# TAP Auction Site Component

Generated by Doxygen 1.8.11

# Contents

# Chapter 1

# TAP Auction Site Component

## 1.1 Introduction

The namespace TAP2018_19.AuctionSite contains the declarations of a set of interfaces modeling the required types for a toy auction site (think ebay, think smaller, smaller, smaller...).

Implementations of the interfaces in AlarmClock.Interfaces will be provided by the AuctionSite clients and are not part of this project.

Only interfaces in AuctionSite.Interfaces are implemented by the AuctionSite component.

Using the ISiteFactory a client of this component can initialize the system, create a new site, or load an existing one.

An ISite represents a site for online auctions, managing users, their sessions, and their auctions. It has a unique name, which is a non empty nor null string, and defines

- the timezone used to get the time;
- the minumum increment allowed in bidding;
- the time out of inactive sessions.

It is the root of an aggregate, in the sense that it owns its users, sessions and auctions, that have no meaning outside their site.

Users, represented by IUser, must have a valid session to act on a site, and no user can have two valid sessions on the same site at the same time.

Sessions, represented by ISession, start with a login, and may either end by an explicit logout, or time out by a prolonged inactivity of their user owner. Each time a user access his/her session by an explicit login, by creating an auction or by bidding on an auction, the expiration time of the session is reset to the site expiration time. Closed sessions have to be dropped each five minutes, or by explicit request of cleanup.

Time management is provided by a required component implementing the interface TAP2018_19.AlarmClock.↩ Interfaces. The main type in such component is IAlarmClock, and it represents a clock synchronized on a specific time zone. IAlarmClock has methods for checking the current time, and setting an alarm, that is, firing the ringing event at a given time.

This is just an overview; each type and each method has its own, more detailed, specific documentation.

## 1.2 Global requirements

Names and Id, when provided for a type, are unique within the context where the type is meaningful. Thus, for instance, the name of an auction site is unique, as well as the name of a user within an auction site (but the same name can be repeated in different auction sites).

Any attempt to obtain two different objects of the same class with the same name (or Id) must fail by throwing NameAlreadyInUseException.

Each method that receives:

- a `null` argument must throw the exception `ArgumentNullException`, unless explicitly specified
- a string that is too short or too long must throw `ArgumentException`; the allowed string length ranges are contained in DomainConstraints

Any method invocation on a deleted object, that is an object whose persistent counterpart has been removed, must throw `InvalidOperationException`.

Any generic failure to persist or retrieve the data to/from the DB must be communicated by throwing `UnavailableDbException`.

Any detected attempt to save an entity on the DB which has been concurrently modified must be communicated by throwing `ConcurrentChangeException`. Concurrency management is not required for a project to be acceptable. But, it is an element of evaluation, and if you want to do it, then you must use `Concurrent↩ ChangeException`.

Your implementation must not throw any exception that has not been explicitly listed here.

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 ;-) )

## 1.3 Implementation

You're required to provide (by committing it on your personal space into the TAP subversion repository) a Visual Studio 2017 project/solution file for building a (managed .NET DLL) assembly containing an implementation for this specification.

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

All type declarations must be contained in a namespace whose name correspond to your family name (if it includes unacceptable chars, repleace them by _).

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

Your DLL must not depend on any library, other than the TAP2018_19.AuctionSite.Interfaces, TAP2018_19.Alarm↩ Clock.Interfaces, 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 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*.

# 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 TAP2018_19 Namespace Reference

### Namespaces

- namespace AlarmClock
- namespace AuctionSite

## 5.2 TAP2018_19.AlarmClock Namespace Reference

### Namespaces

- namespace Interfaces

  *An elementary time management component needed by the AuctionSite component.*

## 5.3 TAP2018_19.AlarmClock.Interfaces Namespace Reference

An elementary time management component needed by the AuctionSite component.

### Classes

- interface IAlarm

  *An alarm, raising a RingingEvent every tot seconds, where tot is defined by the alarm object constructor*
- interface IAlarmClock

  *A clock synchronized on the given Timezone*
- interface IAlarmClockFactory

### 5.3.1 Detailed Description

An elementary time management component needed by the AuctionSite component.

This component provides types to represent alarm clock and alarms.

The implementation of the interfaces defined by this component is not part of the project. Objects implementing them will be provided by the client code of the AuctionSite component via dependency injection

## 5.4 TAP2018_19.AuctionSite Namespace Reference

### Namespaces

- namespace Interfaces

  *The interfaces provided by the AuctionSite component.*

## 5.5 TAP2018_19.AuctionSite.Interfaces Namespace Reference

The interfaces provided by the AuctionSite component.

### Classes

- class AuctionSiteException

  *The root for the exception hierarchy of this component. To be used only if no more specific exception is available for the error (it should not happen that you ever need to throw it).*
- class ConcurrentChangeException

  *Thrown to notify the attempt to save an entity on the DB which has been concurrently modified (concurrency management is not required; but, if you want to do it use this exception)*
- class DomainConstraints

  *Numeric constants used to express constraints on names, passwords, and time zone*
- interface IAuction

  *The auctions managed by sites.*
- class InexistentNameException

  *Thrown to notify the attempt to access by its name an entity which is not available on the database.*
- interface ISession

  *The sessions of the users on the site, valid from login to logout. It may become invalid also if the user is idle for the expiration time set for the site.*
- interface ISite

  *The auction sites.*
- interface ISiteFactory

  *This component factory.*
- interface IUser

  *The users of the auction system. Equals on users must be true iff the two users belong to the same site and have the same Username.*
- class NameAlreadyInUseException

  *Thrown to notify the attempt to create an entity whose name is already in use. For instance, a site with the same name of another site, or a user with the same name of another user on the same site.*
- class UnavailableDbException

  *Thrown to notify that no connection with the database has been established.*
- class UnavailableTimeMachineException

  *Thrown to notify the attempt to create an auction with expiration date in the past.*

### 5.5.1 Detailed Description

The interfaces provided by the AuctionSite component.

Only interfaces in this namespace are implemented by the AuctionSite component.

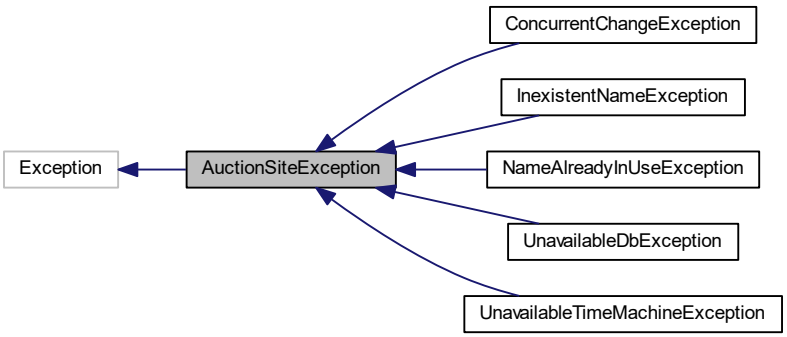
# Chapter 6

# Class Documentation

## 6.1 AuctionSiteException Class Reference

The root for the exception hierarchy of this component. To be used only if no more specific exception is available for the error (it should not happen that you ever need to throw it).

Inheritance diagram for AuctionSiteException:




### Protected Member Functions

- AuctionSiteException ()
- AuctionSiteException (string message)
- AuctionSiteException (string message, Exception inner)
- AuctionSiteException (SerializationInfo info, StreamingContext context)

### 6.1.1 Detailed Description

The root for the exception hierarchy of this component. To be used only if no more specific exception is available for the error (it should not happen that you ever need to throw it).

### 6.1.2 Constructor & Destructor Documentation

#### 6.1.2.1 AuctionSiteException ( ) [protected]

#### 6.1.2.2 AuctionSiteException ( string *message* ) [protected]

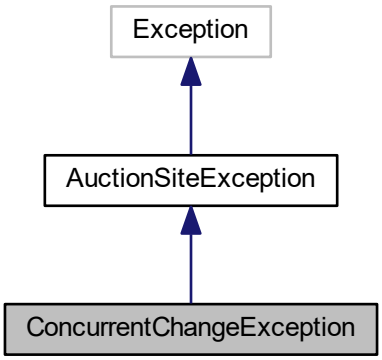
#### 6.1.2.3 AuctionSiteException ( string *message,* Exception *inner* ) [protected]

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

## 6.2 ConcurrentChangeException Class Reference

Thrown to notify the attempt to save an entity on the DB which has been concurrently modified (concurrency management is not required; but, if you want to do it use this exception)

Inheritance diagram for ConcurrentChangeException:




### Public Member Functions

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

### Protected Member Functions

- ConcurrentChangeException (SerializationInfo info, StreamingContext context)

### 6.2.1 Detailed Description

Thrown to notify the attempt to save an entity on the DB which has been concurrently modified (concurrency management is not required; but, if you want to do it use this exception)

### 6.2.2 Constructor & Destructor Documentation

#### 6.2.2.1 ConcurrentChangeException ( )

#### 6.2.2.2 ConcurrentChangeException ( string *message* )

#### 6.2.2.3 ConcurrentChangeException ( string *message,* Exception *inner* )

#### 6.2.2.4 ConcurrentChangeException ( SerializationInfo *info,* StreamingContext *context* ) [protected]

## 6.3 DomainConstraints Class Reference

Numeric constants used to express constraints on names, passwords, and time zone

### Public Attributes

- const int MinSiteName = 1

  *The minimal length of a string to be a well formed site name*
- const int MaxSiteName = 128

  *The maximal length of a string to be a well formed site name*
- const int MinUserName = 3

  *The minimal length of a string to be a well formed user name*
- const int MaxUserName = 64

  *The maximal length of a string to be a well formed user name*
- const int MinUserPassword = 4

  *The minimal length of a string to be an acceptable password*
- const int MinTimeZone = -12

  *The minimal value acceptable as a time zone*
- const int MaxTimeZone = 12

  *The maximal value acceptable as a time zone*

### 6.3.1 Detailed Description

Numeric constants used to express constraints on names, passwords, and time zone

### 6.3.2 Member Data Documentation

#### 6.3.2.1 const int MaxSiteName = 128

The maximal length of a string to be a well formed site name

#### 6.3.2.2 const int MaxTimeZone = 12

The maximal value acceptable as a time zone

#### 6.3.2.3 const int MaxUserName = 64

The maximal length of a string to be a well formed user name

#### 6.3.2.4 const int MinSiteName = 1

The minimal length of a string to be a well formed site name

#### 6.3.2.5 const int MinTimeZone = -12

The minimal value acceptable as a time zone

#### 6.3.2.6 const int MinUserName = 3

The minimal length of a string to be a well formed user name

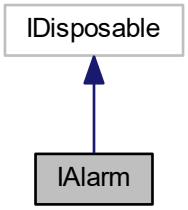
#### 6.3.2.7 const int MinUserPassword = 4

The minimal length of a string to be an acceptable password

## 6.4 IAlarm Interface Reference

An alarm, raising a RingingEvent every tot seconds, where tot is defined by the alarm object constructor

Inheritance diagram for IAlarm:




### Events

- Action RingingEvent

### 6.4.1 Detailed Description

An alarm, raising a RingingEvent every tot seconds, where tot is defined by the alarm object constructor

### 6.4.2 Event Documentation

#### 6.4.2.1 Action RingingEvent

## 6.5 IAlarmClock Interface Reference

A clock synchronized on the given Timezone

### Public Member Functions

- IAlarm InstantiateAlarm (int frequencyInMs)

  *Instantiate an alarm.*

### Properties

- int Timezone `[get]`

  *The time zone for an alarm clock*
- DateTime Now `[get]`

  *The current time*

### 6.5.1 Detailed Description

A clock synchronized on the given Timezone

### 6.5.2 Member Function Documentation

#### 6.5.2.1 IAlarm InstantiateAlarm ( int *frequencyInMs* )

Instantiate an alarm.

**Parameters**

| | |
|---|---|
| *frequencyInMs* | Frequency in milliseconds. |

**Returns**

A new alarm.

**Exceptions**

| | |
|---|---|
| *ArgumentOutOfRangeException* | If the frequency is not positive. |

### 6.5.3 Property Documentation

#### 6.5.3.1 DateTime Now [get]

The current time

#### 6.5.3.2 int Timezone [get]

The time zone for an alarm clock

## 6.6 IAlarmClockFactory Interface Reference

### Public Member Functions

- IAlarmClock InstantiateAlarmClock (int timezone)
  *Instantiate a new alarm clock with the given timezone .*

### 6.6.1 Member Function Documentation

#### 6.6.1.1 IAlarmClock InstantiateAlarmClock ( int *timezone* )

Instantiate a new alarm clock with the given *timezone* .

**Parameters**

| | |
|---|---|
| *timezone* | An integer between -12 and 12 (inclusive). |

**Returns**

A new alarm clock.

**Exceptions**

| | |
|---|---|
| *ArgumentOutOfRangeException* | When *timezone* is not in range. |

## 6.7 IAuction Interface Reference

The auctions managed by sites.

### Public Member Functions

- IUser CurrentWinner ()

  *Returns the user, if any, who has submitted the highest bid so far.*
- double CurrentPrice ()

  *Returns the current price, which is the lowest amount needed to best the second highest bid if two or more bids have been offered; otherwise, it coincides with the starting price.*
- void Delete ()

  *Disposes of the auction and all associated resources, if any.*
- bool BidOnAuction (ISession session, double offer)

  *Makes a bid for this auction on behalf of the session owner; only possible for still open auctions.*

### Properties

- int Id `[get]`

  *Gets the unique key used to identify the auctions.*
- IUser Seller `[get]`

  *Gets the user who is selling the object/service.*
- string Description `[get]`

  *Gets the description of the offered object/service.*
- DateTime EndsOn `[get]`

  *Gets the expiring time of the auction; no bid will be accepted after it.*

## 6.7.1 Detailed Description

The auctions managed by sites.

Equals on auctions must be true iff the two auctions belong to the same site and have the same Id.

## 6.7.2 Member Function Documentation

### 6.7.2.1 bool BidOnAuction ( ISession *session*, double *offer* )

Makes a bid for this auction on behalf of the session owner; only possible for still open auctions.

**Parameters**

| | |
|---|---|
| *session* | A valid session. The expiration time of the session is reset to the expiration time of the site for each correct bid, disregarding whether the bid is accepted or not. |
| *offer* | The amount offered as bid, that is, the maximum amount the user is willing to pay for the item. This maximum amount must remain confidential. |

**Returns**

True iff the bid is accepted, so that the status of the auction is changed.
The bid is rejected, so that the result is false, iff either of the following occurs

- the bidder is (already) the current winner and *offer* is lower than the maximum offer increased by minimumBidIncrement

- the bidder is not the current winner and *offer* is lower than the current price
- the bidder is not the current winner and *offer* is lower than the current price increased by minimumBid↩ Increment AND this is not the first bid

In all other cases, the bid is accepted, the result is true, and the status of the auction is changed as follows:

- if this is the first bid, then the maximum offer is set to *offer* , the current price is not changed (that is, it remains the starting price), and the bidder becomes the current winner;
- if the bidder was already winning this auction, the maximum offer is set to *offer* , current price and current winner are unchanged;
- if this is NOT the first bid, the bidder is NOT the current winner, and *offer* is higher than the current maximum offer, in the following denoted by CMO, then the current price is set to the minimum between *offer* and CMO+minimumBidIncrement, the maximum offer is set to *offer* , and the bidder becomes the current winner;
- if this is NOT the first bid, the bidder is NOT the current winner, and *offer* is NOT higher than the current maximum offer, in the following denoted by CMO, then the current price is set to the minimum between CMO and *offer* +minimumBidIncrement, and the current winner does not change.

**Exceptions**

| | |
|---|---|
| *InvalidOperationException* | The auction is already closed (or the corresponding permanent object does not exist anymore). |
| *ArgumentOutOfRangeException* | If *offer* is negative. |
| *ArgumentNullException* | If *session* is null. |
| *ArgumentException* | If *session* is not null and, one of the following conditions is true:<br>• the session is not valid anymore;<br>• the logged user is also the Seller of this auction;<br>• the logged user is a user of a site different from the site of the Seller. |

#### 6.7.2.2 double CurrentPrice ( )

Returns the current price, which is the lowest amount needed to best the second highest bid if two or more bids have been offered; otherwise, it coincides with the starting price.

#### 6.7.2.3 IUser CurrentWinner ( )

Returns the user, if any, who has submitted the highest bid so far.

In case no bids have been offered yet, it returns null. It may also return null in case of closed auction whose winner has been deleted from the site (after the auction ended).

#### 6.7.2.4 void Delete ( )

Disposes of the auction and all associated resources, if any.

### 6.7.3 Property Documentation

#### 6.7.3.1 string Description [get]

Gets the description of the offered object/service.

#### 6.7.3.2 DateTime EndsOn [get]

Gets the expiring time of the auction; no bid will be accepted after it.

#### 6.7.3.3 int Id [get]

Gets the unique key used to identify the auctions.

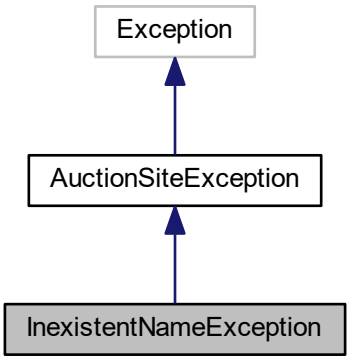
#### 6.7.3.4 IUser Seller [get]

Gets the user who is selling the object/service.

## 6.8 InexistentNameException Class Reference

Thrown to notify the attempt to access by its name an entity which is not available on the database.

Inheritance diagram for InexistentNameException:




### Public Member Functions

- InexistentNameException (string name)
- InexistentNameException (string name, string message)
- InexistentNameException (string name, string message, Exception inner)
- InexistentNameException (SerializationInfo info, StreamingContext context)

### Properties

- string Name [get]

### Additional Inherited Members

#### 6.8.1 Detailed Description

Thrown to notify the attempt to access by its name an entity which is not available on the database.

#### 6.8.2 Constructor & Destructor Documentation

##### 6.8.2.1 InexistentNameException ( string *name* )

##### 6.8.2.2 InexistentNameException ( string *name,* string *message* )

##### 6.8.2.3 InexistentNameException ( string *name,* string *message,* Exception *inner* )

##### 6.8.2.4 InexistentNameException ( SerializationInfo *info,* StreamingContext *context* )

#### 6.8.3 Property Documentation

##### 6.8.3.1 string Name [get]

## 6.9 ISession Interface Reference

The sessions of the users on the site, valid from login to logout. It may become invalid also if the user is idle for the expiration time set for the site.

### Public Member Functions

- bool IsValid ()

  *True iff correctly created (and present on the DB) and not yet logged out or timed out. A session times out if it has not been used for more than SessionExpirationInSeconds (of its site) seconds.*
- void Logout ()

  *Ends the session, making it invalid, and disposes of all associated resources, if any.*
- IAuction CreateAuction (string description, DateTime endsOn, double startingPrice)

  *Yields an auction for the described object/service. As a side effect, the expiration time of the session is reset (to the same value as if the session was newly created).*

### Properties

- string Id [get]

  *Gets the unique key used to identify the sessions.*
- DateTime ValidUntil [get]

  *Gets the current expiration time of the session.*
- IUser User [get]

  *Gets the user owner of the session.*

### 6.9.1 Detailed Description

The sessions of the users on the site, valid from login to logout. It may become invalid also if the user is idle for the expiration time set for the site.

Equals on sessions must be true iff the two sessions have the same Id.

### 6.9.2 Member Function Documentation

#### 6.9.2.1 IAuction CreateAuction ( string *description,* DateTime *endsOn,* double *startingPrice* )

Yields an auction for the described object/service. As a side effect, the expiration time of the session is reset (to the same value as if the session was newly created).

**Parameters**

| | |
|---|---|
| *description* | The description of the object/service for sale, a non-null and non-empty string. |
| *endsOn* | The expiring time of the auction; no bid will be accepted after it. |
| *startingPrice* | The starting price of the auction, a non-negative number; the first bid must be greater than or equal to this value. |

**Returns**

Returns the newly created auction, whose Id is an automatically-generated unique identifier.

**Exceptions**

| | |
|---|---|
| *InvalidOperationException* | The session is not valid (or the corresponding permanent object does not exist anymore). |
| *ArgumentNullException* | If *description* is null. |
| *ArgumentException* | If *description* is not null but empty. |
| *ArgumentOutOfRangeException* | If *startingPrice* is negative |
| *UnavailableTimeMachineException* | If *endsOn* precedes the current time (according to the IAlarmClock of the ISite) |

#### 6.9.2.2 bool IsValid ( )

True iff correctly created (and present on the DB) and not yet logged out or timed out. A session times out if it has not been used for more than SessionExpirationInSeconds (of its site) seconds.

#### 6.9.2.3 void Logout ( )

Ends the session, making it invalid, and disposes of all associated resources, if any.

### 6.9.3 Property Documentation

#### 6.9.3.1 string Id [get]

Gets the unique key used to identify the sessions.

#### 6.9.3.2 IUser User [get]

Gets the user owner of the session.

#### 6.9.3.3 DateTime ValidUntil [get]

Gets the current expiration time of the session.

## 6.10 ISite Interface Reference

The auction sites.

### Public Member Functions

- IEnumerable< IUser > GetUsers ()

  *Yields all the users of the site.*
- IEnumerable< ISession > GetSessions ()

  *Yields all the sessions of the site.*
- IEnumerable< IAuction > GetAuctions (bool onlyNotEnded)

  *Yields all the (not yet ended) auctions of the site.*
- ISession Login (string username, string password)

  *Yields the session for the user, new iff no valid session for him/her exists.*
- ISession GetSession (string sessionId)

  *Returns a session given its identifier.*
- void CreateUser (string username, string password)

  *Add a user of the site.*
- void Delete ()

  *Disposes of the site and all its associated resources.*
- void CleanupSessions ()

  *Immediately deletes all expired sessions. Implementations are required to invoke this method every five minutes (according to the given IAlarmClock).*

### Properties

- string Name [get]

  *The name of the auction site.*
- int Timezone [get]

  *The timezone of the auction site.*
- int SessionExpirationInSeconds [get]

  *The number of seconds needed for the session of an idle user to time out. A positive number*
- double MinimumBidIncrement [get]

  *The minimum amount allowed as increment (from the starting price) for a bid. A positive number*

### 6.10.1 Detailed Description

The auction sites.

Equals on sites must be true iff the two sites have the same Name.

### 6.10.2 Member Function Documentation

#### 6.10.2.1 void CleanupSessions ( )

Immediately deletes all expired sessions. Implementations are required to invoke this method every five minutes (according to the given IAlarmClock).

#### 6.10.2.2 void CreateUser ( string *username*, string *password* )

Add a user of the site.

**Parameters**

| | |
|---|---|
| *username* | User login (minimum length=DomainConstraints.MinUserName, maximum=DomainConstraints.MaxUserName chars). |
| *password* | User password (minimum length=DomainConstraints.MinUserPassword chars). |

**Exceptions**

| | |
|---|---|
| *NameAlreadyInUseException* | If *username* is already in use for a user of the site. |
| *ArgumentNullException* | If *username* or *password* are null. |
| *ArgumentException* | If *username* is not null, but its length is (strictly) smaller than DomainConstraints.MinUserName or (strictly) larger than DomainConstraints.MaxUserName, or if *password* is not null, but its length is (strictly) smaller than DomainConstraints.MinUserPassword. |

#### 6.10.2.3 void Delete ( )

Disposes of the site and all its associated resources.

#### 6.10.2.4 IEnumerable<IAuction> GetAuctions ( bool *onlyNotEnded* )

Yields all the (not yet ended) auctions of the site.

**Parameters**

| | |
|---|---|
| *onlyNotEnded* | If true, only the auctions not yet ended are taken into account. |

**Returns**

If not *onlyNotEnded* , all the auctions, otherwise only those not yet ended.

#### 6.10.2.5 ISession GetSession ( string *sessionId* )

Returns a session given its identifier.

**Parameters**

| | |
|---|---|
| *sessionId* | The session identifier, a non-null string. |

**Returns**

The corresponding session if it exists and is valid, null otherwise.

**Exceptions**

| | |
|---|---|
| *ArgumentNullException* | If *sessionId* is null. |

#### 6.10.2.6 IEnumerable<ISession> GetSessions ( )

Yields all the sessions of the site.

**Returns**

All the sessions of the site.

#### 6.10.2.7 IEnumerable<IUser> GetUsers ( )

Yields all the users of the site.

**Returns**

All the users of the site.

#### 6.10.2.8 ISession Login ( string *username,* string *password* )

Yields the session for the user, new iff no valid session for him/her exists.

No user can have two valid sessions on the same site at the same time.

**Parameters**

| | |
|---|---|
| *username* | User login. |
| *password* | User password. |

**Returns**

The session for the user or null if *username* and *password* do not correspond to a user of the site.

**Exceptions**

| | |
|---|---|
| *ArgumentNullException* | If *username* or *password* are null. |
| *ArgumentException* | If *username* is not null, but its length is (strictly) smaller than DomainConstraints.MinUserName or (strictly) larger than DomainConstraints.MaxUserName, or if *password* is not null, but its length is (strictly) smaller than DomainConstraints.MinUserPassword. |

### 6.10.3 Property Documentation

#### 6.10.3.1 double MinimumBidIncrement [get]

The minimum amount allowed as increment (from the starting price) for a bid. A positive number

#### 6.10.3.2 string Name [get]

The name of the auction site.

#### 6.10.3.3 int SessionExpirationInSeconds [get]

The number of seconds needed for the session of an idle user to time out. A positive number

#### 6.10.3.4 int Timezone [get]

The timezone of the auction site.

## 6.11 ISiteFactory Interface Reference

This component factory.

### Public Member Functions

- void Setup (string connectionString)

  *Creates a new DB (dropping existing previous version, if any), and initialize it with all the necessary DB objects for the component.*
- IEnumerable< string > GetSiteNames (string connectionString)

  *Yields the managed sites.*
- void CreateSiteOnDb (string connectionString, string name, int timezone, int sessionExpirationTimeIn↩ Seconds, double minimumBidIncrement)

  *Create a new site, identified by its name.*
- ISite LoadSite (string connectionString, string name, IAlarmClock alarmClock)

  *Yields the ISite object corresponding to an existing Site.*
- int GetTheTimezoneOf (string connectionString, string name)

  *Return the timezone of the specified site.*

### 6.11.1 Detailed Description

This component factory.

All its methods have a parameter `connectionString` representing the connection string needed to open a connection to an existing Microsoft SQL Server DB, used to permanently store the data of the managed auction sites.

### 6.11.2 Member Function Documentation

#### 6.11.2.1 void CreateSiteOnDb ( string *connectionString*, string *name*, int *timezone*, int *sessionExpirationTimeInSeconds*, double *minimumBidIncrement* )

Create a new site, identified by its name.

**Parameters**

| | |
|---|---|
| *connectionString* | The connection string. |
| *name* | The name of the site (a unique identifier, whose length is between DomainConstraints.MinSiteName and DomainConstraints.MaxSiteName chars). |
| *timezone* | The timezone, an integer between DomainConstraints.MinTimeZone and DomainConstraints.MaxTimeZone (inclusive). |
| *sessionExpirationTimeInSeconds* | Session timeout, in seconds, a positive number. |
| *minimumBidIncrement* | Minimum bid increment, a positive number. |

**Exceptions**

| | |
|---|---|
| *NameAlreadyInUseException* | Thrown if the name of the site is already in use as name of an existing site. |
| *ArgumentNullException* | If *connectionString* or *name* are null. |
| *ArgumentException* | If *name* is not null, but its length is (strictly) smaller than DomainConstraints.MinSiteName or (strictly) larger than DomainConstraints.MaxSiteName. |
| *ArgumentOutOfRangeException* | If *timezone* is (strictly) smaller than DomainConstraints.MinTimeZone or (strictly) larger than DomainConstraints.MaxTimeZone or if *sessionExpirationTimeInSeconds* or *minimumBidIncrement* are not positive. |
| *UnavailableDbException* | The connection string is (non-null but) malformed, the DB server is not responding or returns an unexpected error. |

#### 6.11.2.2 IEnumerable<string> GetSiteNames ( string *connectionString* )

Yields the managed sites.

**Returns**

The names of the managed sites.

**Parameters**

| | |
|---|---|
| *connectionString* | The connection string to an existing Microsoft SQL Server DB. |

**Exceptions**

| | |
|---|---|
| *ArgumentNullException* | If *connectionString* is null. |
| *UnavailableDbException* | If *connectionString* is (non-null but) malformed, the DB server is not responding or returns an unexpected error. |

#### 6.11.2.3 int GetTheTimezoneOf ( string *connectionString*, string *name* )

Return the timezone of the specified site.

**Parameters**

| | |
|---|---|
| *connectionString* | The connection string. |
| *name* | The name of the site. |

**Returns**

The timezone of the site.

**Exceptions**

| | |
|---|---|
| *UnavailableDbException* | The connection string is (non-null but) malformed, the DB server is not responding or returns an unexpected error. |
| *ArgumentNullException* | If *connectionString* or *name* are null. |
| *ArgumentException* | If *name* is not null, but its length is (strictly) smaller than DomainConstraints.MinSiteName or (strictly) larger than DomainConstraints.MaxSiteName. |
| *InexistentNameException* | If *name* is a non-null string of the correct length, but the corresponding site is not present in the DB. |

#### 6.11.2.4 ISite LoadSite ( string *connectionString*, string *name*, IAlarmClock *alarmClock* )

Yields the ISite object corresponding to an existing Site.

**Parameters**

| | |
|---|---|
| *connectionString* | The connection string. |
| *name* | The name of the site. |
| *alarmClock* | The time and alarm provider for the site. |

**Returns**

A new instance for the site.

**Exceptions**

| | |
|---|---|
| *UnavailableDbException* | The connection string is (non-null but) malformed, the DB server is not responding or returns an unexpected error. |
| *ArgumentNullException* | If *connectionString* or *name* or *alarmClock* are null. |
| *ArgumentException* | If *alarmClock* is not null but its timezone is not equal to the one of the site to be loaded, or if *name* is not null, but its length is (strictly) smaller than DomainConstraints.MinSiteName or (strictly) larger than DomainConstraints.MaxSiteName. |
| *InexistentNameException* | If *name* is a non-null string of the correct length, but the corresponding site is not present in the DB. |

#### 6.11.2.5 void Setup ( string *connectionString* )

Creates a new DB (dropping existing previous version, if any), and initialize it with all the necessary DB objects for the component.

**Parameters**

| | |
|---|---|
| *connectionString* | A valid connection string for a Microsoft SQL Server DB. |

**Exceptions**

| | |
|---|---|
| *ArgumentNullException* | If *connectionString* is null. |
| *UnavailableDbException* | If *connectionString* is (non-null but) malformed, the DB server is not responding or returns an unexpected error. |

## 6.12 IUser Interface Reference

The users of the auction system. Equals on users must be true iff the two users belong to the same site and have the same Username.

### Public Member Functions

- IEnumerable< IAuction > WonAuctions ()

  *Yields the auctions won by the user.*
- void Delete ()

  *Disposes of the user and all its resources.*

### Properties

- string Username `[get]`

  *Gets the unique key used to identify the user of a specific site in the system.*

### 6.12.1 Detailed Description

The users of the auction system. Equals on users must be true iff the two users belong to the same site and have the same Username.

### 6.12.2 Member Function Documentation

#### 6.12.2.1 void Delete ( )

Disposes of the user and all its resources.

Users cannot be deleted if they are owners or winners of still open auctions. Thus, a call to Delete on a user who is owner or winner of auctions yet to be adjudicated must throw `InvalidOperationException` Ended owned auctions are disposed of, if any. Ended won auctions are updated, and the information of the winner is removed, if any.

#### 6.12.2.2 IEnumerable<IAuction> WonAuctions ( )

Yields the auctions won by the user.

**Returns**

The auctions won by the user (that is, all the ended auctions of this site where this user is the highest bidder).

### 6.12.3 Property Documentation

#### 6.12.3.1 string Username [get]

Gets the unique key used to identify the user of a specific site in the system.

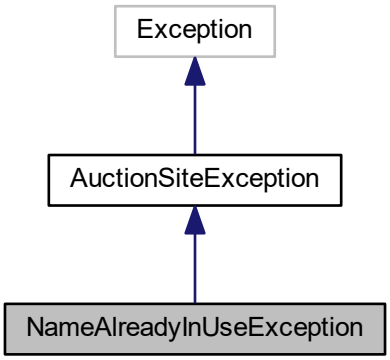
It is a string of at least DomainConstraints.MinUserName, and at most DomainConstraints.MaxUserName characters.

The same username may be used by different users on different sites.

## 6.13 NameAlreadyInUseException Class Reference

Thrown to notify the attempt to create an entity whose name is already in use. For instance, a site with the same name of another site, or a user with the same name of another user on the same site.

Inheritance diagram for NameAlreadyInUseException:




### Public Member Functions

- NameAlreadyInUseException (string name)
- NameAlreadyInUseException (string name, string message)
- NameAlreadyInUseException (string name, string message, Exception inner)
- NameAlreadyInUseException (SerializationInfo info, StreamingContext context)

### Properties

- string Name `[get]`

### Additional Inherited Members

### 6.13.1 Detailed Description

Thrown to notify the attempt to create an entity whose name is already in use. For instance, a site with the same name of another site, or a user with the same name of another user on the same site.

### 6.13.2 Constructor & Destructor Documentation

#### 6.13.2.1 NameAlreadyInUseException ( string *name* )

#### 6.13.2.2 NameAlreadyInUseException ( string *name,* string *message* )

#### 6.13.2.3 NameAlreadyInUseException ( string *name,* string *message,* Exception *inner* )

#### 6.13.2.4 NameAlreadyInUseException ( SerializationInfo *info,* StreamingContext *context* )

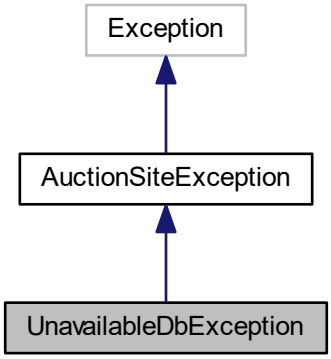
### 6.13.3 Property Documentation

#### 6.13.3.1 string Name [get]

## 6.14 UnavailableDbException Class Reference

Thrown to notify that no connection with the database has been established.

Inheritance diagram for UnavailableDbException:




### Public Member Functions

- UnavailableDbException ()
- UnavailableDbException (string message)
- UnavailableDbException (string message, Exception inner)
- UnavailableDbException (SerializationInfo info, StreamingContext context)

### Additional Inherited Members

### 6.14.1 Detailed Description

Thrown to notify that no connection with the database has been established.

### 6.14.2 Constructor & Destructor Documentation

#### 6.14.2.1 UnavailableDbException ( )

#### 6.14.2.2 UnavailableDbException ( string *message* )

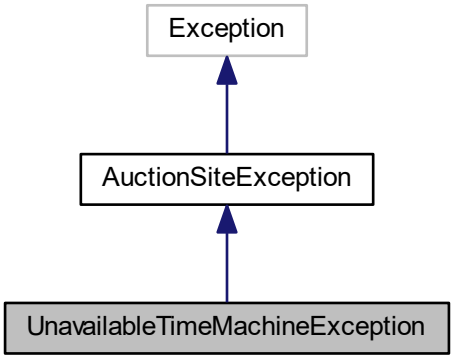
#### 6.14.2.3 UnavailableDbException ( string *message,* Exception *inner* )

#### 6.14.2.4 UnavailableDbException ( SerializationInfo *info,* StreamingContext *context* )

## 6.15 UnavailableTimeMachineException Class Reference

Thrown to notify the attempt to create an auction with expiration date in the past.

Inheritance diagram for UnavailableTimeMachineException:




### Public Member Functions

- UnavailableTimeMachineException ()
- UnavailableTimeMachineException (string message)
- UnavailableTimeMachineException (string message, Exception inner)
- UnavailableTimeMachineException (SerializationInfo info, StreamingContext context)

### Additional Inherited Members

### 6.15.1 Detailed Description

Thrown to notify the attempt to create an auction with expiration date in the past.

### 6.15.2 Constructor & Destructor Documentation

#### 6.15.2.1 UnavailableTimeMachineException ( )

#### 6.15.2.2 UnavailableTimeMachineException ( string *message* )

#### 6.15.2.3 UnavailableTimeMachineException ( string *message,* Exception *inner* )

#### 6.15.2.4 UnavailableTimeMachineException ( SerializationInfo *info,* StreamingContext *context* )