-hops trips between cities.

1.2 Implementation

You're required to provide (by committing on your own private GitHub repository for this assignment) a Visual Studio 2017 project/solution file including distinct projects implementing the two required components. The solution may contain other projects, for instance those for testing.

Since we're using Ninject, the assembly of each component must also contain a Ninject Module binding the specification interfaces to your implementation classes.

With the obvious exception of connecting with the DBs referenced by the connection strings passed to the components' methods, it is forbidden to open or create files/registry keys/..., or establish database/network/... connections and so on.

Your DLLs must not depend on any library, other than the TAP2017_2018_TravelCompanyInterface, or respectively TAP2017_2018_PlannerInterface, Ninject, EF and standard .NET assemblies. You can use other libraries only if they have been explicitly approved in the TAP Forum by the teacher (that is, if you think there is a useful library out there, just ask on the forum if you can use it... the answer will most probably be yes, but you have nonetheless to explicitly ask for it).

Before delivering your work, please keep in mind that passing the provided tests is a minimum requirement only. The fact that your implementation passes these test does not imply, of course, its correctness. Indeed, many other tests and code inspection will be used to evaluate your work and it will be evaluated *once*.

1.3 Travel Company Component

The namespace TAP2017_2018_TravelCompanyInterface contains the declarations of a set of interfaces modeling the required types for a broker of (toy) travel companies. Such a broker manages a group of travel companies, that is, it provides factories to create new, and upload already existing ones. The data of the managed travel companies are saved on the broker database when successfully creating a new travel company, and used to upload them at need. Such database is the parameter of the installation method of the broker.The broker does not directly create and manage travel companies, but works through *factories*, following a standard pattern.

Each travel company manages point-to-point *legs* between cities. We make the simplifying assumption that *cities* are identified by non-null non-empty strings consisting of letters and digits only.

There are different kinds of transportation, categorized by TransportType, and two cities can be connected by many different legs offered by the same travel company (or by different companies). Legs sharing source and destination can differ, for instance, by transportation and/or by cost. However, each company cannot offer two distinct legs having the same values for all their properties. Thus, any attempt to create two legs having the same source, destination, travel mean and cost must fail by throwing TapDuplicatedObjectException.

Any travel company can be accessed through two different interfaces:

- IReadOnlyTravelCompany, providing the end users with functions to find legs satisfiyng some specific requirement, like, for instance, the connected cities, the choice of transportation or the cost;
- ITravelCompany, providing the travel company administrators with functions to manage the available legs, by adding or deleting them.

Distinct factories are provided to create and retrieve objects of IReadOnlyTravelCompany and of ITravelCompany type.

ITravelCompanyFactory provides methods for both creating a new travel company and loading an existing one, while IReadOnlyTravelCompanyFactory only allows to load an existing travel company.

Each travel company owns a database, which is the parameter of its constructor, used to store its legs, and it is not possible that the same database is shared among companies nor that it is used both for a travel company and its broker. Thus, any attempt to use the same database for two different owners must fail by throwing SameConnectionStringException. Moreover, the names of travel companies must be unique; any attempt to create two different travel companies with the same name must fail by throwing TapDuplicatedObjectException.

1.4 Planner Component

The namespace TAP2017_2018_PlannerInterface contains the declarations of a set of interfaces modeling the required types for a travel agency, offering to end users the service of finding a sequence of legs connecting, if possible, the source and destination of a trip in the best way, accordingly to the user requirements.

This component requires another component, the Travel Company Component, which provides interfaces to manage individual legs.

The main interface of this component is IPlanner, with methods to manage the known Travel Companies (adding, removing and listing) and to find the best match for a required trip between two cities, using legs offered by the known Travel Companies.

1.5 Global requirements for both components

Each method that receives:

- a `null` argument must throw the exception ArgumentNullException, unless explicitly specified differently
- an empty string argument must throw the exception ArgumentException, unless explicitly specified differently
- a string that is too short or too long must throw ArgumentException; the allowed string length ranges are contained in TAP2017_2018_TravelCompanyInterface.DomainConstraints
- an (integer) identifier corresponding to a non-existent entity must throw NonexistentObjectException

Any generic failure, not listed above, to persist or retrieve the data to/from the DB must be communicated by throwing DbConnectionException.

Your implementation must not throw any exception that has not been explicitly listed in this documentation; if you think that there is no specific exception for some possible situation, please ask on the forum (if this is really the case, a new exception will be added in this specification).

Please note that these requirements are intentionally *not* repeated for each and every method of the specification, and must be met by all your methods (implementing the interfaces described by this document; private methods can behave as they please ;-))

Chapter 2

Namespace Index

2.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

Namespace Documentation

5.1 TAP2017_2018_PlannerInterface Namespace Reference

declarations of a set of interfaces modeling the required types for a travel agency, offering to end users the service of finding a sequence of legs connecting, if possible, the source and destination of a trip in the best way, accordingly to the user requirements.

Classes

- interface IPlanner

  An IPlanner is an object that creates a (possibly multi-leg) trip between two cities.
- interface IPlannerFactory

  Planner factory.
- interface ITrip

  A trip is represented as a list of subsequent hops between cities, a cost and a distance.

Enumerations

- enum FindOptions { MinimumDistance, MinimumCost, MinimumHops }

  The type to express end user preferences in ITrip.FindTrip

5.1.1 Detailed Description

declarations of a set of interfaces modeling the required types for a travel agency, offering to end users the service of finding a sequence of legs connecting, if possible, the source and destination of a trip in the best way, accordingly to the user requirements.

5.1.2 Enumeration Type Documentation

5.1.2.1 enum FindOptions [strong]

The type to express end user preferences in ITrip.FindTrip

Enumerator

MinimumDistance To be used to find the trip with the smallest distance

MinimumCost To be used to find the cheapest trip

MinimumHops To be used to find the trip requiring less legs to be completed

5.2 TAP2017_2018_TravelCompanyInterface Namespace Reference

declarations of a set of interfaces modeling the required types for a broker of (toy) travel companies.

Namespaces

- namespace Exceptions

  Exceptions for travel company component

Classes

- class DomainConstraints
- interface ILegDTO

  DTO Object for ILeg. It encapsulates the raw data of a direct connection between two cities
- interface IReadOnlyTravelCompany

  Travel company interface for the end users, it limoits accesses to the travel company to read only ones.
- interface IReadOnlyTravelCompanyFactory

  Read only travel company factory.
- interface ITravelCompany

  Travel company interface for the travel company staff.
- interface ITravelCompanyBroker

  The broker for a group of travel companies.
- interface ITravelCompanyBrokerFactory

  The factory for travel company brokers.
- interface ITravelCompanyFactory

  Travel company factory.

Enumerations

- enum TransportType {
  None = 0, Plane = 1, Train = 2, Bus = 4,
  Ship = 8 }

  Enumerates the transport types.

5.2.1 Detailed Description

declarations of a set of interfaces modeling the required types for a broker of (toy) travel companies.

5.2.2 Enumeration Type Documentation

5.2.2.1 enum TransportType [strong]

Enumerates the transport types.

Flags to select possibly several transport types at the same time

Enumerator

None

Plane

Train

Bus

Ship

5.3 TAP2017_2018_TravelCompanyInterface.Exceptions Namespace Reference

Exceptions for travel company component

Classes

- class DbConnectionException

  Thrown to notify connection errors
- class NonexistentObjectException

  Thrown if the code try to access a nonexistent object
- class NonexistentTravelCompanyException

  Thrown if the code try to get a nonexistent (read only) travel company
- class SameConnectionStringException

  Thrown to notify that the connection string is already in use for a different purpose
- class TapDuplicatedObjectException

  Thrown to notify the attempt to create two instances of the same object
- class TapException

  Superclass of most exceptions for the component

5.3.1 Detailed Description

Exceptions for travel company component

Chapter 6

Class Documentation

6.1 DbConnectionException Class Reference

Thrown to notify connection errors

Public Member Functions

- DbConnectionException ()
- DbConnectionException (string message)
- DbConnectionException (string message, Exception inner)

Protected Member Functions

- DbConnectionException (SerializationInfo info, StreamingContext context)

6.1.1 Detailed Description

Thrown to notify connection errors

6.1.2 Constructor & Destructor Documentation

6.1.2.1 DbConnectionException ( )

6.1.2.2 DbConnectionException (string *message*)

6.1.2.3 DbConnectionException (string *message,* Exception *inner*)

6.1.2.4 DbConnectionException (SerializationInfo *info,* StreamingContext *context*) [protected]

6.2 DomainConstraints Class Reference

Public Attributes

- const int NameMinLength = 1

The minimum number of chars allowed in names for all the elements of the domain

- const int NameMaxLength = 32

  The maximum number of chars allowed in names for all the elements of the domain

- const int ConnectionStringMinLength = 10

  The minimum number of chars allowed in connection strings

- const int ConnectionStringMaxLength = 200

  The maximum number of chars allowed in connection strings

6.2.1 Member Data Documentation

6.2.1.1 const int ConnectionStringMaxLength = 200

The maximum number of chars allowed in connection strings

6.2.1.2 const int ConnectionStringMinLength = 10

The minimum number of chars allowed in connection strings

6.2.1.3 const int NameMaxLength = 32

The maximum number of chars allowed in names for all the elements of the domain

6.2.1.4 const int NameMinLength = 1

The minimum number of chars allowed in names for all the elements of the domain

6.3 ILegDTO Interface Reference

DTO Object for ILeg. It encapsulates the raw data of a direct connection between two cities

Properties

- string From `[get]`

  Gets the source city

- string To `[get]`

  Gets the destination city

- int Distance `[get]`

  Gets the distance between From and To

- int Cost `[get]`

  Gets the cost of going from From to To

- TransportType Type `[get]`

  Gets the transport type

6.3.1 Detailed Description

DTO Object for ILeg. It encapsulates the raw data of a direct connection between two cities

6.3.2 Property Documentation

6.3.2.1 int Cost [get]

Gets the cost of going from From to To

6.3.2.2 int Distance [get]

Gets the distance between From and To

6.3.2.3 string From [get]

Gets the source city

6.3.2.4 string To [get]

Gets the destination city

6.3.2.5 TransportType Type [get]

Gets the transport type

6.4 IPlanner Interface Reference

An IPlanner is an object that creates a (possibly multi-leg) trip between two cities.

Public Member Functions

- void AddTravelCompany (IReadOnlyTravelCompany readonlyTravelCompany)

  Adds a travel company destination the collection of those destination be used for the planning.
- void RemoveTravelCompany (IReadOnlyTravelCompany readonlyTravelCompany)

  Removes a travel company from the collection of those destination be used for the planning.
- bool ContainsTravelCompany (IReadOnlyTravelCompany readonlyTravelCompany)

  Verifies if the travel company in the collection of those in use by the planner
- IEnumerable< IReadOnlyTravelCompany > KnownTravelCompanies ()

  Returns the sequence of travel companies in use by the planner (in an unknown order).
- ITrip FindTrip (string source, string destination, FindOptions options, TransportType allowedTransportTypes)

  Finds the best trip source source destination destination , according destination options , using only transport of a type flagged in allowedTransportType .

6.4.1 Detailed Description

An IPlanner is an object that creates a (possibly multi-leg) trip between two cities.

6.4.2 Member Function Documentation

6.4.2.1 void AddTravelCompany (IReadOnlyTravelCompany *readonlyTravelCompany*)

Adds a travel company destination the collection of those destination be used for the planning.

Parameters

readonlyTravelCompany	The readonly travel company destination be added.

Exceptions

TapDuplicatedObjectException	Thrown if *readonlyTravelCompany* is already in use by the planner

6.4.2.2 bool ContainsTravelCompany (IReadOnlyTravelCompany *readonlyTravelCompany*)

Verifies if the travel company in the collection of those in use by the planner

Returns

`true`, if *readonlyTravelCompany* is contained in the set of travel companies in use by the planner, `false` otherwise.

Parameters

readonlyTravelCompany	Any readonly travel company destination be looked for among those in use by the planner.

6.4.2.3 ITrip FindTrip (string *source,* string *destination,* FindOptions *options,* TransportType *allowedTransportTypes*)

Finds the best trip source *source* destination *destination* , according destination *options* , using only transport of a type flagged in *allowedTransportType* .

Parameters

source	The source city
destination	The destination city
options	Specifies what is the required goal, that is what needs destination be minimized
allowedTransportTypes	Flags specifying which are the allowed transport types.

Returns

The best trip, if it exists, or `null` if there is no path from *source* to *destination* . A city is always considered connected destination itself, so an empty trip is returned when *source* and *destination* coincide

Exceptions

ArgumentNullException	Thrown if *source* or *destination* are `null`
ArgumentException	Thrown if any of the following conditions holds • *source* or *destination* is not `null` but is not well-formed. These parameters must be non-empty strings consisting of letters and/or digits only • *allowedTransportTypes* is TransportType.None

6.4.2.4 IEnumerable<IReadOnlyTravelCompany> KnownTravelCompanies ( )

Returns the sequence of travel companies in use by the planner (in an unknown order).

Returns

The travel companies used by the planner.

6.4.2.5 void RemoveTravelCompany (IReadOnlyTravelCompany *readonlyTravelCompany*)

Removes a travel company from the collection of those destination be used for the planning.

Parameters

readonlyTravelCompany	The readonly travel company destination be removed.

Exceptions

NonexistentTravelCompanyException	Thrown if *readonlyTravelCompany* is not in the set of travel companies in use by the planner

6.5 IPlannerFactory Interface Reference

Planner factory.

Public Member Functions

- IPlanner CreateNew ()

  Creates a new IPlanner.

6.5.1 Detailed Description

Planner factory.

6.5.2 Member Function Documentation

6.5.2.1 IPlanner CreateNew ( )

Creates a new IPlanner.

Returns

The new IPlanner.

6.6 IReadOnlyTravelCompany Interface Reference

Travel company interface for the end users, it limoits accesses to the travel company to read only ones.

Public Member Functions

- ReadOnlyCollection< ILegDTO > FindLegs (Expression< Func< ILegDTO, bool >> predicate)
  Finds all legs that satisfy the predicate.
- ReadOnlyCollection< ILegDTO > FindDepartures (string from, TransportType allowedTransportTypes)
  Finds all legs departing from the city from and using any of the transport types in allowedTransportTypes .

6.6.1 Detailed Description

Travel company interface for the end users, it limoits accesses to the travel company to read only ones.

6.6.2 Member Function Documentation

6.6.2.1 ReadOnlyCollection<ILegDTO> FindDepartures (string *from*, TransportType *allowedTransportTypes*)

Finds all legs departing from the city *from* and using any of the transport types in *allowedTransportTypes* .

Returns

The collection of legs having *from* as source and using only transport types flagged in *allowedTransportTypes* .

Parameters

from	The city source of all desired legs.
allowedTransportTypes	The flags of all allowed transport types

Exceptions

ArgumentException	Thrown if *from* is not alphanumeric

6.6.2.2 ReadOnlyCollection<ILegDTO> FindLegs (Expression< Func< ILegDTO, bool >> *predicate*)

Finds all legs that satisfy the predicate.

Returns

The collection of legs satisfying the predicate.

Parameters

predicate	Predicate to be validated in order to get the legs.

Exceptions

ArgumentNullException	Throws if *predicate* is null

6.7 IReadOnlyTravelCompanyFactory Interface Reference

Read only travel company factory.

Public Member Functions

- IReadOnlyTravelCompany Get (string name)
  Load the read-only interface to a given travel company

6.7.1 Detailed Description

Read only travel company factory.

6.7.2 Member Function Documentation

6.7.2.1 IReadOnlyTravelCompany Get (string *name*)

Load the read-only interface to a given travel company

Returns

A new IReadOnlyTravelCompany

Parameters

name	Name of the TravelCompany

Exceptions

ArgumentException	Thrown if *name* is empty string.
NonexistentObjectException	Thrown if *name* does not denote a known travel company.

6.8 ITravelCompany Interface Reference

Travel company interface for the travel company staff.

Public Member Functions

- int CreateLeg (string from, string to, int cost, int distance, TransportType transportType)

  Creates an ILeg which connects two cities.
- void DeleteLeg (int legToBeRemovedId)

  Deletes the specified leg.
- ILegDTO GetLegDTOFromId (int legId)

  Gets a leg DTO from the leg identifier.

Properties

- string Name `[get]`

6.8.1 Detailed Description

Travel company interface for the travel company staff.

6.8.2 Member Function Documentation

6.8.2.1 int CreateLeg (string *from*, string *to*, int *cost*, int *distance*, TransportType *transportType*)

Creates an ILeg which connects two cities.

Returns

The leg id.

Parameters

from	Source of the leg
to	Destination of the leg
cost	Cost of the leg
distance	Distance to be covered
transportType	Transport type

Exceptions

ArgumentException	Thrown if any of the following conditions holds: • *cost* or *distance* are not strictly positive • *transportType* is `None` • *from* or *to* are not alphanumeric • *from* and *to* are equal
AlreadyExistsException	If a leg having the same values for all the parameters exists.

6.8.2.2 void DeleteLeg (int *legToBeRemovedId*)

Deletes the specified leg.

Parameters

legToBeRemovedId	The id of the leg to be deleted

6.8.2.3 ILegDTO GetLegDTOFromId (int *legId*)

Gets a leg DTO from the leg identifier.

Returns

Returns a DTO for the corresponding Leg

Parameters

legId	Leg identifier.

6.8.3 Property Documentation

6.8.3.1 string Name `[get]`

6.9 ITravelCompanyBroker Interface Reference

The broker for a group of travel companies.

Public Member Functions

- ITravelCompanyFactory GetTravelCompanyFactory ()

  Creates a travel company factory. The travel companies created by the resulting factory will be part of the group managed by the broker. The resulting factory will be able to upload only travel companies managed by the broker.
- IReadOnlyTravelCompanyFactory GetReadOnlyTravelCompanyFactory ()

  Creates a factory for read-only interfaces to travel companies. The resulting factory will be able to upload only read only versions of the travel companies managed by the broker.
- ReadOnlyCollection< string > KnownTravelCcompanies ()

  Returns a collection of all the travel companies created using this broker.

6.9.1 Detailed Description

The broker for a group of travel companies.

6.9.2 Member Function Documentation

6.9.2.1 IReadOnlyTravelCompanyFactory GetReadOnlyTravelCompanyFactory ( )

Creates a factory for read-only interfaces to travel companies. The resulting factory will be able to upload only read only versions of the travel companies managed by the broker.

Returns

a factory for read-only travel companies managed by the broker

6.9.2.2 ITravelCompanyFactory GetTravelCompanyFactory ( )

Creates a travel company factory. The travel companies created by the resulting factory will be part of the group managed by the broker. The resulting factory will be able to upload only travel companies managed by the broker.

Returns

a travel company factory working on the group managed by the broker

6.9.2.3 ReadOnlyCollection<string> KnownTravelCompanies ( )

Returns a collection of all the travel companies created using this broker.

Returns

A collection of travel companies.

6.10 ITravelCompanyBrokerFactory Interface Reference

The factory for travel company brokers.

Public Member Functions

- ITravelCompanyBroker CreateNewBroker (string dbConnectionString)

  Creates a new broker for travel companies. The database denoted by dbConnectionString is initialized (previous data, if any, will be lost) and will be used to store information about the travel companies managed by the resulting broker.
- ITravelCompanyBroker GetBroker (string dbConnectionString)

  Load an existing broker for travel companies. The database denoted by dbConnectionString must be already initialized.

6.10.1 Detailed Description

The factory for travel company brokers.

6.10.2 Member Function Documentation

6.10.2.1 ITravelCompanyBroker CreateNewBroker (string *dbConnectionString*)

Creates a new broker for travel companies. The database denoted by *dbConnectionString* is initialized (previous data, if any, will be lost) and will be used to store information about the travel companies managed by the resulting broker.

Parameters

dbConnectionString	Connection string for the database used by the broker to memorize the data of managed travel companies, that is their name and private DBs

Returns

A new broker, not managing any travel company yet.

6.10.2.2 ITravelCompanyBroker GetBroker (string *dbConnectionString*)

Load an existing broker for travel companies. The database denoted by *dbConnectionString* must be already initialized.

Parameters

dbConnectionString	Connection string for the database used by the broker to memorize the data of managed travel companies, that is their name and private DBs

Returns

The broker managing travel companies whose data are saved in *dbConnectionString*

6.11 ITravelCompanyFactory Interface Reference

Travel company factory.

Public Member Functions

- ITravelCompany CreateNew (string travelCompanyConnectionString, string name)

  Creates new travel company. The data of the newly created travel company are permanently stored by the component.
- ITravelCompany Get (string name)

  Load the specified ITravelCompany.

6.11.1 Detailed Description

Travel company factory.

6.11.2 Member Function Documentation

6.11.2.1 ITravelCompany CreateNew (string *travelCompanyConnectionString*, string *name*)

Creates new travel company. The data of the newly created travel company are permanently stored by the component.

Parameters

travelCompanyConnectionString	ConnectionString which identifies the travel company database. The database is initialized by this operation.
name	Travel Company name

Returns

The newly created travel company

Exceptions

TapDuplicatedObjectException	If *name* already denotes a known travel company.
SameConnectionStringException	If *travelCompanyConnectionString* already denotes the database of another travel company or the database of the component.

6.11.2.2 ITravelCompany Get (string *name*)

Load the specified ITravelCompany.

Returns

The ITravelCompany having the given *name*

Parameters

name	The name of the desired ITravelCompany

Exceptions

NonexistentTravelCompanyException	If *name* does not denote a known travel company.

6.12 ITrip Interface Reference

A trip is represented as a list of subsequent hops between cities, a cost and a distance.

Properties

- string From `[get]`

  Gets the source of its first leg (`null` if the trip consists of no legs)
- string To `[get]`

  Gets the destination of its last leg (`null` if the trip consists of no legs)
- ReadOnlyCollection< ILegDTO > Path `[get]`

  Gets the sequence of legs this trip consists of; the destination of a leg in the sequence is equal to the source of the subsequent leg
- int TotalCost `[get]`

  Gets the total cost (0 if the trip consists of no legs)
- int TotalDistance `[get]`

  Gets the total distance (0 if the trip consists of no legs)

6.12.1 Detailed Description

A trip is represented as a list of subsequent hops between cities, a cost and a distance.

6.12.2 Property Documentation

6.12.2.1 string From `[get]`

Gets the source of its first leg (`null` if the trip consists of no legs)

6.12.2.2 ReadOnlyCollection<ILegDTO> Path `[get]`

Gets the sequence of legs this trip consists of; the destination of a leg in the sequence is equal to the source of the subsequent leg

6.12.2.3 string To `[get]`

Gets the destination of its last leg (`null` if the trip consists of no legs)

6.12.2.4 int TotalCost [get]

Gets the total cost (0 if the trip consists of no legs)

6.12.2.5 int TotalDistance [get]

Gets the total distance (0 if the trip consists of no legs)

6.13 NonexistentObjectException Class Reference

Thrown if the code try to access a nonexistent object

Public Member Functions

- NonexistentObjectException ()
- NonexistentObjectException (string message)
- NonexistentObjectException (string message, Exception inner)

Protected Member Functions

- NonexistentObjectException (SerializationInfo info, StreamingContext context)

6.13.1 Detailed Description

Thrown if the code try to access a nonexistent object

6.13.2 Constructor & Destructor Documentation

6.13.2.1 NonexistentObjectException ( )

6.13.2.2 NonexistentObjectException (string *message*)

6.13.2.3 NonexistentObjectException (string *message,* Exception *inner*)

6.13.2.4 NonexistentObjectException (SerializationInfo *info,* StreamingContext *context*) [protected]

6.14 NonexistentTravelCompanyException Class Reference

Thrown if the code try to get a nonexistent (read only) travel company

Public Member Functions

- NonexistentTravelCompanyException ()
- NonexistentTravelCompanyException (string message)
- NonexistentTravelCompanyException (string message, Exception innerException)

Protected Member Functions

- NonexistentTravelCompanyException (SerializationInfo info, StreamingContext context)

6.14.1 Detailed Description

Thrown if the code try to get a nonexistent (read only) travel company

6.14.2 Constructor & Destructor Documentation

6.14.2.1 NonexistentTravelCompanyException ( )

6.14.2.2 NonexistentTravelCompanyException (string *message*)

6.14.2.3 NonexistentTravelCompanyException (string *message,* Exception *innerException*)

6.14.2.4 NonexistentTravelCompanyException (SerializationInfo *info,* StreamingContext *context*) [protected]

6.15 SameConnectionStringException Class Reference

Thrown to notify that the connection string is already in use for a different purpose

Public Member Functions

- SameConnectionStringException ()
- SameConnectionStringException (string message)
- SameConnectionStringException (string message, Exception inner)

Protected Member Functions

- SameConnectionStringException (SerializationInfo info, StreamingContext context)

6.15.1 Detailed Description

Thrown to notify that the connection string is already in use for a different purpose

6.15.2 Constructor & Destructor Documentation

6.15.2.1 SameConnectionStringException ( )

6.15.2.2 SameConnectionStringException (string *message*)

6.15.2.3 SameConnectionStringException (string *message,* Exception *inner*)

6.15.2.4 SameConnectionStringException (SerializationInfo *info,* StreamingContext *context*) [protected]

6.16 TapDuplicatedObjectException Class Reference

Thrown to notify the attempt to create two instances of the same object

Public Member Functions

- TapDuplicatedObjectException ()
- TapDuplicatedObjectException (string message)
- TapDuplicatedObjectException (string message, Exception inner)

Protected Member Functions

- TapDuplicatedObjectException (SerializationInfo info, StreamingContext context)

6.16.1 Detailed Description

Thrown to notify the attempt to create two instances of the same object

6.16.2 Constructor & Destructor Documentation

6.16.2.1 TapDuplicatedObjectException ( )

6.16.2.2 TapDuplicatedObjectException (string *message*)

6.16.2.3 TapDuplicatedObjectException (string *message,* Exception *inner*)

6.16.2.4 TapDuplicatedObjectException (SerializationInfo *info,* StreamingContext *context*) [protected]

6.17 TapException Class Reference

Superclass of most exceptions for the component

Public Member Functions

- TapException ()
- TapException (string message)
- TapException (string message, Exception inner)

Protected Member Functions

- TapException (SerializationInfo info, StreamingContext context)

6.17.1 Detailed Description

Superclass of most exceptions for the component

6.17.2 Constructor & Destructor Documentation

6.17.2.1 TapException ( )

6.17.2.2 TapException (string *message*)

6.17.2.3 TapException (string *message,* Exception *inner*)

6.17.2.4 TapException (SerializationInfo *info,* StreamingContext *context*) [protected]

Index