Black Hole Core ORM 6.0.0

Generated by Doxygen 1.9.6

# BlackHole.Core Namespace

# BlackHole.Core.BHDataProvider

Contains build in methods that generate sql commands

### BlackHole.Core.IBHDataProvider

Interface of BHDataProvider. It is required in order to use BHDataProvider methods

### BlackHole.Core.BHConnection

Allows to use custom commands, scalars and queries

#### BlackHole.Core.IBHConnection

Interface of BHConnection. It is automatically injected with Dependency Injection

# BlackHole.Core.BHParameters

Dynamic Parameters Class, for custom commands. Is is also used by Build-in methods

#### BlackHole.Core.BHTransaction

Transaction Class. It creates a connection and a transaction that can be passed to the methods

## BlackHole.Core.BHExtensions

Extension methods for the BHDataProvider that are used to build a Joins View

### BlackHole.Core.BHViewStorage

A Class that contains the functionality to execute stored views

# BlackHole.Core.IBHViewStorage

An Interface that is required, in order to use BHViewStorage

# **BlackHole.Configuration Namespace**

# BlackHole.Configuration.BlackHoleConfiguration

Static Class that contains the Extension method SuperNova, that is used to initialize BlackHole

## BlackHole.Configuration.BlackHoleSettings

A Class that contains all the configuration classed for the BlackHole

### BlackHole.Configuration.ConnectionSettings

Contains the connection string and methods to choose the type of the database

### BlackHole.Configuration.ConnectionAdditionalSettings

Additional Advanced Settings for using specific namespaces or additional assemblies

# BlackHole.Configuration.EntitiesWithNamespace

Part of BlackHole.Configuration.ConnectionAdditionalSettings to specify Entities namespaces

## BlackHole.Configuration.ServicesWithNamespace

Part of BlackHole.Configuration.ConnectionAdditionalSettings to specify Services namespaces

## BlackHole.Configuration.AssembliesUsed

Part of BlackHole.Configuration.ConnectionAdditionalSettings to specify Assemblies

### BlackHole.Configuration.DataPathSettings

Settings for the Path where BlackHole will store the logs and the Sqlite. Also settings for the logger behavior

BlackHole.Conf	iguration	Namespace
----------------	-----------	-----------

# **BlackHole.Entities Namespace**

# BlackHole.Entities.BlackHoleEntity

Make a class inherit from this, to be recognized as an Entity

### BlackHole.Entities.BlackHoleDto

Make a class inherit from this to be recognized as Data Transfer Object

### BlackHole.Entities.ForeignKey

An attribute that is used on a Property of a class to set a Foreign key constraint on that column

# BlackHole.Entities.NotNullable

An attribute that is used on a Property of a class to set a Not Null constraint on that column

# BlackHole.Entities.VarCharSize

An attribute that is used on a string Property of a class to set the character size of the column

## BlackHole.Entities.UseActivator

An attribute that is used on an Entity class to take advantage of the IsActive column

# **BlackHole.Services Namespace**

# BlackHole.Services.BlackHoleSingleton

Make a class inherit from this, to be recognized as a Singleton Service

# BlackHole.Services.BlackHoleScoped

Make a class inherit from this to be recognized as a Scoped Service

# BlackHole.Services.BlackHoleTransient

Make a class inherit from this to be recognized as a Transient Service

# **Hierarchical Index**

# 5.1 Class Hierarchy

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

BlackHole.Configuration.AssembliesUsed	?
BlackHole.Core.BHExtensions	?
BlackHole.Core.BHParameters	?
BlackHole.Core.BHTransaction	?
BlackHole.Configuration.BlackHoleConfiguration	?
$\label{eq:blackHole} \textit{BlackHoleDto} < G > \dots \dots \dots \dots \dots \dots \dots \dots \qquad \textbf{?}$	?
$\label{eq:blackHoleEntity} \textbf{BlackHoleEntity} < \textbf{G} > \dots $	?
BlackHole.Services.BlackHoleScoped	?
BlackHole.Configuration.BlackHoleSettings	?
BlackHole.Services.BlackHoleSingleton	?
BlackHole.Services.BlackHoleTransient	?
BlackHole.Configuration.ConnectionAdditionalSettings	?
BlackHole.Configuration.ConnectionSettings	
BlackHole.Configuration.DataPathSettings	
BlackHole.Configuration.EntitiesWithNamespace	?
BlackHole.Entities.ForeignKey	
BlackHole.Core.IBHConnection	?
BlackHole.Core.BHConnection	?
$BlackHole.Core.IBHDataProvider < T, G > \dots \dots$	?
BlackHole.Core.BHDataProvider< T, G >	?
BlackHole.Core.IBHViewStorage	?
BlackHole.Core.BHViewStorage	?
	?
	· ?
	· ?
	?

10 Hierarchical Index

# **Data Structure Index**

# 6.1 Data Structures

Here are the data structures with brief descriptions:

BlackHole.Configuration.AssembliesUsed	7
BlackHole.Core.BHConnection	?
BlackHole.Core.BHDataProvider< T, G >	?
BlackHole.Core.BHExtensions	
Queries for the Table Joins Feature	?
BlackHole.Core.BHParameters	?
BlackHole.Core.BHTransaction	?
BlackHole.Core.BHViewStorage	?
BlackHole.Configuration.BlackHoleConfiguration	?
BlackHole.Entities.BlackHoleDto< G >	
Is Required to use Mapping on DTO, Views and Joins Functionality	?
BlackHole.Entities.BlackHoleEntity< G >	
Black Hole Entity. The table in database is based on this	?
BlackHole.Services.BlackHoleScoped	?
BlackHole.Configuration.BlackHoleSettings	?
BlackHole.Services.BlackHoleSingleton	?
BlackHole.Services.BlackHoleTransient	3
BlackHole.Configuration.ConnectionAdditionalSettings	3
BlackHole.Configuration.ConnectionSettings	3
BlackHole.Configuration.DataPathSettings	3
BlackHole.Configuration.EntitiesWithNamespace	3
BlackHole.Entities.ForeignKey	3
BlackHole.Core.IBHConnection	3
BlackHole.Core.IBHDataProvider< T, G >	
Provides all the functionality to communicate with the database with more than 100 methods and	
commands	3
BlackHole.Core.IBHViewStorage	3
BlackHole.Entities.NotNullable	3
BlackHole.Configuration.ServicesWithNamespace	?
BlackHole.Entities.UseActivator	?
BlackHole Entities VarCharSize	7

12 Data Structure Index

# **Namespace Documentation**

# 7.1 BlackHole Namespace Reference

# **Namespaces**

- namespace Configuration
- namespace Core
- namespace Entities
- namespace Services

# 7.2 BlackHole.Configuration Namespace Reference

## **Data Structures**

- · class AssembliesUsed
- class BlackHoleConfiguration
- class BlackHoleSettings
- · class ConnectionAdditionalSettings
- class ConnectionSettings
- class DataPathSettings
- · class EntitiesWithNamespace
- · class ServicesWithNamespace

# 7.2.1 Detailed Description

Contains all the required classes and methods to define the database type, the entities, the services, the namespaces and the assemblies that will be used. Also contains configuration for the automatic logging.

# BlackHole.Configuration.BlackHoleConfiguration

Static Class that contains the Extension method SuperNova, that is used to initialize BlackHole

# BlackHole.Configuration.BlackHoleSettings

A Class that contains all the configuration classed for the BlackHole

#### BlackHole.Configuration.ConnectionSettings

Contains the connection string and methods to choose the type of the database

### BlackHole.Configuration.ConnectionAdditionalSettings

Additional Advanced Settings for using specific namespaces or additional assemblies

### BlackHole.Configuration.EntitiesWithNamespace

Part of BlackHole.Configuration.ConnectionAdditionalSettings to specify Entities namespaces

### BlackHole.Configuration.ServicesWithNamespace

Part of BlackHole.Configuration.ConnectionAdditionalSettings to specify Services namespaces

#### BlackHole.Configuration.AssembliesUsed

Part of BlackHole.Configuration.ConnectionAdditionalSettings to specify Assemblies

### BlackHole.Configuration.DataPathSettings

Settings for the Path where BlackHole will store the logs and the Sqlite. Also settings for the logger behavior

# 7.3 BlackHole.ConnectionProvider Namespace Reference

# 7.4 BlackHole.Core Namespace Reference

# **Data Structures**

- class BHConnection
- · class BHDataProvider
- · class BHExtensions
- · class BHParameters
- class BHTransaction
- class BHViewStorage
- interface IBHConnectioninterface IBHDataProvider
- interface IBHViewStorage

# 7.4.1 Detailed Description

This Namespace contains all the required functionality to make your application, communicate with the database

# BlackHole.Core.BHDataProvider

Contains build in methods that generate sql commands

## BlackHole.Core.IBHDataProvider

Interface of BHDataProvider. It is required in order to use BHDataProvider methods

# BlackHole.Core.BHConnection

Allows to use custom commands, scalars and queries

BlackHole.Core.IBHConnection

Interface of BHConnection. It is automatically injected with Dependency Injection

#### BlackHole.Core.BHParameters

Dynamic Parameters Class, for custom commands. Is is also used by Build-in methods

### BlackHole.Core.BHTransaction

Transaction Class. It creates a connection and a transaction that can be passed to the methods

#### BlackHole.Core.BHExtensions

Extension methods for the BHDataProvider that are used to build a Joins View

# BlackHole.Core.BHViewStorage

A Class that contains the functionality to execute stored views

# BlackHole.Core.IBHViewStorage

An Interface that is required, in order to use BHViewStorage

# 7.5 BlackHole.CoreSupport Namespace Reference

# 7.6 BlackHole.DataProviders Namespace Reference

# 7.7 BlackHole.Entities Namespace Reference

# **Data Structures**

- class BlackHoleDto
- class BlackHoleEntity
- class ForeignKey
- · class NotNullable
- class UseActivator
- · class VarCharSize

# 7.7.1 Detailed Description

This Namespace contains all the required classes to turn your Entities into Tables with columns and constraints

### BlackHole.Entities.BlackHoleEntity

Make a class inherit from this, to be recognized as an Entity

## BlackHole.Entities.BlackHoleDto

Make a class inherit from this to be recognized as Data Transfer Object

#### BlackHole.Entities.ForeignKey

An attribute that is used on a Property of a class to set a Foreign key constraint on that column

### BlackHole.Entities.NotNullable

An attribute that is used on a Property of a class to set a Not Null constraint on that column

### BlackHole.Entities.VarCharSize

An attribute that is used on a string Property of a class to set the character size of the column

### BlackHole.Entities.UseActivator

An attribute that is used on an Entity class to take advantage of the IsActive column

# 7.8 BlackHole.Enums Namespace Reference

# 7.9 BlackHole.ExecutionProviders Namespace Reference

# 7.10 BlackHole.Internal Namespace Reference

# 7.11 BlackHole.Logger Namespace Reference

# 7.12 BlackHole.Services Namespace Reference

# **Data Structures**

- · class BlackHoleScoped
- · class BlackHoleSingleton
- class BlackHoleTransient

# 7.12.1 Detailed Description

Contains classes that help your services registered automatically in the IServiceCollection on startup

# BlackHole.Services.BlackHoleSingleton

Make a class inherit from this, to be recognized as a Singleton Service

## BlackHole.Services.BlackHoleScoped

Make a class inherit from this to be recognized as a Scoped Service

### BlackHole.Services.BlackHoleTransient

Make a class inherit from this to be recognized as a Transient Service

# 7.13 BlackHole.Statics Namespace Reference

# **Data Structure Documentation**

# 8.1 BlackHole.Configuration.AssembliesUsed Class Reference

# **Public Member Functions**

void UseOtherAssembly (Assembly otherAssembly)

# 8.1.1 Detailed Description

Settings for the target Assemblies.
Part of BlackHole.Configuration.ConnectionAdditionalSettings

### 8.1.2 Member Function Documentation

# 8.1.2.1 UseOtherAssembly()

```
void BlackHole.Configuration.AssembliesUsed.UseOtherAssembly ( {\tt Assembly}\ other {\tt Assembly}\ )
```

Scans a specified assembly for BlackHole Entities and Services in the previous specified Namespaces and uses only them.

#### **Parameters**

otherAssembly Full Assembly

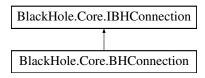
# Example:

services.SuperNova(settings => settings.AddDatabase(connection => connection.UseMySql(connection.UseMySql(connection.UseEntitiesInNamespaces(List<string> NamespacesFullNames).AddServicesFromNamespacesFullNames).

.UseOtherAssembly(Assembly assembly)))

# 8.2 BlackHole.Core.BHConnection Class Reference

Inheritance diagram for BlackHole.Core.BHConnection:



### **Public Member Functions**

- BHConnection ()
- G? ExecuteScalar < G > (string commandText)
- G? ExecuteScalar < G > (string commandText, BHParameters parameters)
- G? ExecuteScalar < G > (string commandText, object parametersObject)
- G? ExecuteScalar < G > (string commandText, BHParameters parameters, BHTransaction bHTransaction)
- G? ExecuteScalar < G > (string commandText, BHTransaction bHTransaction)
- G? ExecuteScalar < G > (string commandText, object parametersObject, BHTransaction bHTransaction)
- async Task< G?> ExecuteScalarAsync< G > (string commandText)
- async Task< G?> ExecuteScalarAsync< G > (string commandText, BHParameters parameters)
- async Task< G?> ExecuteScalarAsync< G > (string commandText, object parametersObject)
- async Task< G?> ExecuteScalarAsync< G > (string commandText, BHTransaction bHTransaction)
- async Task< G?> ExecuteScalarAsync< G > (string commandText, BHParameters parameters, BHTransaction bHTransaction)
- async Task< G?> ExecuteScalarAsync< G > (string commandText, object parametersObject, BHTransaction bHTransaction)
- bool JustExecute (string commandText)
- bool JustExecute (string commandText, BHParameters parameters)
- bool JustExecute (string commandText, object parametersObject)
- bool JustExecute (string commandText, BHTransaction bHTransaction)
- bool JustExecute (string commandText, BHParameters parameters, BHTransaction bHTransaction)
- bool JustExecute (string commandText, object parametersObject, BHTransaction bHTransaction)
- async Task< bool > JustExecuteAsync (string commandText)
- async Task< bool > JustExecuteAsync (string commandText, BHParameters parameters)
- async Task< bool > JustExecuteAsync (string commandText, object parametersObject)
- async Task< bool > JustExecuteAsync (string commandText, BHTransaction bHTransaction)
- async Task< bool > JustExecuteAsync (string commandText, BHParameters parameters, BHTransaction bHTransaction)
- async Task< bool > JustExecuteAsync (string commandText, object parametersObject, BHTransaction b← HTransaction)
- List< T > Query< T > (string commandText)
- List< T > Query< T > (string commandText, BHParameters parameters)
- List< T > Query< T > (string commandText, object parametersObject)
- List< T > Query< T > (string commandText, BHTransaction bHTransaction)
- List< T > Query< T > (string commandText, BHParameters parameters, BHTransaction bHTransaction)
- List< T > Query< T > (string commandText, object parametersObject, BHTransaction bHTransaction)
- async Task< List< T > > QueryAsync< T > (string commandText)

- async Task< List< T > > QueryAsync< T > (string commandText, BHParameters parameters)
- async Task< List< T >> QueryAsync< T > (string commandText, object parametersObject)
- async Task< List< T >> QueryAsync< T > (string commandText, BHTransaction bHTransaction)
- async Task< List< T > > QueryAsync< T> (string commandText, BHParameters parameters, BHTransaction bHTransaction)
- async Task< List< T >> QueryAsync< T > (string commandText, object parametersObject, BHTransaction bHTransaction)
- T? QueryFirst< T > (string commandText)
- T? QueryFirst< T > (string commandText, BHParameters parameters)
- T? QueryFirst< T > (string commandText, object parametersObject)
- T? QueryFirst< T > (string commandText, BHTransaction bHTransaction)
- T? QueryFirst< T > (string commandText, BHParameters parameters, BHTransaction bHTransaction)
- T? QueryFirst< T > (string commandText, object parametersObject, BHTransaction bHTransaction)
- async Task< T?> QueryFirstAsync< T > (string commandText)
- async Task< T?> QueryFirstAsync< T > (string commandText, BHParameters parameters)
- async Task< T?> QueryFirstAsync< T > (string commandText, object parametersObject)
- async Task< T?> QueryFirstAsync< T > (string commandText, BHTransaction bHTransaction)
- async Task< T?> QueryFirstAsync< T > (string commandText, BHParameters parameters, BHTransaction bHTransaction)
- async Task< T?> QueryFirstAsync< T > (string commandText, object parametersObject, BHTransaction bHTransaction)
- G? ExecuteScalar < G > (string commandText)
- G? ExecuteScalar < G > (string commandText, BHParameters parameters)
- G? ExecuteScalar < G > (string commandText, object parametersObject)
- G? ExecuteScalar < G > (string commandText, BHParameters parameters, BHTransaction bHTransaction)
- G? ExecuteScalar < G > (string commandText, BHTransaction bHTransaction)
- G? ExecuteScalar < G > (string commandText, object parametersObject, BHTransaction bHTransaction)
- Task< G?> ExecuteScalarAsync< G > (string commandText)
- Task< G?> ExecuteScalarAsync< G > (string commandText, BHParameters parameters)
- Task< G?> ExecuteScalarAsync< G > (string commandText, object parametersObject)
- Task< G?> ExecuteScalarAsync< G > (string commandText, BHTransaction bHTransaction)
- Task< G?> ExecuteScalarAsync< G > (string commandText, BHParameters parameters, BHTransaction bHTransaction)
- Task< G?> ExecuteScalarAsync< G > (string commandText, object parametersObject, BHTransaction b← HTransaction)
- bool JustExecute (string commandTex)
- bool JustExecute (string commandText, BHParameters parameters)
- bool JustExecute (string commandText, object parametersObject)
- bool JustExecute (string commandText, BHTransaction bHTransaction)
- bool JustExecute (string commandText, BHParameters parameters, BHTransaction bHTransaction)
- bool JustExecute (string commandText, object parametersObject, BHTransaction bHTransaction)
- Task< bool > JustExecuteAsync (string commandText)
- Task< bool > JustExecuteAsync (string commandText, BHParameters parameters)
- Task< bool > JustExecuteAsync (string commandText, object parametersObject)
- Task< bool > JustExecuteAsync (string commandText, BHTransaction bHTransaction)
- Task< bool > JustExecuteAsync (string commandText, BHParameters parameters, BHTransaction b $\leftarrow$  HTransaction)
- Task< bool > JustExecuteAsync (string commandText, object parametersObject, BHTransaction b← HTransaction)
- T? QueryFirst< T > (string commandText)
- T? QueryFirst< T > (string commandText, BHParameters parameters)
- T? QueryFirst < T > (string commandText, object parametersObject)
- T? QueryFirst< T > (string commandText, BHTransaction bHTransaction)

- T? QueryFirst< T > (string commandText, BHParameters parameters, BHTransaction bHTransaction)
- T? QueryFirst < T > (string commandText, object parametersObject, BHTransaction bHTransaction)
- List< T > Query< T > (string commandText)
- List< T > Query< T > (string commandText, BHParameters parameters)
- List< T > Query< T > (string commandText, object parametersObject)
- List< T > Query< T > (string commandText, BHTransaction bHTransaction)
- List< T > Query< T > (string commandText, BHParameters parameters, BHTransaction bHTransaction)
- List< T > Query< T > (string commandText, object parametersObject, BHTransaction bHTransaction)
- Task< T?> QueryFirstAsync< T > (string commandText)
- Task< T?> QueryFirstAsync< T > (string commandText, BHParameters parameters)
- Task< T?> QueryFirstAsync< T > (string commandText, object parametersObject)
- Task< T?> QueryFirstAsync< T > (string commandText, BHTransaction bHTransaction)
- Task< T?> QueryFirstAsync< T > (string commandText, object parametersObject, BHTransaction b← HTransaction)
- Task< List< T >> QueryAsync< T > (string commandText)
- Task< List< T > > QueryAsync< T > (string commandText, BHParameters parameters)
- Task < List < T > > QueryAsync < T > (string commandText, object parametersObject)
- Task< List< T > > QueryAsync< T > (string commandText, BHTransaction bHTransaction)
- Task< List< T > > QueryAsync< T > (string commandText, BHParameters parameters, BHTransaction bHTransaction)
- Task< List< T >> QueryAsync< T > (string commandText, object parametersObject, BHTransaction b← HTransaction)

# 8.2.1 Detailed Description

A Class that gives all the required methods to perform custom sql commands

It's already registered in the ServiceCollection and it can be used to your services with Dependency Injection

The connection is automatically generated and disposed after each execution

## 8.2.2 Constructor & Destructor Documentation

# 8.2.2.1 BHConnection()

```
BlackHole.Core.BHConnection.BHConnection ( )
```

An Interface that gives all the required methods to perform custom sql commands.

It's already registered in the ServiceCollection and it can be used to your services with Dependency Injection

The connection is automatically generated and disposed after each execution

# Example:

```
IBHConnection connection = new BHConnection();
```

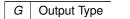
# 8.2.3 Member Function Documentation

# 8.2.3.1 ExecuteScalar< G >() [1/6]

```
G? BlackHole.Core.BHConnection.ExecuteScalar< G > ( string commandText )
```

Classic Execute Scalar

**Template Parameters** 



#### **Parameters**

```
commandText | Command Text
```

Returns

The First Value of the Result

# Example:

```
IBHConnection connection = new BHConnection();
int result = connection.ExecuteScalar<int>(string command);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

# 8.2.3.2 ExecuteScalar < G >() [2/6]

```
G? BlackHole.Core.BHConnection.ExecuteScalar< G > ( string commandText, BHParameters parameters)
```

Classic Execute Scalar with BHParameters

**Template Parameters** 

```
G Output Type
```

#### **Parameters**

commandText	Command Text
parameters	BHParameters Class, populated with black hole parameters

# Returns

The First Value of the Result

# Example:

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();
parameters.Add("Price", 5.5);
int result = connection.ExecuteScalar<int>(string command, parameters);
```

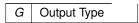
**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

# 8.2.3.3 ExecuteScalar < G >() [3/6]

```
G? BlackHole.Core.BHConnection.ExecuteScalar<br/>< G > ( string \ commandText, object parametersObject )
```

Classic Execute Scalar with Object as Parameters

# **Template Parameters**



#### **Parameters**

	commandText	Command Text	
Ī	parametersObject	Class with properties as Parameters	

#### Returns

The First Value of the Result

# Example:

IBHConnection connection = new BHConnection();

```
var someClass = new SomeClass{
    Price = 5.5
}
int result = connection.ExecuteScalar<int>(string command, someClass);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

# 8.2.3.4 ExecuteScalar< G >() [4/6]

Transaction. Classic Execute Scalar with BHParameters

### **Template Parameters**

```
G Output Type
```

# **Parameters**

commandText	Command Text
parameters	BHParameters Class, populated with black hole parameters
bHTransaction	BHTransaction Class, contains connection and transaction

# Returns

The First Value of the Result

# Example:

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();

parameters.Add("Price", 5.5);

using(BHTransaction transaction = new BHTransaction())
{
   int result = connection.ExecuteScalar<int>(string command, parameters, transaction);
}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

# 8.2.3.5 ExecuteScalar< G >() [5/6]

```
G? BlackHole.Core.BHConnection.ExecuteScalar<br/>< G > ( string \ commandText, BHTransaction \ bHTransaction )
```

Transaction. Classic Execute Scalar

**Template Parameters** 

```
G Output Type
```

#### **Parameters**

commandText	Command Text
bHTransaction	BHTransaction Class, contains connection and transaction

# Returns

The First Value of the Result

# Example:

```
IBHConnection connection = new BHConnection();
using(BHTransaction transaction = new BHTransaction())
{
   int result = connection.ExecuteScalar<int>(string command, transaction);
}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

# 8.2.3.6 ExecuteScalar < G >() [6/6]

Transaction. Classic Execute Scalar with Object as Parameters

# **Template Parameters**

G Output Type

### **Parameters**

commandText	Command Text
parametersObject	Class with properties as Parameters
bHTransaction	BHTransaction Class, contains connection and transaction

# Returns

The First Value of the Result

### Example:

```
IBHConnection connection = new BHConnection();

var someClass = new SomeClass{
    Price = 5.5
}

using(BHTransaction transaction = new BHTransaction())
{
    int result = connection.ExecuteScalar<int>(string command, someClass, transaction);
}
```

Tip: For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text

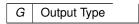
Implements BlackHole.Core.IBHConnection.

# 8.2.3.7 ExecuteScalarAsync< G >() [1/6]

```
async Task< G?> BlackHole.Core.BHConnection.ExecuteScalarAsync< G > ( string \ commandText \ )
```

Asyncronous. Classic Execute Scalar

# **Template Parameters**



# **Parameters**

commandText	Command Text
-------------	--------------

# Returns

The First Value of the Result

# Example:

```
IBHConnection connection = new BHConnection();
int result = await connection.ExecuteScalarAsync<int>(string command);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

# 8.2.3.8 ExecuteScalarAsync< G >() [2/6]

```
async Task< G?> BlackHole.Core.BHConnection.ExecuteScalarAsync< G > ( string \ commandText, \\ BHParameters \ parameters )
```

Asyncronous. Classic Execute Scalar with BHParameters

# **Template Parameters**



#### **Parameters**

commandText	Command Text
parameters	BHParameters Class, populated with black hole parameters

### Returns

The First Value of the Result

### Example:

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();
parameters.Add("Price", 5.5);
int result = await connection.ExecuteScalarAsync<int>(string command, parameters);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

### 8.2.3.9 ExecuteScalarAsync< G >() [3/6]

```
async Task<br/>< G?> BlackHole.Core.BHConnection.ExecuteScalarAsync<br/>< G > ( string commandText, object parametersObject)
```

Asyncronous. Classic Execute Scalar with Object as Parameters

# **Template Parameters**

G	Output Type

#### **Parameters**

commandText	Command Text
parametersObject	Class with properties as Parameters

# Returns

The First Value of the Result

# Example:

```
IBHConnection connection = new BHConnection();
var someClass = new SomeClass{
    Price = 5.5
}
int result = await connection.ExecuteScalarAsync<int>(string command, someClass);
```

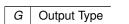
**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

# 8.2.3.10 ExecuteScalarAsync< G > () [4/6]

```
async Task<br/>< G?> BlackHole.Core.BHConnection.ExecuteScalarAsync<br/>< G > ( string\ commandText, BHTransaction\ bHTransaction\ )
```

Asyncronous. Transaction. Classic Execute Scalar

# **Template Parameters**



## **Parameters**

commandText	Command Text
bHTransaction	BHTransaction Class, contains connection and transaction

#### Returns

The First Value of the Result

# Example:

```
IBHConnection connection = new BHConnection();
using(BHTransaction transaction = new BHTransaction())
{
   int result = await connection.ExecuteScalarAsync<int>(string command, transaction);
}
```

 $\textbf{Tip:} \ \ \text{For Oracle and Postgres} \ \ , \ \ \text{Double Quotes are required for the Table and Column Names in your command text}$ 

Implements BlackHole.Core.IBHConnection.

# 8.2.3.11 ExecuteScalarAsync< G >() [5/6]

Asyncronous. Transaction. Classic Execute Scalar with BHParameters

# **Template Parameters**

```
G Output Type
```

# **Parameters**

ſ	commandText	Command Text
	parameters	BHParameters Class, populated with black hole parameters
Ì	bHTransaction	BHTransaction Class, contains connection and transaction

# Returns

The First Value of the Result

# Example:

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();
parameters.Add("Price", 5.5);
using(BHTransaction transaction = new BHTransaction()){
```

```
int result = await connection.ExecuteScalarAsync<int>(string command, parameters, transa)
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

# 8.2.3.12 ExecuteScalarAsync< G >() [6/6]

Asyncronous. Transaction. Classic Execute Scalar with Object as Parameters

### **Template Parameters**

-
---

#### **Parameters**

commandText	Command Text
parametersObject	Class with properties as Parameters
bHTransaction	BHTransaction Class, contains connection and transaction

## Returns

The First Value of the Result

# Example:

```
IBHConnection connection = new BHConnection();

var someClass = new SomeClass{
    Price = 5.5
}

using(BHTransaction transaction = new BHTransaction())
{
    int result = await connection.ExecuteScalarAsync<int>(string command, someClass, transaction)}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

## 8.2.3.13 JustExecute() [1/6]

```
bool BlackHole.Core.BHConnection.JustExecute ( string\ commandText\ )
```

Classic Execute without output.

**Parameters** 

```
commandText | Command Text
```

### Returns

Success Boolean

### Example:

```
IBHConnection connection = new BHConnection();
bool success = connection.JustExecute(string command);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

# 8.2.3.14 JustExecute() [2/6]

Classic Execute with BHParameters.

#### **Parameters**

commandText	Command Text
parameters	BHParameters Class, populated with black hole parameters

# Returns

Success Boolean

### Example:

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();
parameters.Add("Price", 5.5);
```

```
bool success = connection.JustExecute(string command, parameters);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

# 8.2.3.15 JustExecute() [3/6]

Classic Execute with Object as Parameters.

#### **Parameters**

commandText	Command Text
parametersObject	Class with properties as Parameters

#### Returns

Success Boolean

# Example:

```
IBHConnection connection = new BHConnection();

var someClass = new SomeClass{
    Price = 5.5
}

bool success = connection.JustExecute(string command, someClass);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

# 8.2.3.16 JustExecute() [4/6]

Transaction. Classic Execute without output.

#### **Parameters**

commandText	Command Text	
bHTransaction	BHTransaction Class, contains connection and transaction	]

### Returns

Success Boolean

# Example:

```
IBHConnection connection = new BHConnection();
using(BHTransaction transaction = new BHTransaction())
{
    bool success = connection.JustExecute(string command, transaction);
}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

# 8.2.3.17 JustExecute() [5/6]

Transaction. Classic Execute without output.

# **Parameters**

commandText	Command Text
parameters	BHParameters Class, populated with black hole parameters
bHTransaction	BHTransaction Class, contains connection and transaction

## Returns

Success Boolean

# Example:

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();
parameters.Add("Price", 5.5);
```

```
using(BHTransaction transaction = new BHTransaction())
{
   bool success = connection.JustExecute(string command, parameters, transaction);
}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

## 8.2.3.18 JustExecute() [6/6]

Transaction. Classic Execute with Object as Parameters.

#### **Parameters**

commandText	Command Text
parametersObject	Class with properties as Parameters
bHTransaction	BHTransaction Class, contains connection and transaction

# Returns

Success Boolean

### Example:

```
IBHConnection connection = new BHConnection();

var someClass = new SomeClass{
    Price = 5.5
}

using(BHTransaction transaction = new BHTransaction())
{
    bool success = connection.JustExecute(string command, someClass, transaction);
}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

### 8.2.3.19 JustExecuteAsync() [1/6]

```
async Task<br/>< bool > BlackHole.Core.BHConnection.JustExecuteAsync ( string\ commandText\ )
```

Asyncronous. Classic Execute without output.

#### **Parameters**

```
commandText | Command Text
```

# Returns

Success Boolean

### Example:

```
IBHConnection connection = new BHConnection();
bool success = await connection.JustExecuteAsync(string command);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

### 8.2.3.20 JustExecuteAsync() [2/6]

Asyncronous. Classic Execute with BHParameters.

### **Parameters**

commandText	Command Text
parameters	BHParameters Class, populated with black hole parameters

#### Returns

Success Boolean

# Example:

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();
parameters.Add("Price", 5.5);
bool success = await connection.JustExecuteAsync(string command, parameters);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

### 8.2.3.21 JustExecuteAsync() [3/6]

Asyncronous. Classic Execute with Object as Parameters.

#### **Parameters**

commandText	Command Text
parametersObject	Class with properties as Parameters

#### Returns

Success Boolean

### Example:

```
IBHConnection connection = new BHConnection();

var someClass = new SomeClass{
    Price = 5.5
}

bool success = await connection.JustExecuteAsync(string command, someClass);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

#### 8.2.3.22 JustExecuteAsync() [4/6]

```
async Task<br/>< bool > BlackHole.Core.BHConnection.JustExecuteAsync ( string commandText, BHTransaction bHTransaction )
```

Asyncronous. Transaction. Classic Execute without output.

#### **Parameters**

commandText	Command Text
bHTransaction	BHTransaction Class, contains connection and transaction

#### Returns

Success Boolean

```
IBHConnection connection = new BHConnection();
using(BHTransaction transaction = new BHTransaction())
{
    bool success =await connection.JustExecuteAsync(string command, transaction);
}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

### 8.2.3.23 JustExecuteAsync() [5/6]

Asyncronous. Transaction. Classic Execute with BHParameters.

#### **Parameters**

commandText	Command Text
parameters	BHParameters Class, populated with black hole parameters
bHTransaction	BHTransaction Class, contains connection and transaction

### Returns

Success Boolean

## Example:

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();

parameters.Add("Price", 5.5);

using(BHTransaction transaction = new BHTransaction())
{
   bool success = await connection.JustExecuteAsync(string command, parameters, transaction}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

### 8.2.3.24 JustExecuteAsync() [6/6]

Asyncronous. Transaction. Classic Execute with Object as Parameters.

#### **Parameters**

commandText	Command Text
parametersObject	Class with properties as Parameters
bHTransaction	BHTransaction Class, contains connection and transaction

#### Returns

Success Boolean

## Example:

```
IBHConnection connection = new BHConnection();

var someClass = new SomeClass{
    Price = 5.5
}

using(BHTransaction transaction = new BHTransaction())
{
    bool success = await connection.JustExecuteAsync(string command, someClass, transaction)}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

## 8.2.3.25 Query < T >() [1/6]

```
List< T > BlackHole.Core.BHConnection.Query< T > ( string \ commandText \ )
```

A Query that returns all Lines of the Result as List.

## **Template Parameters**

T Output Type

### Returns

The First Line of the Result

### Example:

```
IBHConnection connection = new BHConnection();
List<Customer> customers = connection.Query<Customer>(string command);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

#### 8.2.3.26 Query < T >() [2/6]

```
List< T > BlackHole.Core.BHConnection.Query< T > ( string \ commandText, \\ BHParameters \ parameters )
```

A Query that takes BHParameters and returns all Lines of the Result as List.

## **Template Parameters**

```
T Output Type
```

## **Parameters**

commandText	Command Text	
parameters	BHParameters Class, populated with black hole parameters	

#### Returns

The First Line of the Result

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();
parameters.Add("FirstName", "Nick");
List<Customer> customers = connection.Query<Customer>(string command, parameters);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

### 8.2.3.27 Query < T >() [3/6]

```
List< T > BlackHole.Core.BHConnection.Query< T > ( string \ commandText, object \ parametersObject )
```

A Query that takes an Object as parameters and returns all Lines of the Result as List.

#### **Template Parameters**

```
T Output Type
```

#### **Parameters**

commandText	Command text
parametersObject	Class with properties as Parameters

#### Returns

The First Line of the Result

## Example:

```
IBHConnection connection = new BHConnection();

var someClass = new SomeClass{
    FirstName = "Nick"
};

List<Customer> customers = connection.Query<Customer>(string command, someClass);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

## 8.2.3.28 Query < T >() [4/6]

```
List< T > BlackHole.Core.BHConnection.Query< T > ( string\ commandText, BHTransaction\ bHTransaction\ )
```

Transaction. A Query that returns all Lines of the Result as List.

T Output Type
---------------

#### **Parameters**

commandText	Command Text	
bHTransaction	BHTransaction Class, contains connection and transaction	

### Returns

The First Line of the Result

## Example:

```
IBHConnection connection = new BHConnection();
using(BHTransaction transaction = new BHTransaction())
{
    List<Customer> customers = connection.Query<Customer>(string command, transaction);
}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

## 8.2.3.29 Query < T >() [5/6]

Transaction. A Query that takes BHParameters and returns all Lines of the Result as List.

#### **Template Parameters**

Τ	Output Type

#### **Parameters**

commandText	Command Text
parameters	BHParameters Class, populated with black hole parameters
bHTransaction	BHTransaction Class, contains connection and transaction

#### Returns

The First Line of the Result

### Example:

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();

parameters.Add("FirstName", "Nick");

using(BHTransaction transaction = new BHTransaction())
{
    List<Customer> customers = connection.Query<Customer>(string command, parameters, transaction)}
```

Tip: For Oracle and Postgres , Double Quotes are required for the Table and Column Names in your command text

Implements BlackHole.Core.IBHConnection.

### 8.2.3.30 Query < T >() [6/6]

Transaction. A Query that takes an Object as parameters and returns all Lines of the Result as List.

#### **Template Parameters**

T	Output Type
'	Output Type

### **Parameters**

commandText	Command Text
parametersObject	Class with properties as Parameters
bHTransaction	BHTransaction Class, contains connection and transaction

### Returns

The First Line of the Result

```
IBHConnection connection = new BHConnection();
var someClass = new SomeClass{
    Price = 5.5
```

```
}
using(BHTransaction transaction = new BHTransaction())
{
    List<Customer> customer = connection.Query<Customer>(string command, someClass, transact)
}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

## 8.2.3.31 QueryAsync< T >() [1/6]

```
async Task< List< T >> BlackHole.Core.BHConnection.QueryAsync< T > ( string commandText )
```

Asyncronous. A Query that returns all Lines of the Result as List.

#### **Template Parameters**

```
T Output List Type
```

### **Parameters**

```
commandText | Command Text
```

### Returns

List of Lines

### Example:

```
IBHConnection connection = new BHConnection();
List<Customer> customers = await connection.QueryAsync<Customer>(string command);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

## 8.2.3.32 QueryAsync< T >() [2/6]

```
async Task< List< T >> BlackHole.Core.BHConnection.QueryAsync< T > ( string commandText, BHParameters parameters)
```

Asyncronous. A Query that takes BHParameters and returns all Lines of the Result as List.

```
T Output List Type
```

#### **Parameters**

commandText	Command Text	
parameters	BHParameters Class, populated with black hole parameters	

## Returns

List of Lines

### Example:

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();

parameters.Add("FirstName", "Nick");

List<Customer> customers = await connection.QueryAsync<Customer>(string command, parameters)
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

## 8.2.3.33 QueryAsync< T >() [3/6]

```
async Task< List< T >> BlackHole.Core.BHConnection.QueryAsync< T > ( string commandText, object parametersObject)
```

Asyncronous. A Query that takes an Object as parameters and returns all Lines of the Result as List.

## **Template Parameters**

```
T Output List Type
```

### **Parameters**

commandText	Command Text
parametersObject	Class with properties as Parameters

### Returns

List of Lines

### Example:

```
IBHConnection connection = new BHConnection();

var someClass = new SomeClass{
    FirstName = "Nick"
};

List<Customer> customers = await connection.QueryAsync<Customer>(string command, someClass);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

## 8.2.3.34 QueryAsync< T >() [4/6]

```
async Task< List< T >> BlackHole.Core.BHConnection.QueryAsync< T > ( string commandText, BHTransaction bHTransaction )
```

Asyncronous. Transaction. A Query that returns all Lines of the Result as List.

## **Template Parameters**

```
T Output List Type
```

### **Parameters**

cor	mmandText	Command Text
bH	Transaction	BHTransaction Class, contains connection and transaction

#### Returns

List of Lines

### Example:

```
IBHConnection connection = new BHConnection();
using(BHTransaction transaction = new BHTransaction())
{
    List<Customer> customers = await connection.QueryAsync<Customer>(string command, transaction)}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

### 8.2.3.35 QueryAsync< T >() [5/6]

```
async Task< List< T >> BlackHole.Core.BHConnection.QueryAsync< T > ( string commandText, BHParameters parameters, BHTransaction bHTransaction )
```

Asyncronous. Transaction. A Query that takes BHParameters and returns all Lines of the Result as List.

#### **Template Parameters**

```
T Output List Type
```

#### **Parameters**

commandText	Command Text
parameters	BHParameters Class, populated with black hole parameters
bHTransaction	BHTransaction Class, contains connection and transaction

### Returns

List of Lines

### Example:

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();

parameters.Add("FirstName", "Nick");

using(BHTransaction transaction = new BHTransaction())
{
    List<Customer> customers = await connection.QueryAsync<Customer>(string command, parameter);
}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

### 8.2.3.36 QueryAsync< T >() [6/6]

```
async Task< List< T >> BlackHole.Core.BHConnection.QueryAsync< T > ( string commandText, object parametersObject, BHTransaction bHTransaction )
```

Asyncronous. Transaction. A Query that takes an Object as parameters and returns all Lines of the Result as List.

T Output List Type	
--------------------	--

#### **Parameters**

commandText	Command Text
parametersObject	Class with properties as Parameters
bHTransaction	BHTransaction Class, contains connection and transaction

#### Returns

List of Lines

## Example:

```
IBHConnection connection = new BHConnection();

var someClass = new SomeClass{
    Price = 5.5
}

using(BHTransaction transaction = new BHTransaction())
{
    List<Customer> customers = await connection.QueryAsync<Customer>(string command, someClass)
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

## 8.2.3.37 QueryFirst< T >() [1/6]

```
T? BlackHole.Core.BHConnection.QueryFirst< T > ( string commandText )
```

A Query that returns only the first Line of the result.

## **Template Parameters**

T Output Type	

### **Parameters**

commandText	Command Text
Command lext	Command Text

#### Returns

The First Line of the Result

### Example:

```
IBHConnection connection = new BHConnection();
Customer? customer = connection.QueryFirst<Customer>(string command);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

## 8.2.3.38 QueryFirst< T >() [2/6]

```
T? BlackHole.Core.BHConnection.QueryFirst< T > ( string commandText, BHParameters parameters)
```

A Query that takes BHParameters and returns only the first Line of the result.

### **Template Parameters**

```
T Output Type
```

#### **Parameters**

command Text   Command Text		Command Text
	parameters	BHParameters Class, populated with black hole parameters

#### Returns

The First Line of the Result

## Example:

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();

parameters.Add("FirstName", "Nick");

Customer? customer = connection.QueryFirst<Customer>(string command, parameters);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

## 8.2.3.39 QueryFirst< T >() [3/6]

A Query that takes an Object as parameters and returns only the first Line of the result.

### **Template Parameters**

```
T Output Type
```

#### **Parameters**

commandText	Command text
parametersObject	Class with properties as Parameters

### Returns

The First Line of the Result

### Example:

```
IBHConnection connection = new BHConnection();

var someClass = new SomeClass{
    FirstName = "Nick"
};

Customer? customer = connection.QueryFirst<Customer>(string command, someClass);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

## 8.2.3.40 QueryFirst< T >() [4/6]

```
T? BlackHole.Core.BHConnection.QueryFirst< T > ( string commandText, BHTransaction bHTransaction )
```

Transaction. A Query that returns only the first Line of the result.

## **Template Parameters**

T Output Type

commandText	Command Text	
bHTransaction	BHTransaction Class, contains connection and transaction	

#### Returns

The First Line of the Result

### Example:

```
IBHConnection connection = new BHConnection();
using(BHTransaction transaction = new BHTransaction())
{
    Customer? customer = connection.QueryFirst<Customer>(string command, transaction);
}
```

 $\textbf{Tip:} \ \ \text{For Oracle and Postgres} \ \ , \ \ \text{Double Quotes are required for the Table and Column Names in your command text}$ 

Implements BlackHole.Core.IBHConnection.

## 8.2.3.41 QueryFirst< T >() [5/6]

Transaction. A Query that takes BHParameters and returns only the first Line of the result.

## **Template Parameters**

T	Output Type

#### **Parameters**

commandText	Command Text
parameters	BHParameters Class, populated with black hole parameters
bHTransaction	BHTransaction Class, contains connection and transaction

### Returns

The First Line of the Result

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();

parameters.Add("FirstName", "Nick");

using(BHTransaction transaction = new BHTransaction())
{
   Customer? customer = connection.QueryFirst<Customer>(string command, parameters, transaction)}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

## 8.2.3.42 QueryFirst< T >() [6/6]

Transaction. A Query that takes an Object as parameters and returns only the first Line of the result.

## **Template Parameters**

```
T Output Type
```

#### **Parameters**

commandText	Command Text
parametersObject	Class with properties as Parameters
bHTransaction	BHTransaction Class, contains connection and transaction

### Returns

The First Line of the Result

```
IBHConnection connection = new BHConnection();

var someClass = new SomeClass{
    Price = 5.5
}

using(BHTransaction transaction = new BHTransaction())
{
    Customer? customer = connection.QueryFirst<Customer>(string command, someClass, transact)}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

### 8.2.3.43 QueryFirstAsync< T >() [1/6]

```
async Task< T?> BlackHole.Core.BHConnection.QueryFirstAsync< T > ( string \ commandText \ )
```

Asyncronous. A Query that returns only the first Line of the result.

#### **Template Parameters**

```
T Output Type
```

### **Parameters**

```
commandText | Command Text
```

#### Returns

The First Line of the Result

### Example:

```
IBHConnection connection = new BHConnection();
Customer? customer = await connection.QueryFirstAsync<Customer>(string command);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

## 8.2.3.44 QueryFirstAsync< T >() [2/6]

```
async Task< T?> BlackHole.Core.BHConnection.QueryFirstAsync< T > ( string \ commandText, \\ BHParameters \ parameters )
```

Asyncronous. A Query that takes BHParameters and returns only the first Line of the result.

#### **Template Parameters**

```
T Output Type
```

commandText	Command Text
parameters	BHParameters Class, populated with black hole parameters

### Returns

The First Line of the Result

### Example:

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();

parameters.Add("FirstName", "Nick");

Customer? customer = await connection.QueryFirstAsync<Customer>(string command, parameters);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

## 8.2.3.45 QueryFirstAsync< T >() [3/6]

```
async Task< T?> BlackHole.Core.BHConnection.QueryFirstAsync< T > ( string \ commandText, \\ object \ parametersObject \ )
```

Asyncronous. A Query that takes an Object as parameters and returns only the first Line of the result.

### **Template Parameters**

Τ	Output Type

#### **Parameters**

commandText	Command text
parametersObject	Class with properties as Parameters

#### Returns

The First Line of the Result

```
IBHConnection connection = new BHConnection();
```

```
var someClass = new SomeClass{
    FirstName = "Nick"
};

Customer? customer = await connection.QueryFirstAsync<Customer>(string command, someClass);
```

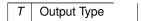
**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

## 8.2.3.46 QueryFirstAsync< T >() [4/6]

```
async Task< T?> BlackHole.Core.BHConnection.QueryFirstAsync< T > ( string \ commandText, \\ BHTransaction \ bHTransaction )
```

Asyncronous. Transaction. A Query that returns only the first Line of the result.

#### **Template Parameters**



### **Parameters**

commandText	Command Text	
bHTransaction	BHTransaction Class, contains connection and transaction	]

#### Returns

The First Line of the Result

#### Example:

```
IBHConnection connection = new BHConnection();
using(BHTransaction transaction = new BHTransaction())
{
   Customer? customer = await connection.QueryFirstAsync<Customer>(string command, transact)}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

### 8.2.3.47 QueryFirstAsync< T >() [5/6]

Asyncronous. Transaction. A Query that takes BHParameters and returns only the first Line of the result.

#### **Template Parameters**

|--|

#### **Parameters**

commandText	Command Text
parameters	BHParameters Class, populated with black hole parameters
bHTransaction	BHTransaction Class, contains connection and transaction

### Returns

The First Line of the Result

### Example:

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();

parameters.Add("FirstName", "Nick");

using(BHTransaction transaction = new BHTransaction())
{
    Customer? customer = await connection.QueryFirstAsync<Customer>(string command, parameter)}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

### 8.2.3.48 QueryFirstAsync< T >() [6/6]

Asyncronous. Transaction. A Query that takes an Object as parameters and returns only the first Line of the result.

utput Type	Т
------------	---

#### **Parameters**

commandText	Command Text	
parametersObject	Class with properties as Parameters	
bHTransaction	BHTransaction Class, contains connection and transaction	

#### Returns

The First Line of the Result

### Example:

```
IBHConnection connection = new BHConnection();

var someClass = new SomeClass{
    Price = 5.5
}

using(BHTransaction transaction = new BHTransaction())
{
    Customer? customer = await connection.QueryFirstAsync<Customer>(string command, someClass)
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implements BlackHole.Core.IBHConnection.

# 8.3 BlackHole.Core.BHDataProvider< T, G > Class Template Reference

Inheritance diagram for BlackHole.Core.BHDataProvider< T, G >:



### **Public Member Functions**

- BHDataProvider ()
- List< T > GetAllEntries ()
- List< T > GetAllEntries (BHTransaction transaction)
- List< Dto > GetAllEntries< Dto > ()

- List< Dto > GetAllEntries< Dto > (BHTransaction transaction)
- List< T > GetAllInactiveEntries ()
- List< T > GetAllInactiveEntries (BHTransaction transaction)
- T? GetEntryByld (G ld)
- T? GetEntryById (G Id, BHTransaction transaction)
- Dto? GetEntryById< Dto > (G Id)
- Dto? GetEntryById< Dto > (G Id, BHTransaction transaction)
- T? GetEntryWhere (Expression < Func < T, bool > > predicate)
- T? GetEntryWhere (Expression< Func< T, bool > > predicate, BHTransaction transaction)
- Dto? GetEntryWhere Dto > (Expression < Func < T, bool > > predicate)
- Dto? GetEntryWhere
   Dto > (Expression
   Func
   T, bool > > predicate, BHTransaction transaction)
- List< T > GetEntriesWhere (Expression< Func< T, bool > > predicate)
- List< T > GetEntriesWhere (Expression< Func< T, bool > > predicate, BHTransaction transaction)
- List< Dto > GetEntriesWhere< Dto > (Expression< Func< T, bool > > predicate)
- List< Dto > GetEntriesWhere< Dto > (Expression< Func< T, bool > > predicate, BHTransaction transaction)
- G? InsertEntry (T entry)
- G? InsertEntry (T entry, BHTransaction transaction)
- List< G?> InsertEntries (List< T > entries)
- List< G?> InsertEntries (List< T > entries, BHTransaction transaction)
- bool UpdateEntryById (T entry)
- bool UpdateEntryByld (T entry, BHTransaction transaction)
- bool UpdateEntryById< Columns > (T entry)
- bool UpdateEntryById< Columns > (T entry, BHTransaction transaction)
- bool UpdateEntriesByld (List< T > entries)
- bool UpdateEntriesByld (List< T > entries, BHTransaction transaction)
- bool UpdateEntriesById< Columns > (List< T > entries)
- bool UpdateEntriesById Columns > (List < T > entries, BHTransaction transaction)
- bool UpdateEntriesWhere (Expression< Func< T, bool > > predicate, T entry)
- $\bullet \ \ bool \ \ Update Entries Where \ (Expression < Func < T, bool >> predicate, T \ entry, BHTransaction \ transaction)\\$
- $\bullet \ \ bool \ \ UpdateEntriesWhere < Columns > (Expression < Func < T, bool >> predicate, Columns \ entry)\\$
- bool UpdateEntriesWhere Columns > (Expression < Func < T, bool > > predicate, Columns entry, BHTransaction transaction)
- bool DeleteAllEntries ()
- bool DeleteAllEntries (BHTransaction transaction)
- bool DeleteEntryById (G Id)
- bool DeleteEntryByld (G Id, BHTransaction transaction)
- bool DeleteInactiveEntryById (G Id)
- bool DeleteInactiveEntryById (G Id, BHTransaction transaction)
- bool ReactivateEntryById (G Id)
- bool ReactivateEntryById (G Id, BHTransaction transaction)
- bool DeleteEntriesWhere (Expression< Func< T, bool > > predicate)
- bool DeleteEntriesWhere (Expression < Func < T, bool > > predicate, BHTransaction transaction)
- Task< List< T >> GetAllEntriesAsync ()
- Task< List< T >> GetAllEntriesAsync (BHTransaction transaction)
- Task< List< Dto > > GetAllEntriesAsync< Dto > ()
- Task< List< Dto > > GetAllEntriesAsync< Dto > (BHTransaction transaction)
- Task< List< T > > GetAllInactiveEntriesAsync ()
- Task< List< T >> GetAllInactiveEntriesAsync (BHTransaction transaction)
- Task< T?> GetEntryByldAsync (G ld)
- Task< T?> GetEntryByldAsync (G ld, BHTransaction transaction)
- Task< Dto?> GetEntryByIdAsync< Dto > (G Id)
- Task< Dto?> GetEntryByIdAsync< Dto > (G Id, BHTransaction transaction)
- Task< T?> GetEntryAsyncWhere (Expression< Func< T, bool >> predicate)
- Task< T?> GetEntryAsyncWhere (Expression< Func< T, bool > > predicate, BHTransaction transaction)

- Task< Dto?> GetEntryAsyncWhere< Dto > (Expression< Func< T, bool > > predicate)
- Task< Dto?> GetEntryAsyncWhere< Dto > (Expression< Func< T, bool > > predicate, BHTransaction transaction)
- Task< List< T >> GetEntriesAsyncWhere (Expression< Func< T, bool >> predicate)
- Task< List< T >> GetEntriesAsyncWhere (Expression< Func< T, bool >> predicate, BHTransaction transaction)
- Task< List< Dto >> GetEntriesAsyncWhere< Dto > (Expression< Func< T, bool >> predicate)
- Task< List< Dto > > GetEntriesAsyncWhere< Dto > (Expression< Func< T, bool > > predicate, BHTransaction transaction)
- Task< G?> InsertEntryAsync (T entry)
- Task< G?> InsertEntryAsync (T entry, BHTransaction transaction)
- Task< List< G?> > InsertEntriesAsync (List< T > entries)
- Task< List< G?> > InsertEntriesAsync (List< T > entries, BHTransaction transaction)
- Task< bool > UpdateEntryByldAsync (T entry)
- Task< bool > UpdateEntryByIdAsync (T entry, BHTransaction transaction)
- Task< bool > UpdateEntryByIdAsync< Columns > (T entry)
- Task< bool > UpdateEntryByIdAsync< Columns > (T entry, BHTransaction transaction)
- Task< bool > UpdateEntriesByldAsync (List< T > entries)
- Task< bool > UpdateEntriesByIdAsync (List< T > entries, BHTransaction transaction)
- Task< bool > UpdateEntriesByldAsync< Columns > (List< T > entries)
- Task< bool > UpdateEntriesByIdAsync< Columns > (List< T > entries, BHTransaction transaction)
- Task< bool > UpdateEntriesAsyncWhere (Expression< Func< T, bool > > predicate, T entry)
- Task< bool > UpdateEntriesAsyncWhere (Expression< Func< T, bool > > predicate, T entry, BHTransaction transaction)
- Task< bool > UpdateEntriesAsyncWhere< Columns > (Expression< Func< T, bool > > predicate, Columns entry)
- Task< bool > UpdateEntriesAsyncWhere< Columns > (Expression< Func< T, bool > > predicate, Columns entry, BHTransaction transaction)
- Task< bool > DeleteAllEntriesAsync ()
- Task< bool > DeleteAllEntriesAsync (BHTransaction transaction)
- Task< bool > DeleteEntryByldAsync (G ld)
- Task< bool > DeleteEntryByldAsync (G ld, BHTransaction transaction)
- Task< bool > DeleteInactiveEntryByIdAsync (G Id)
- Task< bool > DeleteInactiveEntryByIdAsync (G Id, BHTransaction transaction)
- Task< bool > ReactivateEntryByldAsync (G ld)
- Task< bool > ReactivateEntryByIdAsync (G Id, BHTransaction transaction)
- Task< bool > DeleteEntriesAsyncWhere (Expression< Func< T, bool > > predicate)
- Task< bool > DeleteEntriesAsyncWhere (Expression< Func< T, bool > > predicate, BHTransaction transaction)
- G? GetIdWhere (Expression < Func < T, bool > > predicate)
- G? GetIdWhere (Expression < Func < T, bool > > predicate, BHTransaction transaction)
- List< G > GetIdsWhere (Expression< Func< T, bool > > predicate)
- List< G > GetIdsWhere (Expression< Func< T, bool > > predicate, BHTransaction transaction)
- Task< G?> GetIdAsyncWhere (Expression< Func< T, bool > > predicate)
- Task< G?> GetIdAsyncWhere (Expression< Func< T, bool >> predicate, BHTransaction transaction)
- Task< List< G >> GetIdsAsyncWhere (Expression< Func< T, bool >> predicate)
- Task< List< G >> GetIdsAsyncWhere (Expression< T, bool >> predicate, BHTransaction transaction)
- JoinsData< Dto, T, TOther > InnerJoin< TOther, Tkey, Dto > (Expression< Func< T, Tkey > > key, Expression< Func< TOther, Tkey > > otherKey)
- JoinsData< Dto, T, TOther > OuterJoin< TOther, Tkey, Dto > (Expression< Func< T, Tkey > > key, Expression< Func< TOther, Tkey > > otherKey)
- JoinsData< Dto, T, TOther > LeftJoin< TOther, Tkey, Dto > (Expression< Func< T, Tkey > > key, Expression< Func< TOther, Tkey > > otherKey)
- JoinsData< Dto, T, TOther > RightJoin< TOther, Tkey, Dto > (Expression< Func< T, Tkey > > key, Expression< Func< TOther, Tkey > > otherKey)

## 8.3.1 Detailed Description

Provides all the functionality to communicate with the database with many Build-in methods and commands.

### **Template Parameters**

T	Type of Entity
G	Type of Entity's Id

#### Example:

```
IBHDataProvider<Customers,int> _customerService = new BHDataProvider<Customers,int>();
```

Tip:For custom commands, use BlackHole.Core.IBHConnection Interface

**Type Constraints** 

T: BlackHoleEntity<G>

## 8.3.2 Constructor & Destructor Documentation

## 8.3.2.1 BHDataProvider()

```
BlackHole.Core.BHDataProvider < T, G > .BHDataProvider ( )
```

Create a Data Provider that Automatically Communicates with the Database Using the Black Hole Entity you pass in.

```
IBHDataProvider<Customers,int> _customerService = new BHDataProvider<Customers,int>();
```

### 8.4 BlackHole.Core.BHExtensions Class Reference

#### **Static Public Member Functions**

- static JoinsData< Dto, Tsource, TOther > RightJoinOn< Tsource, TOther, Tkey, Dto > (this JoinsData< Dto > data, Expression< Func< Tsource, Tkey > > key, Expression< Func< TOther, Tkey > > otherkey)
- static JoinsData< Dto, Tsource, TOther > LeftJoinOn< Tsource, TOther, Tkey, Dto > (this JoinsData< Dto > data, Expression< Func< Tsource, Tkey > > key, Expression< Func< TOther, Tkey > > otherkey)
- static JoinsData< Dto, Tsource, TOther > OuterJoinOn< Tsource, TOther, Tkey, Dto > (this JoinsData< Dto > data, Expression< Func< Tsource, Tkey > > key, Expression< Func< TOther, Tkey > > otherkey)
- static JoinsData< Dto, Tsource, TOther > InnerJoinOn< Tsource, TOther, Tkey, Dto > (this JoinsData< Dto > data, Expression< Func< Tsource, Tkey > > key, Expression< Func< TOther, Tkey > > otherkey)
- static JoinsData
   Dto, Tsource, TOther > And
   Dto, Tsource, TOther, Tkey > (this JoinsData
   Dto, Tsource, TOther > data, Expression
   Func
   Tsource, Tkey > > key, Expression
   Func
   Tother, Tkey > otherkey)
- static JoinsData < Dto, Tsource, TOther > Or < Dto, Tsource, TOther, Tkey > (this JoinsData < Dto, Tsource, TOther > data, Expression < Func < Tsource, Tkey > > key, Expression < Func < Tother, Tkey > > otherkey)
- static JoinsData< Dto, Tsource, TOther > CastColumnOfFirstAs< Dto, Tsource, TOther, Tkey, TotherKey >
   (this JoinsData< Dto, Tsource, TOther > data, Expression< Func< Tsource, Tkey > > predicate,
   Expression< Func< Dto, TotherKey > > castOnDto)
- static JoinsData< Dto, Tsource, TOther > WhereFirst< Dto, Tsource, TOther > (this JoinsData< Dto, Tsource, TOther > data, Expression< Func< Tsource, bool > > predicate)
- static JoinsData
   Dto, Tsource, TOther > WhereSecond
   Dto, Tsource, TOther > (this JoinsData
   Dto, Tsource, TOther > data, Expression
   Func
   TOther, bool > > predicate)
- static JoinsData < Dto > Then < Dto, Tsource, TOther > (this JoinsData < Dto, Tsource, TOther > data)
- static int StoreAsView<  $\mbox{Dto} > \mbox{(this JoinsData} < \mbox{Dto} > \mbox{data)}$
- static List< Dto > ExecuteQuery< Dto > (this JoinsData< Dto > data)
- static List< Dto > ExecuteQuery< Dto > (this JoinsData< Dto > data, BHTransaction bHTransaction)
- static async Task< List< Dto > ExecuteQueryAsync< Dto > (this JoinsData< Dto > data)
- static async Task< List< Dto > > ExecuteQueryAsync< Dto > (this JoinsData< Dto > data, BHTransaction bHTransaction)

### 8.4.1 Detailed Description

### 8.4.2 Member Function Documentation

#### 8.4.2.1 RightJoinOn< Tsource, TOther, Tkey, Dto >()

Performs a Right Join between the First and the Second specified Entities.

Tsource	First Entity
TOther	Second Entity
Tkey	Type of their Joint Column
Dto	Class of the Output

#### **Parameters**

data	Previous Joins Data
key	First Table Joint Column
otherkey	Second Table Joint Column

#### Returns

The Calculated Data of this Join

### Example:

```
List<OrdersDtoG> ordersJoin = _orderLinesServiceg.InnerJoin<GOrders, Guid, OrdersDtoG>(x =: .Then().RightJoinOn<GOrders, GCustomers, Guid, OrdersDtoG>(x => x.Customer, z => z.Id)
.WhereSecond(x => x.UserType != Guid.Empty).Then()
.InnerJoinOn<GOrderLines, GBonusPoints, Guid, OrdersDtoG>(x => x.BonusPoints, x => x.Id)
.CastColumnOfSecondAs(x => x.Points, x => x.BonusP).CastColumnOfSecondAs(x => x.Id, x =: .Then()
.ExecuteQueryAsync();
```

Tip: Make sure that the DTO Class has all the required properties as the columns that you expect from this Join

### **Type Constraints**

TOther: BlackHoleEntity
Tsource: BlackHoleEntity
Tkey: IComparable

### 8.4.2.2 LeftJoinOn< Tsource, TOther, Tkey, Dto >()

Performs a Left Join between the First and the Second specified Entities.

Tsource	First Entity
TOther	Second Entity
Tkey	Type of their Joint Column
Dto	Class of the Output

#### **Parameters**

data	Previous Joins Data
key	First Table Joint Column
otherkey	Second Table Joint Column

#### Returns

The Calculated Data of this Join

#### Example:

```
List<OrdersDtoG> ordersJoin = _orderLinesServiceg.InnerJoin<GOrders, Guid, OrdersDtoG>(x =: .Then().LeftJoinOn<GOrders, GCustomers, Guid, OrdersDtoG>(x => x.Customer, z => z.Id)
.WhereSecond(x => x.UserType != Guid.Empty).Then()
.InnerJoinOn<GOrderLines, GBonusPoints, Guid, OrdersDtoG>(x => x.BonusPoints, x => x.Id)
.CastColumnOfSecondAs(x => x.Points, x => x.BonusP).CastColumnOfSecondAs(x => x.Id, x =: .Then()
.ExecuteQueryAsync();
```

Tip: Make sure that the DTO Class has all the required properties as the columns that you expect from this Join

### **Type Constraints**

TOther: BlackHoleEntity
Tsource: BlackHoleEntity
Tkey: IComparable

### 8.4.2.3 OuterJoinOn< Tsource, TOther, Tkey, Dto >()

Performs an Outer Join between the First and the Second specified Entities.

Tsource	First Entity
TOther	Second Entity
Tkey	Type of their Joint Column
Dto	Class of the Output

#### **Parameters**

data	Previous Joins Data
key	First Table Joint Column
otherkey	Second Table Joint Column

#### Returns

The Calculated Data of this Join

### Example:

```
List<OrdersDtoG> ordersJoin = _orderLinesServiceg.InnerJoin<GOrders, Guid, OrdersDtoG>(x =: .Then().OuterJoinOn<GOrders, GCustomers, Guid, OrdersDtoG>(x => x.Customer, z => z.Id)
.WhereSecond(x => x.UserType != Guid.Empty).Then()
.InnerJoinOn<GOrderLines, GBonusPoints, Guid, OrdersDtoG>(x => x.BonusPoints, x => x.Id)
.CastColumnOfSecondAs(x => x.Points, x => x.BonusP).CastColumnOfSecondAs(x => x.Id, x =: .Then()
.ExecuteQueryAsync();
```

Tip: Make sure that the DTO Class has all the required properties as the columns that you expect from this Join

### **Type Constraints**

TOther: BlackHoleEntity
Tsource: BlackHoleEntity
Tkey: IComparable

### 8.4.2.4 InnerJoinOn< Tsource, TOther, Tkey, Dto >()

Performs an Inner Join between the First and the Second specified Entities.

Tsource	First Entity
TOther	Second Entity
Tkey	Type of their Joint Column
Dto	Class of the Output

#### **Parameters**

data	Previous Joins Data
key	First Table Joint Column
otherkey	Second Table Joint Column

#### Returns

The Calculated Data of this Join

### Example:

```
List<OrdersDtoG> ordersJoin = _orderLinesServiceg.InnerJoin<GOrders, Guid, OrdersDtoG>(x =:
.Then().InnerJoinOn<GOrders, GCustomers, Guid, OrdersDtoG>(x => x.Customer, z => z.Id)
.WhereSecond(x => x.UserType != Guid.Empty).Then()
.InnerJoinOn<GOrderLines, GBonusPoints, Guid, OrdersDtoG>(x => x.BonusPoints, x => x.Id;
.CastColumnOfSecondAs(x => x.Points, x => x.BonusP).CastColumnOfSecondAs(x => x.Id, x =:
.Then()
.ExecuteQueryAsync();
```

Tip: Make sure thet the DTO Class has all the required properties as the columns that you expect from this Join

### **Type Constraints**

TOther: BlackHoleEntity
Tsource: BlackHoleEntity

Tkey: IComparable
Dto: BlackHoleDto

### 8.4.2.5 And< Dto, Tsource, TOther, Tkey >()

Uses Additional columns to the Join.

Dto	Class of the Output
Tsource	First Entity
TOther	Second Entity
Tkey	Type of their Joint Column

#### **Parameters**

data	Previous Joins Data
key	First Table Joint Column
otherkey	Second Table Joint Column

#### Returns

The Calculated Data of this Join

#### Example:

```
List<OrdersDtoG> ordersJoin = await _orderLinesServiceg.InnerJoin<GOrders, Guid, OrdersDtoG
.Then().RightJoinOn<GOrders, GCustomers, Guid, OrdersDtoG>(x => x.Customer, z => z.Id).
.WhereSecond(x => x.UserType != Guid.Empty).Then()
.InnerJoinOn<GOrderLines, GBonusPoints, Guid, OrdersDtoG>(x => x.BonusPoints, x => x.Id)
.CastColumnOfSecondAs(x => x.Points, x => x.BonusP).CastColumnOfSecondAs(x => x.Id, x =:
.Then()
.ExecuteQueryAsync();
```

Tip: Make sure that the DTO Class has all the required properties as the columns that you expect from this Join

### **Type Constraints**

Tkey: IComparable

### 8.4.2.6 Or < Dto, Tsource, TOther, Tkey >()

Uses Additional case to the Join.

## **Template Parameters**

Dto	Class of the Output
Tsource	First Entity
TOther	Second Entity
Tkey	Type of their Joint Column

data	Previous Joins Data
key	First Table Joint Column
otherkey	Second Table Joint Column

#### Returns

The Calculated Data of this Join

#### Example:

```
List<OrdersDtoG> ordersJoin = await _orderLinesServiceg.InnerJoin<GOrders, Guid, OrdersDto .Then().RightJoinOn<GOrders, GCustomers, Guid, OrdersDtoG>(x => x.Customer, z => z.Id).( .WhereSecond(x => x.UserType != Guid.Empty).Then() .InnerJoinOn<GOrderLines, GBonusPoints, Guid, OrdersDtoG>(x => x.BonusPoints, x => x.Id) .CastColumnOfSecondAs(x => x.Points, x => x.BonusP).CastColumnOfSecondAs(x => x.Id, x => .Then() .ExecuteQueryAsync();
```

Tip: Make sure that the DTO Class has all the required properties as the columns that you expect from this Join

### **Type Constraints**

Tkey: IComparable

## 8.4.2.7 CastColumnOfSecondAs< Dto, Tsource, TOther, Tkey, TotherKey >()

Casts a column of the second Entity as a column of the output's DTO.

!!Important!! => There are some restrictions to the types of the properties that can be casted. Read the Documentation for details. If a data type is not allowed to be converted to another type, then the cast will be ignored in the Execution and the DTO column will be null.

Tip: Cast has priority over normal mapping, For example the Column Id of the DTO is by default mapped to the First Entity of all Joins. If you want to map a different Entity's Id into that column, use a Cast.

### **Template Parameters**

Dto	Class of the Output
Tsource	First Entity
TOther	Second Entity
Tkey	Type of their Joint Column

data	Previous Joins Data
predicate	First Table Joint Column
castOnDto	Second Table Joint Column

#### Returns

The Calculated Data of this Join

#### Example:

```
List<OrdersDtoG> ordersJoin = await _orderLinesServiceg.InnerJoin<GOrders, Guid, OrdersDtoG. Then().RightJoinOn<GOrders, GCustomers, Guid, OrdersDtoG>(x => x.Customer, z => z.Id)
.WhereSecond(x => x.UserType != Guid.Empty).Then()
.InnerJoinOn<GOrderLines, GBonusPoints, Guid, OrdersDtoG>(x => x.BonusPoints, x => x.Id
.CastColumnOfSecondAs(x => x.Points, x => x.BonusP).CastColumnOfSecondAs(x => x.Id, x =
.Then()
.ExecuteQueryAsync();
```

**Tip:** Cast has priority over normal mapping, For example the Column Id of the DTO is by default mapped to the First Entity of all Joins. If you want to map a different Entity's Id into that column, use a Cast.

### 8.4.2.8 CastColumnOfFirstAs< Dto, Tsource, TOther, Tkey, TotherKey >()

Casts a column of the first Entity as a column of the output's DTO.

!!Important!! => There are some restrictions to the types of the properties that can be casted. Read the Documentation for details. If a data type is not allowed to be converted to another type, then the cast will be ignored in the Execution and the DTO column will be null.

Tip: Cast has priority over normal mapping, For example the Column Id of the DTO is by default mapped to the First Entity of all Joins. If you want to map a different Entity's Id into that column, use a Cast.

#### **Template Parameters**

Dto	Class of the Output
Tsource	First Entity
TOther	Second Entity
Tkey	Type of their Joint Column

#### **Parameters**

data	Previous Joins Data
------	---------------------

predicate	First Table Joint Column
castOnDto	Second Table Joint Column

#### Returns

The Calculated Data of this Join

### Example:

```
List<OrdersDtoG> ordersJoin = await _orderLinesServiceg.InnerJoin<GOrders, Guid, OrdersDtoG. Then().RightJoinOn<GOrders, GCustomers, Guid, OrdersDtoG>(x => x.Customer, z => z.Id)
.WhereSecond(x => x.UserType != Guid.Empty).Then()
.InnerJoinOn<GOrderLines, GBonusPoints, Guid, OrdersDtoG>(x => x.BonusPoints, x => x.Id
.CastColumnOfSecondAs(x => x.Points, x => x.BonusP).CastColumnOfFirstAs(x => x.Id, x => .Then()
.ExecuteQueryAsync();
```

**Tip:** Cast has priority over normal mapping, For example the Column Id of the DTO is by default mapped to the First Entity of all Joins. If you want to map a different Entity's Id into that column, use a Cast.

### 8.4.2.9 WhereFirst< Dto, Tsource, TOther >()

Uses a Lambda Expression as filters for the First table of the current Join.Only the entries of the table that match these filters will be used.

### **Template Parameters**

Dto	Class of the Output
Tsource	First Entity
TOther	Second Entity
Tkey	Type of their Joint Column

#### **Parameters**

data	Previous Joins Data
predicate	First Table Joint Column

#### Returns

The Calculated Data of this Join

```
List<OrdersDtoG> ordersJoin = await _orderLinesServiceg.InnerJoin<GOrders, Guid, OrdersDtoG
.Then().RightJoinOn<GOrders, GCustomers, Guid, OrdersDtoG>(x => x.Customer, z => z.Id).?
.WhereFirst(x => x.CustomerId != Guid.Empty).Then()
.InnerJoinOn<GOrderLines, GBonusPoints, Guid, OrdersDtoG>(x => x.BonusPoints, x => x.Id)
.CastColumnOfSecondAs(x => x.Points, x => x.BonusP).CastColumnOfSecondAs(x => x.Id, x => .Then()
.ExecuteQueryAsync();
```

8.4.2.10 WhereSecond< Dto, Tsource, TOther >()

Uses a Lambda Expression as filters for the Second table of the current Join. Only the entries of the table that match these filters will be used.

#### **Template Parameters**

Dto	Class of the Output
Tsource	First Entity
TOther	Second Entity
Tkey	Type of their Joint Column

#### **Parameters**

data	Previous Joins Data
predicate	First Table Joint Column

### Returns

The Calculated Data of this Join

### Example:

```
List<OrdersDtoG> ordersJoin = await _orderLinesServiceg.InnerJoin<GOrders, Guid, OrdersDtoG
.Then().RightJoinOn<GOrders, GCustomers, Guid, OrdersDtoG>(x => x.Customer, z => z.Id).A
.WhereSecond(x => x.CustomerId != Guid.Empty).Then()
.InnerJoinOn<GOrderLines, GBonusPoints, Guid, OrdersDtoG>(x => x.BonusPoints, x => x.Id)
.CastColumnOfSecondAs(x => x.Points, x => x.BonusP).CastColumnOfSecondAs(x => x.Id, x => .Then()
.ExecuteQueryAsync();
```

Generated by Doxygen

## 8.4.2.11 Then< Dto, Tsource, TOther >()

```
\label{eq:core.BHExtensions.Then} \mbox{Static JoinsData} < \mbox{Dto} > \mbox{BlackHole.Core.BHExtensions.Then} < \mbox{Dto}, \mbox{Tsource}, \mbox{Tother} > \mbox{data} > \mbox{(static)}
```

Combines the Data of the Join and let's you start another Join or Execute the Joins Data or Store them as View.

Dto	
Tsource	
TOther	

#### **Parameters**

data	
------	--

Returns

### Example:

```
List<OrdersDtoG> ordersJoin = await _orderLinesServiceg.InnerJoin<GOrders, Guid, OrdersDtoG
.Then().RightJoinOn<GOrders, GCustomers, Guid, OrdersDtoG>(x => x.Customer, z => z.Id).A
.WhereSecond(x => x.CustomerId != Guid.Empty).Then()
.InnerJoinOn<GOrderLines, GBonusPoints, Guid, OrdersDtoG>(x => x.BonusPoints, x => x.Id)
.CastColumnOfSecondAs(x => x.Points, x => x.BonusP).CastColumnOfSecondAs(x => x.Id, x => .Then()
.ExecuteQueryAsync();
```

8.4.2.12 StoreAsView < Dto >()

```
static int BlackHole.Core.BHExtensions.StoreAsView<br/>< Dto > ( this JoinsData<br/>< Dto > data ) [static]
```

Stores the Joins Data with the DTO Class as Identifier and then you can execute them as many times as you want using the 'IBlackHoleViewStorage' Interface.

Benefit: With this method, the program doesn't have to calculate the Joins Data multiple times and it executes the Joins faster.

Tip: This method is recommended if the parameters in the current Joins Data are not depending on the user's inputs. Run your Joins Once in the StartUp of your program and store them as Views.

### **Template Parameters**

D.	D . T . ( O): . O
Dto	Data Transfer Object. Output

### **Parameters**

data	Joins Data
------	------------

#### Returns

The index of this Joins Data in the Stored Views List

#### Example:

```
int ViewPositionInViewStorage = await _orderLinesServiceg.InnerJoin<GOrders, Guid, OrdersDto
    .Then().RightJoinOn<GOrders, GCustomers, Guid, OrdersDtoG>(x => x.Customer, z => z.Id).
    .WhereSecond(x => x.CustomerId != Guid.Empty).Then()
    .InnerJoinOn<GOrderLines, GBonusPoints, Guid, OrdersDtoG>(x => x.BonusPoints, x => x.Id)
    .CastColumnOfSecondAs(x => x.Points, x => x.BonusP).CastColumnOfSecondAs(x => x.Id, x => ...
    .Then()
    .StoreAsView();
IBHViewStorage viewStorage = new BHViewStorage();
```

. **Tip:** If the parameters of the view are always the same in your application, you can make a method to store all the views on the startup.

List<OrdersDtoG> ordersJoin = viewStorage.ExecuteView<OrdersDtoG>();

This will increase performace, as the Joins calculations will only be performed once.

**Type Constraints** 

Dto: BlackHoleDto

#### 8.4.2.13 ExecuteQuery< Dto >() [1/2]

```
static List< Dto > BlackHole.Core.BHExtensions.ExecuteQuery< Dto > ( this JoinsData< Dto > data ) [static]
```

Executes the Joins Data and returns the result.

**Template Parameters** 

```
Dto Data Transfer Object. Output
```

#### **Parameters**

```
data Joins Data
```

#### Returns

The Entries of the Joins mapped into DTO

```
List<OrdersDtoG> ordersJoin = _orderLinesServiceg.InnerJoin<GOrders, Guid, OrdersDtoG>(x = x)
```

```
.Then().RightJoinOn<GOrders, GCustomers, Guid, OrdersDtoG>(x => x.Customer, z => z.Id). A .WhereSecond(x => x.UserType != Guid.Empty).Then() .InnerJoinOn<GOrderLines, GBonusPoints, Guid, OrdersDtoG>(x => x.BonusPoints, x => x.Id) .CastColumnOfSecondAs(x => x.Points, x => x.BonusP).CastColumnOfSecondAs(x => x.Id, x => .Then() .ExecuteQuery();
```

Tip: Make sure thet the DTO Class has all the required properties as the columns that you expect from this Join

**Type Constraints** 

Dto: BlackHoleDto

# 8.4.2.14 ExecuteQuery< Dto >() [2/2]

```
static List< Dto > BlackHole.Core.BHExtensions.ExecuteQuery< Dto > ( this JoinsData< Dto > data, BHTransaction bHTransaction) [static]
```

Transaction. Executes the Joins Data and returns the result.

**Template Parameters** 

```
Dto Data Transfer Object. Output
```

#### **Parameters**

```
data Joins Data
```

Returns

The Entries of the Joins mapped into DTO

#### Example:

```
using(BHTransaction transaction = new BHTransaction())
{
    List<OrdersDtoG> ordersJoin = _orderLinesServiceg.InnerJoin<GOrders, Guid, OrdersDtoG>
    .Then().RightJoinOn<GOrders, GCustomers, Guid, OrdersDtoG>(x => x.Customer, z => z.T.)
    .WhereSecond(x => x.UserType != Guid.Empty).Then()
    .InnerJoinOn<GOrderLines, GBonusPoints, Guid, OrdersDtoG>(x => x.BonusPoints, x => x.CustColumnOfSecondAs(x => x.Points, x => x.BonusP).CastColumnOfSecondAs(x => x.Id, .Then()
    .ExecuteQuery(transaction);
}
```

Tip: Make sure that the DTO Class has all the required properties as the columns that you expect from this Join

**Type Constraints** 

Dto: BlackHoleDto

#### 8.4.2.15 ExecuteQueryAsync< Dto >() [1/2]

```
static async Task< List< Dto >> BlackHole.Core.BHExtensions.ExecuteQueryAsync< Dto > ( this JoinsData< Dto > data ) [static]
```

Asyncronous. Executes the Joins Data and returns the result.

**Template Parameters** 

```
Dto Data Transfer Object. Output
```

# **Parameters**

```
data Joins Data
```

#### Returns

The Entries of the Joins mapped into DTO

# Example:

```
List<OrdersDtoG> ordersJoin = await _orderLinesServiceg.InnerJoin<GOrders, Guid, OrdersDtoG
.Then().RightJoinOn<GOrders, GCustomers, Guid, OrdersDtoG>(x => x.Customer, z => z.Id).
.WhereSecond(x => x.UserType != Guid.Empty).Then()
.InnerJoinOn<GOrderLines, GBonusPoints, Guid, OrdersDtoG>(x => x.BonusPoints, x => x.Id)
.CastColumnOfSecondAs(x => x.Points, x => x.BonusP).CastColumnOfSecondAs(x => x.Id, x => .Then()
.ExecuteQueryAsync();
```

Tip: Make sure thet the DTO Class has all the required properties as the columns that you expect from this Join

**Type Constraints** 

Dto: BlackHoleDto

#### 8.4.2.16 ExecuteQueryAsync< Dto >() [2/2]

```
static async Task< List< Dto >> BlackHole.Core.BHExtensions.ExecuteQueryAsync< Dto > ( this JoinsData< Dto > data, BHTransaction bHTransaction) [static]
```

Transaction. Asyncronous. Executes the Joins Data and returns the result.

**Template Parameters** 

```
Dto Data Transfer Object. Output
```

```
data Joins Data
```

Returns

The Entries of the Joins mapped into DTO

#### Example:

```
using(BHTransaction transaction = new BHTransaction())
{
    List<OrdersDtoG> ordersJoin = await _orderLinesServiceg.InnerJoin<GOrders, Guid, Orders.
    .Then().RightJoinOn<GOrders, GCustomers, Guid, OrdersDtoG>(x => x.Customer, z => z...
    .WhereSecond(x => x.UserType != Guid.Empty).Then()
    .InnerJoinOn<GOrderLines, GBonusPoints, Guid, OrdersDtoG>(x => x.BonusPoints, x => x.CustColumnOfSecondAs(x => x.Points, x => x.BonusP).CastColumnOfSecondAs(x => x.Id, .Then()
    .ExecuteQueryAsync(transaction);
}
```

Tip: Make sure thet the DTO Class has all the required properties as the columns that you expect from this Join

**Type Constraints** 

Dto: BlackHoleDto

# 8.5 BlackHole.Core.BHParameters Class Reference

# **Public Member Functions**

· void Add (string? Name, object? Value)

# 8.5.1 Detailed Description

An Object that can store Dynamic Parameters that get translated into SQL Parameters in every provider.

# 8.5.2 Member Function Documentation

#### 8.5.2.1 Add()

Adds a parameter with name and value.

Name Name of the Parame	
Value Value of the Parame	

# Example:

```
BHParameters parameters = new BHParameters();
parameters.Add("Price" ,5.18);
parameters.Add("Name", "Milk");

IBHConnection connection = new BHConnection();

connection.JustExecute("Update Product set Price=@Price where Name=@Name", parameters);
```

# 8.6 BlackHole.Core.BHTransaction Class Reference

Inherits IDisposable.

#### **Public Member Functions**

- BHTransaction ()
- bool Commit ()
- bool DoNotCommit ()
- bool RollBack ()
- void Dispose ()

# 8.6.1 Detailed Description

A Transaction Object that automatically creates a connection and a transaction and it can be used in every BlackHole Method.

**Tip:** Don't forget to dispose this Object after using it. If you don't perform any action on this class, the Commit Method gets triggered on Dispose

# 8.6.2 Constructor & Destructor Documentation

# 8.6.2.1 BHTransaction()

```
BlackHole.Core.BHTransaction.BHTransaction ( )
```

A Transaction Object that automatically creates a connection and a transaction and it can be used in every BlackHole Method.

Tip: Don't forget to dispose this Object after using it. If you don't perform any action on this class, the Commit Method gets triggered on Dispose

# Example:

```
IBHDataProvider<Customer,int> _customerService = new BHDataProvider<Customer,int>();
using(BHTransaction transaction = new BHTransaction())
{
    List<int> InsertedIds = _customerService.InsertEntries(List<Customer> customers , transaction(InsertedIds.Count == 0)
    {
        transaction.DoNotCommit();
    }

    bool Success = _customerService.DeleteEntriesWhere(x=> x.UserType == 'Premium', transaction(Isuccess)
    {
        transaction.RollBack();
    }
}
```

Tip: It can be used in any method.

# 8.6.3 Member Function Documentation

#### 8.6.3.1 Commit()

```
bool BlackHole.Core.BHTransaction.Commit ( )
```

Commit the transaction.

Returns

Successful Commit Boolean

#### 8.6.3.2 DoNotCommit()

```
bool BlackHole.Core.BHTransaction.DoNotCommit ( )
```

Block the transaction.

Returns

Successful Block Boolean

# 8.6.3.3 RollBack()

```
bool BlackHole.Core.BHTransaction.RollBack ( )
```

RollBack the transaction

Returns

Successful Rollback Boolean

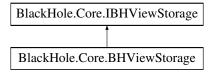
# 8.6.3.4 Dispose()

```
void BlackHole.Core.BHTransaction.Dispose ( )
```

Disposes the Connection and the transaction. If no other action have been used, it also Commits the transaction.

# 8.7 BlackHole.Core.BHViewStorage Class Reference

Inheritance diagram for BlackHole.Core.BHViewStorage:



# **Public Member Functions**

- BHViewStorage ()
- List< Dto > ExecuteView< Dto > ()
- List< Dto > ExecuteView< Dto > (BHTransaction transaction)
- Task< List< Dto > > ExecuteViewAsync< Dto > ()
- Task< List< Dto > > ExecuteViewAsync< Dto > (BHTransaction transaction)

# 8.7.1 Detailed Description

A Class that contains the functionality to execute the Stored Views.

# 8.7.2 Constructor & Destructor Documentation

#### 8.7.2.1 BHViewStorage()

```
BlackHole.Core.BHViewStorage.BHViewStorage ( )
```

Creates an interface to access the stored views.

# 8.8 BlackHole.Configuration.BlackHoleConfiguration Class Reference

#### **Static Public Member Functions**

- static IServiceCollection SuperNova (this IServiceCollection services, Action< BlackHoleSettings > settings)
- static bool TestDatabase ()
- static bool DropDatabase ()

# 8.8.1 Detailed Description

All Black Hole Configuration Contains methods to Initialize ,Drop and Test database

# 8.8.2 Member Function Documentation

# 8.8.2.1 SuperNova()

```
static IServiceCollection BlackHole.Configuration.BlackHoleConfiguration.SuperNova (  this \ IServiceCollection \ services, \\ Action < \ BlackHoleSettings > settings \ ) \ \ [static]
```

Generates a Database , based on the inserted connection string, to an Existing Database Server.

The connection string Must lead to the server and the Database part of the connection string will be used to create the database.

It uses the BlackHole Entities and Services of the Calling Assembly or other Assemblies depending on the settings.

You can choose to use only specific namespaces for the Entities and the Services.

services   IServiceCollection	
settings Black Hole Settings Cla	

#### Returns

IService Collection with BlackHole Services added

#### Example:

services.SuperNova(settings => settings.AddDatabase(connection => connection.UseMySql(connection)

# 8.8.2.2 TestDatabase()

static bool BlackHole.Configuration.BlackHoleConfiguration.TestDatabase ( ) [static]

Checks the database's condition.

#### Returns

Database is Up Boolean

# Example:

BlackHoleConfiguration.TestDatabase()

Tip: This must be used after the initialization method BlackHole.Configuration.BlackHoleConfiguration.SuperNova

# 8.8.2.3 DropDatabase()

static bool BlackHole.Configuration.BlackHoleConfiguration.DropDatabase ( ) [static]

Closes all connections and drops the database. Works only in Developer Mode.

#### Returns

Successful Drop Boolean

#### Example:

BlackHoleConfiguration.DropDatabase()

Tip: This must be used after the initialization method BlackHole.Configuration.BlackHoleConfiguration.SuperNova

# 8.9 BlackHole.Entities.BlackHoleDto < G > Class Template Reference

Inherits BlackHoleDto.

# **Properties**

• G ld [get, set]

# 8.9.1 Detailed Description

**Template Parameters** 

```
G Type of the base Entity Id
```

Data Transfer Object. It is Required to use Mapping on DTO, Views and Joins Functionality. Make a class inherit from this, to be recognized as a DTO

# Example:

```
public class CustomerDTO : BlackHoleDto<int>
```

# 8.9.2 Property Documentation

#### 8.9.2.1 ld

```
G BlackHole.Entities.BlackHoleDto< G >.Id [get], [set]
```

It Contains the column Id to make sure that at least one column will match with the database table.

# ${\bf 8.10 \quad Black Hole. Entities. Black Hole Entity} < {\bf G} > {\bf Class\ Template\ Reference}$

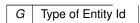
Inherits BlackHoleEntity.

# **Properties**

```
• G ld [get, set]
```

# 8.10.1 Detailed Description

**Template Parameters** 



Black Hole Entity. The table in database is based on this. Make a class inherit from this, to be recognized as an Entity.

```
public class Customer : BlackHoleEntity<int>
```

# 8.10.2 Property Documentation

# 8.10.2.1 ld

```
G BlackHole.Entities.BlackHoleEntity< G >.Id [get], [set]
```

It Contains the column Id that is the Primary Key of the Table

# 8.11 BlackHole.Services.BlackHoleScoped Class Reference

# **Public Member Functions**

• BlackHoleScoped ()

# 8.11.1 Detailed Description

Make a service Inherit from this class to automatically get registered as Scoped.

There is no need to Add your service on the startup.

BlackHole will find it and add it to the IServiceCollection.

This also works if you are using services from a different assembly.

# Example:

```
public class CustomerService : BlackHoleScoped , ICustomerService
```

# 8.11.2 Constructor & Destructor Documentation

# 8.11.2.1 BlackHoleScoped()

```
BlackHole.Services.BlackHoleScoped.BlackHoleScoped ( )
```

Make a service Inherit from this class to automatically get registered as Scoped.

# 8.12 BlackHole.Configuration.BlackHoleSettings Class Reference

# **Public Member Functions**

- DataPathSettings AddDatabase (Action< ConnectionSettings > connectionSettings)
- BlackHoleSettings IsDeveloperMode (bool isDevMode)

# 8.12.1 Detailed Description

The full configuration object. It contains all the settings for this library.

#### Example:

```
services.SuperNova(settings => settings....)
```

# 8.12.2 Member Function Documentation

# 8.12.2.1 AddDatabase()

```
\label{lem:decomposition} \begin{tabular}{ll} DataPathSettings & BlackHole.Configuration.BlackHoleSettings.AddDatabase ( \\ & Action < ConnectionSettings > connectionSettings ) \end{tabular}
```

Add the configuration for a database.

#### **Parameters**

connectionSettings connection settings

# Returns

BlackHole.Configuration.DataPathSettings

# Example:

```
services.SuperNova(settings => settings.AddDatabase(connection => connection.UseMySql(connection)
```

# 8.12.2.2 IsDeveloperMode()

```
\begin{tabular}{lll} {\tt BlackHoleSettings} & {\tt BlackHole.Configuration.BlackHoleSettings.IsDeveloperMode} & ( & {\tt bool} & isDevMode & ) \end{tabular}
```

If this is set to TRUE, then all the changes to the Entities will be applied to the database.

If this is set to FALSE, BlackHole will switch to production mode and it will not allow to drop any tables or columns from the database, to protect the production database from data loss in case of developer's mistake

The default value is FALSE

isDevMode

Returns

BlackHole.Configuration.BlackHoleSettings

# Example:

services.SuperNova(settings => settings.IsDeveloperMode(true).AddDatabase(connection => con

# 8.13 BlackHole.Services.BlackHoleSingleton Class Reference

#### **Public Member Functions**

• BlackHoleSingleton ()

# 8.13.1 Detailed Description

Make a service Inherit from this class to automatically get registered as Singleton.

There is no need to Add your service on the startup.

BlackHole will find it and add it to the IServiceCollection.

This also works if you are using services from a different assembly.

# Example:

 $\verb|public class CustomerService : BlackHoleSingleton , ICustomerService|\\$ 

# 8.13.2 Constructor & Destructor Documentation

#### 8.13.2.1 BlackHoleSingleton()

BlackHole.Services.BlackHoleSingleton.BlackHoleSingleton ( )

Make a service Inherit from this class to automatically get registered as Singleton.

# 8.14 BlackHole.Services.BlackHoleTransient Class Reference

#### **Public Member Functions**

BlackHoleTransient ()

# 8.14.1 Detailed Description

Make a service Inherit from this class to automatically get registered as Transient.

There is no need to Add your service on the startup.

BlackHole will find it and add it to the IServiceCollection.

This also works if you are using services from a different assembly.

#### Example:

public class CustomerService : BlackHoleTransient , ICustomerService

#### 8.14.2 Constructor & Destructor Documentation

#### 8.14.2.1 BlackHoleTransient()

```
BlackHole.Services.BlackHoleTransient.BlackHoleTransient ( )
```

Make a service Inherit from this class to automatically get registered as Transient.

# 8.15 BlackHole.Configuration.ConnectionAdditionalSettings Class Reference

# **Public Member Functions**

- ServicesWithNamespace UseEntitiesInNamespace (string entityNamespace)
- ServicesWithNamespace UseEntitiesInNamespaces (List< string > entityNamespaces)
- ServicesWithNamespace UseEntitiesInNamespaces (Action < List < string > > entityNamespaces)
- EntitiesWithNamespace AddServicesFromNamespace (string servicesNamespace)
- EntitiesWithNamespace AddServicesFromNamespaces (List< string > servicesNamespaces)
- EntitiesWithNamespace AddServicesFromNamespaces (Action < List < string > > servicesNamespaces)
- void UseOtherAssembly (Assembly otherAssembly)
- void UseOtherAssemblies (List< Assembly > otherAssemblies)
- void UseOtherAssemblies (Action < List < Assembly > > otherAssemblies)
- · void UseAdditionalAssembly (Assembly additionalAssembly)
- void UseAdditionalAssemblies (List< Assembly > additionalAssemblies)
- void UseAdditionalAssemblies (Action < List < Assembly > > additionalAssemblies)

#### 8.15.1 Detailed Description

Additional configuration object. for the Namespaces and the Assemblies that will be used by the BlackHole.

# 8.15.2 Member Function Documentation

#### 8.15.2.1 UseEntitiesInNamespace()

Using only the Entities that are in the specified Namespace.

#### **Parameters**

entityNamespace | Namespace Full Name 'MyProject.Entities....'

#### Returns

BlackHole.Configuration.ServicesWithNamespace

# Example:

```
services.SuperNova(settings => settings.AddDatabase(connection => connection.UseMySql(connection.UseEntitiesInNamespace(string NamespaceFullName)))
```

# 8.15.2.2 UseEntitiesInNamespaces() [1/2]

```
\label{lem:servicesWithNamespace} Services \mbox{WithNamespace BlackHole.} Configuration. Connection \mbox{Additional Settings.} \mbox{UseEntities In} \mbox{$\leftarrow$} \mbox{Namespaces} \mbox{\ (} \mbox{$List<$ string $>$ entityNamespaces $)$}
```

Using only the Entities in the specified Namespaces.

#### **Parameters**

```
entityNamespaces | List of Namespaces Full Names
```

#### Returns

BlackHole.Configuration.ServicesWithNamespace

```
services.SuperNova(settings => settings.AddDatabase(connection => connection.UseMySql(connection))
.UseEntitiesInNamespaces(List<string> NamespacesFullNames)))
```

#### 8.15.2.3 UseEntitiesInNamespaces() [2/2]

```
ServicesWithNamespace BlackHole.Configuration.ConnectionAdditionalSettings.UseEntitiesIn \leftrightarrow Namespaces (
Action < List < string > > entityNamespaces )
```

Using only the Entities in the specified Namespaces.

#### **Parameters**

entityNamespaces | List of Namespaces Full Names

#### Returns

BlackHole.Configuration.ServicesWithNamespace

#### Example:

# 8.15.2.4 AddServicesFromNamespace()

```
\label{lem:entitiesWithNamespace} \begin{tabular}{ll} EntitiesWithNamespace & BlackHole.Configuration.ConnectionAdditionalSettings.AddServicesFrom $\longleftrightarrow$ Namespace ( & string $servicesNamespace ) \end{tabular}
```

Using only the services that are in the specified Namespace and inherit from BlackHole Services Classes.

#### **Parameters**

```
servicesNamespace | Namespace Full Name 'MyProject.Services....'
```

#### Returns

BlackHole.Configuration.EntitiesWithNamespace

```
services.SuperNova(settings => settings.AddDatabase(connection => connection.UseMySql(connection.AddServicesFromNamespace(string NamespaceFullName)))
```

#### 8.15.2.5 AddServicesFromNamespaces() [1/2]

Using only the services that are in the specified Namespaces and inherit from BlackHole Services Classes.

#### **Parameters**

servicesNamespaces | List of Namespaces Full Names

#### Returns

BlackHole.Configuration.EntitiesWithNamespace

#### Example:

# 8.15.2.6 AddServicesFromNamespaces() [2/2]

```
\label{lem:entitiesWithNamespace} \begin{tabular}{ll} EntitiesWithNamespace BlackHole.Configuration.ConnectionAdditionalSettings.AddServicesFrom$$\hookrightarrow$ $$Namespaces ($$Action< List< string $>>$$ $$servicesNamespaces ($$
```

Using only the services that are in the specified Namespaces and inherit from BlackHole Services Classes.

#### **Parameters**

```
servicesNamespaces List of Namespaces Full Names
```

#### Returns

BlackHole.Configuration.EntitiesWithNamespace

#### 8.15.2.7 UseOtherAssembly()

```
\label{thm:configuration} \mbox{\sc ConnectionAdditionalSettings.UseOtherAssembly (} \\ \mbox{\sc Assembly otherAssembly )}
```

Using BlackHole Entities and BlackHole Services ONLY from the specified Assembly.

**Parameters** 

```
otherAssembly Full Assembly
```

# Example:

services.SuperNova(settings => settings.AddDatabase(connection => connection.UseMySql(connection.UseOtherAssembly(Assembly assembly)))

# 8.15.2.8 UseOtherAssemblies() [1/2]

```
\label{local_configuration} \mbox{\sc ConnectionAdditionalSettings.UseOtherAssemblies (} \\ \mbox{\sc List< Assembly } \mbox{\sc otherAssemblies )}
```

Using BlackHole Entities and BlackHole Services ONLY from the specified Assemblies.

#### **Parameters**

```
otherAssemblies List of Assemblies
```

#### Example:

```
services.SuperNova(settings => settings.AddDatabase(connection => connection.UseMySql(connection))
.UseOtherAssemblies(List<Assembly> assemblies)))
```

# 8.15.2.9 UseOtherAssemblies() [2/2]

Using BlackHole Entities and BlackHole Services ONLY from the specified Assemblies.

#### **Parameters**

otherAssemblies List of Assemblies

#### Example:

# 8.15.2.10 UseAdditionalAssembly()

```
\label{thm:configuration} \mbox{\tt ConnectionAdditionalSettings.UseAdditionalAssembly ($$Assembly additionalAssembly)$}
```

Using BlackHole Entities and BlackHole Services from the specified Assembly and the Calling Assembly.

#### **Parameters**

```
additional Assembly Full Assembly
```

#### Example:

```
services.SuperNova(settings => settings.AddDatabase(connection => connection.UseMySql(connection
.UseAdditionalAssembly(Assembly assembly)))
```

# 8.15.2.11 UseAdditionalAssemblies() [1/2]

```
\label{thm:configuration.ConnectionAdditionalSettings. Use Additional Assemblies ( \\ List < Assembly > additional Assemblies )
```

Using BlackHole Entities and BlackHole Services from the specified Assemblies and the Calling Assembly.

#### **Parameters**

```
additionalAssemblies List of Assemblies
```

```
services.SuperNova(settings => settings.AddDatabase(connection => connection.UseMySql(connection
.UseAdditionalAssemblies(List<Assembly> assemblies)))
```

#### 8.15.2.12 UseAdditionalAssemblies() [2/2]

```
\label{eq:configuration.ConnectionAdditionalSettings.UseAdditionalAssemblies ( \\ Action < List < Assembly > \\ additionalAssemblies )
```

Using BlackHole Entities and BlackHole Services from the specified Assemblies and the Calling Assembly.

#### **Parameters**

```
additional Assemblies List of Assemblies
```

# Example:

# 8.16 BlackHole.Configuration.ConnectionSettings Class Reference

#### **Public Member Functions**

- ConnectionAdditionalSettings UseSqlServer (string connectionString)
- ConnectionAdditionalSettings UseNpgSql (string connectionString)
- ConnectionAdditionalSettings UseMySql (string connectionString)
- ConnectionAdditionalSettings UseSqlite (string databaseName)
- ConnectionAdditionalSettings UseOracle (string connectionString)

# 8.16.1 Detailed Description

Basic settings to connect to a database server. Choose one of the 5 supported databases type. Insert the connection string.

# 8.16.2 Member Function Documentation

#### 8.16.2.1 UseSqlServer()

```
{\tt ConnectionAdditionalSettings~BlackHole.Configuration.ConnectionSettings.UseSqlServer~(string~connectionString~)}
```

Use the data provider for Microsoft Sql Server.

Do not use the Name of an Existing Database.

BlackHole is going to create the database, based on the connection string.

connectionString | connection string to the database

Returns

Additional Settings

#### Example:

services.SuperNova(settings => settings.AddDatabase(connection => connection.UseSqlServer(co

# 8.16.2.2 UseNpgSql()

Use the data provider for Postgresql.

Do not use the Name of an Existing Database.

BlackHole is going to create the database, based on the connection string.

**Parameters** 

connectionString connection string to the database

Returns

Additional Settings

# Example:

services.SuperNova(settings => settings.AddDatabase(connection => connection.UseNpgSql(connection)

# 8.16.2.3 UseMySql()

 ${\tt ConnectionAdditionalSettings~BlackHole.Configuration.ConnectionSettings.UseMySql~(string~connectionString~)}$ 

Use the data provider for MySql.

Do not use the Name of an Existing Database.

BlackHole is going to create the database, based on the connection string.

connectionString | connection string to the database

Returns

Additional Settings

#### Example:

services.SuperNova(settings => settings.AddDatabase(connection => connection.UseMySql(connection)

#### 8.16.2.4 UseSqlite()

Use the data provider for Sqlite.

The Sqlite database is stored in the Default BlackHole DataPath. You can only choose the file name here.

If you need to move it elsewhere you have to use BlackHole.Configuration.DataPathSettings.SetDataPath

**Parameters** 

databaseName | Just the name of the database

Returns

Additional Settings

# Example:

services.SuperNova(settings => settings.AddDatabase(connection => connection.UseSqlite(connection)

#### 8.16.2.5 UseOracle()

```
{\tt ConnectionAdditionalSettings~BlackHole.Configuration.ConnectionSettings.UseOracle~(string~connectionString~)}
```

Use the data provider for Oracle database.

BlackHole can not setup an oracle database on your system.

Make sure to install the database first and then this library will create the tables.

connectionString | connection string to the database

Returns

Additional Settings

#### Example:

services.SuperNova(settings => settings.AddDatabase(connection => connection.UseOracle(connection)

# 8.17 BlackHole.Configuration.DataPathSettings Class Reference

#### **Public Member Functions**

- void SetDataPath (string dataPath)
- void SetDataPathWithoutLogging (string dataPath)
- void SetDataPath (string dataPath, bool useLogsCleaner)
- void SetDataPath (string dataPath, int cleanUpDays)
- void SetLogsCleanUpDays (int cleanUpDays)
- · void DisableLogging ()
- void DisableLogsCleaner ()

# 8.17.1 Detailed Description

Settings for the path of the BlackHole logs and Sqlite database. Also contains settings for the automatic logs cleaner

This setting is not required.

The default data path is in Current User's folder/BlackHoleData

The logger is enabled by default.

The logs cleaner cleans the logs that are older than 60 days by default.

#### 8.17.2 Member Function Documentation

# 8.17.2.1 SetDataPath() [1/3]

```
void BlackHole.Configuration.DataPathSettings.SetDataPath ( string \ dataPath \ )
```

Set the path of the folder where BlackHole will store Sqlite databases, Logs and other data that will be required for the features in later updates.

dataPath Full path of the data folder

# Example:

services.SuperNova(settings => settings.AddDatabase(connection => connection.UseSqlite(connection)

# 8.17.2.2 SetDataPathWithoutLogging()

```
\label{thm:configuration} \mbox{\tt NataPathSettings.SetDataPathWithoutLogging (string $dataPath$)} \\
```

Set the path of the folder where BlackHole will store Sqlite databases and other data that will be required for the features in later updates.

Also disables the error logs.

#### **Parameters**

dataPath	Full path of the data folder
----------	------------------------------

#### Example:

services.SuperNova(settings => settings.AddDatabase(connection => connection.UseSqlite(connection)

# 8.17.2.3 SetDataPath() [2/3]

```
void BlackHole.Configuration.DataPathSettings.SetDataPath ( string \ dataPath, \\ bool \ useLogsCleaner )
```

Set the path of the folder where BlackHole will store Sqlite databases, Logs and other data that will be required for the features in later updates.

#### **Parameters**

	dataPath	Full path of the data folder	
useLogsCleaner Choose if the logs will be automatically clea		Choose if the logs will be automatically cleaned	

services.SuperNova(settings => settings.AddDatabase(connection => connection.UseSqlite(connection)

# 8.17.2.4 SetDataPath() [3/3]

Set the path of the folder where BlackHole will store Sqlite databases, Logs and other data that will be required for the features in later updates.

#### **Parameters**

dataPath Full path of the data folder	
cleanUpDays	Choose the age of the logs before they get deleted

#### Example:

services.SuperNova(settings => settings.AddDatabase(connection => connection.UseSqlite(connection)

# 8.17.2.5 SetLogsCleanUpDays()

```
\label{local_poly_solution} \mbox{Void BlackHole.Configuration.DataPathSettings.SetLogsCleanUpDays (} \\ \mbox{int } \mbox{\it cleanUpDays} \mbox{\ )}
```

Uses the default data path for the logs, and once per day cleans up the logs that are older than the selected days.

# **Parameters**

```
cleanUpDays
```

# Example:

services.SuperNova(settings => settings.AddDatabase(connection => connection.UseSqlite(connection)

# 8.17.2.6 DisableLogging()

 $\verb|void BlackHole.Configuration.DataPathSettings.DisableLogging ( )|\\$ 

Disables the error logs and using the default folder path for Sqlite and other data that will be required for the features in later updates.

#### Example:

services.SuperNova(settings => settings.AddDatabase(connection => connection.UseSqlite(connection)

# 8.17.2.7 DisableLogsCleaner()

```
void BlackHole.Configuration.DataPathSettings.DisableLogsCleaner ( )
```

Disbales the logs cleaner thread and using the default folder path for Sqlite, Logs and other data that will be required for the features in later updates.

Warning. By disabling the logs cleaner, possible error log files will pile up over time, taking storage in your system

#### Example:

services.SuperNova(settings => settings.AddDatabase(connection => connection.UseSqlite(connection)

# 8.18 BlackHole.Configuration.EntitiesWithNamespace Class Reference

# **Public Member Functions**

- AssembliesUsed UseEntitiesInNamespace (string entityNamespace)
- AssembliesUsed UseEntitiesInNamespaces (List< string > entityNamespaces)
- void UseOtherAssembly (Assembly otherAssembly)

# 8.18.1 Detailed Description

Settings for the Entities Namespaces that will be used.
Part of BlackHole.Configuration.ConnectionAdditionalSettings

# 8.18.2 Member Function Documentation

#### 8.18.2.1 UseEntitiesInNamespace()

```
AssembliesUsed BlackHole.Configuration.EntitiesWithNamespace.UseEntitiesInNamespace ( string entityNamespace )
```

Using only the Entities that are in the specified Namespace.

entityNamespace | Namespace Full Name 'MyProject.Entities....'

Returns

BlackHole.Configuration.AssembliesUsed

#### Example:

services.SuperNova(settings => settings.AddDatabase(connection => connection.UseMySql(connection)
.AddServicesFromNamespaces(List<string> NamespacesFullNames).UseEntitiesInNamespaces

# 8.18.2.2 UseEntitiesInNamespaces()

```
AssembliesUsed BlackHole.Configuration.EntitiesWithNamespace.UseEntitiesInNamespaces (
List< string > entityNamespaces)
```

Using only the Entities in the specified Namespaces.

#### **Parameters**

entityNamespaces | List of Namespaces Full Names

Returns

BlackHole.Configuration.AssembliesUsed

# Example:

services.SuperNova(settings => settings.AddDatabase(connection => connection.UseMySql(connection.AddServicesFromNamespaces(List<string> NamespacesFullNames).UseEntitiesInNamespacesFullNames

#### 8.18.2.3 UseOtherAssembly()

```
void BlackHole.Configuration.EntitiesWithNamespace.UseOtherAssembly ( {\tt Assembly}\ other {\tt Assembly}\ )
```

Scans a specified assembly for BlackHole Entities and Services and uses only them.

```
otherAssembly Full Assembly
```

# Example:

services.SuperNova(settings => settings.AddDatabase(connection => connection.UseMySql(connection)
.AddServicesFromNamespaces(List<string> NamespacesFullNames).UseOtherAssemble

# 8.19 BlackHole.Entities.ForeignKey Class Reference

Inherits Attribute.

# **Public Member Functions**

- ForeignKey (Type table, bool isNullable)
- ForeignKey (Type table)

# 8.19.1 Detailed Description

It's an Attribute that can be used on the Entity's properties

Sets Foreign Key for this Column.

# 8.19.2 Constructor & Destructor Documentation

# 8.19.2.1 ForeignKey() [1/2]

```
BlackHole.Entities.ForeignKey.ForeignKey (  \label{eq:table} {\it Type table,}  bool isNullable )
```

This Overload of the Constructor Sets by Default the corresponding column on the Primary Table as Id. You Can choose the Primary Table and if the Foreign Key is Nullable.

#### **Parameters**

table	Type of the parent Table	
isNullable	Is this Column Nullable?	

```
[ForeignKey(typeof(Customer), false)]
public int CustomerId {get;set;}
```

# 8.19.2.2 ForeignKey() [2/2]

This Overload of the Constructor Sets by Default the corresponding column on the Primary Table as Id and makes the Foreign Key Column Nullable. You Can choose the Primary Table.

#### **Parameters**

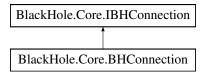
```
table Type of the parent Table
```

#### Example:

```
[ForeignKey(typeof(Customer))]
public int CustomerId {get;set;}
```

# 8.20 BlackHole.Core.IBHConnection Interface Reference

Inheritance diagram for BlackHole.Core.IBHConnection:



# **Public Member Functions**

- G? ExecuteScalar < G > (string commandText)
- G? ExecuteScalar < G > (string commandText, BHParameters parameters)
- G? ExecuteScalar< G > (string commandText, object parametersObject)
- G? ExecuteScalar < G > (string commandText, BHParameters parameters, BHTransaction bHTransaction)
- G? ExecuteScalar < G > (string commandText, BHTransaction bHTransaction)
- G? ExecuteScalar < G > (string commandText, object parametersObject, BHTransaction bHTransaction)
- Task< G?> ExecuteScalarAsync< G > (string commandText)
- Task< G?> ExecuteScalarAsync< G > (string commandText, BHParameters parameters)
- Task< G?> ExecuteScalarAsync< G > (string commandText, object parametersObject)
- Task< G?> ExecuteScalarAsync< G > (string commandText, BHTransaction bHTransaction)
- Task< G?> ExecuteScalarAsync< G > (string commandText, BHParameters parameters, BHTransaction bHTransaction)

- Task< G?> ExecuteScalarAsync< G > (string commandText, object parametersObject, BHTransaction b← HTransaction)
- bool JustExecute (string commandTex)
- bool JustExecute (string commandText, BHParameters parameters)
- bool JustExecute (string commandText, object parametersObject)
- bool JustExecute (string commandText, BHTransaction bHTransaction)
- bool JustExecute (string commandText, BHParameters parameters, BHTransaction bHTransaction)
- bool JustExecute (string commandText, object parametersObject, BHTransaction bHTransaction)
- Task< bool > JustExecuteAsync (string commandText)
- Task < bool > JustExecuteAsync (string commandText, BHParameters parameters)
- Task< bool > JustExecuteAsync (string commandText, object parametersObject)
- Task< bool > JustExecuteAsync (string commandText, BHTransaction bHTransaction)
- Task< bool > JustExecuteAsync (string commandText, BHParameters parameters, BHTransaction b← HTransaction)
- Task< bool > JustExecuteAsync (string commandText, object parametersObject, BHTransaction b← HTransaction)
- T? QueryFirst< T > (string commandText)
- T? QueryFirst < T > (string commandText, BHParameters parameters)
- T? QueryFirst< T > (string commandText, object parametersObject)
- T? QueryFirst< T > (string commandText, BHTransaction bHTransaction)
- T? QueryFirst< T > (string commandText, BHParameters parameters, BHTransaction bHTransaction)
- T? QueryFirst < T > (string commandText, object parametersObject, BHTransaction bHTransaction)
- List< T > Query< T > (string commandText)
- List< T > Query< T > (string commandText, BHParameters parameters)
- List< T > Query< T > (string commandText, object parametersObject)
- List< T > Query< T > (string commandText, BHTransaction bHTransaction)
- List< T > Query< T > (string commandText, BHParameters parameters, BHTransaction bHTransaction)
- List< T > Query< T > (string commandText, object parametersObject, BHTransaction bHTransaction)
- Task< T?> QueryFirstAsync< T > (string commandText)
- Task < T?> QueryFirstAsync < T > (string commandText, BHParameters parameters)
- Task< T?> QueryFirstAsync< T > (string commandText, object parametersObject)
- Task< T?> QueryFirstAsync< T > (string commandText, BHTransaction bHTransaction)
- Task< T?> QueryFirstAsync< T > (string commandText, BHParameters parameters, BHTransaction b← HTransaction)
- Task< T?> QueryFirstAsync< T > (string commandText, object parametersObject, BHTransaction b← HTransaction)
- Task< List< T >> QueryAsync< T > (string commandText)
- Task< List< T >> QueryAsync< T > (string commandText, BHParameters parameters)
- Task< List< T >> QueryAsync< T > (string commandText, object parametersObject)
- Task< List< T >> QueryAsync< T > (string commandText, BHTransaction bHTransaction)
- Task< List< T > > QueryAsync< T > (string commandText, BHParameters parameters, BHTransaction bHTransaction)
- Task< List< T > > QueryAsync< T > (string commandText, object parametersObject, BHTransaction b← HTransaction)

# 8.20.1 Detailed Description

An Interface that gives all the required methods to perform custom sql commands

It's already registered in the ServiceCollection and it can be used to your services with Dependency Injection

The connection is automatically generated and disposed after each execution

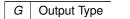
# 8.20.2 Member Function Documentation

# 8.20.2.1 ExecuteScalar < G >() [1/6]

```
G? BlackHole.Core.IBHConnection.ExecuteScalar< G > ( string commandText )
```

Classic Execute Scalar

**Template Parameters** 



#### **Parameters**

```
commandText | Command Text
```

Returns

The First Value of the Result

# Example:

```
IBHConnection connection = new BHConnection();
int result = connection.ExecuteScalar<int>(string command);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

# 8.20.2.2 ExecuteScalar < G >() [2/6]

```
G? BlackHole.Core.IBHConnection.ExecuteScalar<br/>< G > ( string\ commandText, BHParameters\ parameters\ )
```

Classic Execute Scalar with BHParameters

**Template Parameters** 

G Output Type

commandText	Command Text
parameters BHParameters Class, populated with black hole parameter	

# Returns

The First Value of the Result

# Example:

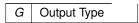
```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();
parameters.Add("Price", 5.5);
int result = connection.ExecuteScalar<int>(string command, parameters);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

# 8.20.2.3 ExecuteScalar < G >() [3/6]

Classic Execute Scalar with Object as Parameters

# **Template Parameters**



#### **Parameters**

	commandText	Command Text	
Ī	parametersObject	Class with properties as Parameters	

#### Returns

The First Value of the Result

# Example:

IBHConnection connection = new BHConnection();

```
var someClass = new SomeClass{
    Price = 5.5
}
int result = connection.ExecuteScalar<int>(string command, someClass);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

# 8.20.2.4 ExecuteScalar < G >() [4/6]

Transaction. Classic Execute Scalar with BHParameters

#### **Template Parameters**

```
G Output Type
```

# **Parameters**

commandText   Command Text	
parameters BHParameters Class, populated with black hole parameters	
bHTransaction	BHTransaction Class, contains connection and transaction

# Returns

The First Value of the Result

# Example:

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();

parameters.Add("Price", 5.5);

using(BHTransaction transaction = new BHTransaction())
{
   int result = connection.ExecuteScalar<int>(string command, parameters, transaction);
}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

# 8.20.2.5 ExecuteScalar < G >() [5/6]

```
G? BlackHole.Core.IBHConnection.ExecuteScalar<br/>< G > ( string\ commandText, BHTransaction\ bHTransaction\ )
```

Transaction. Classic Execute Scalar

**Template Parameters** 

```
G Output Type
```

#### **Parameters**

commandText	Command Text
bHTransaction	BHTransaction Class, contains connection and transaction

# Returns

The First Value of the Result

# Example:

```
IBHConnection connection = new BHConnection();
using(BHTransaction transaction = new BHTransaction())
{
   int result = connection.ExecuteScalar<int>(string command, transaction);
}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

# 8.20.2.6 ExecuteScalar < G >() [6/6]

Transaction. Classic Execute Scalar with Object as Parameters

# **Template Parameters**

G Output Type

commandText Command Text		Command Text	l
parametersObject Class with properties as Parameters		Class with properties as Parameters	1
	bHTransaction	BHTransaction Class, contains connection and transaction	ĺ

#### Returns

The First Value of the Result

#### Example:

```
IBHConnection connection = new BHConnection();

var someClass = new SomeClass{
    Price = 5.5
}

using(BHTransaction transaction = new BHTransaction())
{
    int result = connection.ExecuteScalar<int>(string command, someClass, transaction);
}
```

 $\textbf{Tip:} \ \ \text{For Oracle and Postgres} \ \ , \ \ \text{Double Quotes are required for the Table and Column Names in your command text}$ 

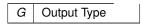
Implemented in BlackHole.Core.BHConnection.

# 8.20.2.7 ExecuteScalarAsync < G >() [1/6]

```
Task< G?> BlackHole.Core.IBHConnection.ExecuteScalarAsync< G > ( string\ commandText\ )
```

Asyncronous. Classic Execute Scalar

#### **Template Parameters**



# **Parameters**

commandText	Command Text
-------------	--------------

# Returns

The First Value of the Result

```
IBHConnection connection = new BHConnection();
int result = await connection.ExecuteScalarAsync<int>(string command);
```

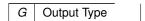
**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

# 8.20.2.8 ExecuteScalarAsync < G >() [2/6]

```
Task< G?> BlackHole.Core.IBHConnection.ExecuteScalarAsync< G > ( string \ commandText, \\ BHParameters \ parameters \ )
```

Asyncronous. Classic Execute Scalar with BHParameters

# **Template Parameters**



#### **Parameters**

commandText	Command Text
parameters	BHParameters Class, populated with black hole parameters

#### Returns

The First Value of the Result

#### Example:

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();
parameters.Add("Price", 5.5);
int result = await connection.ExecuteScalarAsync<int>(string command, parameters);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

# 8.20.2.9 ExecuteScalarAsync < G >() [3/6]

```
Task< G?> BlackHole.Core.IBHConnection.ExecuteScalarAsync< G > ( string \ commandText, \\ object \ parametersObject \ )
```

Asyncronous. Classic Execute Scalar with Object as Parameters

|--|

#### **Parameters**

commandText	Command Text
parametersObject	Class with properties as Parameters

## Returns

The First Value of the Result

## Example:

```
IBHConnection connection = new BHConnection();
var someClass = new SomeClass{
    Price = 5.5
}
int result = await connection.ExecuteScalarAsync<int>(string command, someClass);
```

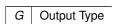
**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

## 8.20.2.10 ExecuteScalarAsync< G >() [4/6]

```
Task< G?> BlackHole.Core.IBHConnection.ExecuteScalarAsync< G > ( string \ commandText, \\ BHTransaction \ bHTransaction \ )
```

Asyncronous. Transaction. Classic Execute Scalar

## **Template Parameters**



#### **Parameters**

commandText	Command Text	
bHTransaction	BHTransaction Class, contains connection and transaction	1

#### Returns

The First Value of the Result

## Example:

```
IBHConnection connection = new BHConnection();
using(BHTransaction transaction = new BHTransaction())
{
   int result = await connection.ExecuteScalarAsync<int>(string command, transaction);
}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

# 8.20.2.11 ExecuteScalarAsync< G >() [5/6]

Asyncronous. Transaction. Classic Execute Scalar with BHParameters

## **Template Parameters**

```
G Output Type
```

#### **Parameters**

commandText	Command Text
parameters	BHParameters Class, populated with black hole parameters
bHTransaction	BHTransaction Class, contains connection and transaction

## Returns

The First Value of the Result

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();
parameters.Add("Price", 5.5);
using(BHTransaction transaction = new BHTransaction()){
```

```
int result = await connection.ExecuteScalarAsync<int>(string command, parameters, transa)
```

## 8.20.2.12 ExecuteScalarAsync< G >() [6/6]

Asyncronous. Transaction. Classic Execute Scalar with Object as Parameters

#### **Template Parameters**

_
---

#### **Parameters**

commandText	Command Text
parametersObject	Class with properties as Parameters
bHTransaction	BHTransaction Class, contains connection and transaction

#### Returns

The First Value of the Result

## Example:

```
IBHConnection connection = new BHConnection();

var someClass = new SomeClass{
    Price = 5.5
}

using(BHTransaction transaction = new BHTransaction())
{
    int result = await connection.ExecuteScalarAsync<int>(string command, someClass, transaction)}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

## 8.20.2.13 JustExecute() [1/6]

```
bool BlackHole.Core.IBHConnection.JustExecute ( string\ commandTex\ )
```

Classic Execute without output.

#### **Parameters**

```
commandText | Command Text
```

#### Returns

Success Boolean

#### Example:

```
IBHConnection connection = new BHConnection();
bool success = connection.JustExecute(string command);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

## 8.20.2.14 JustExecute() [2/6]

Classic Execute with BHParameters.

#### **Parameters**

commandText	Command Text
parameters	BHParameters Class, populated with black hole parameters

## Returns

Success Boolean

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();
parameters.Add("Price", 5.5);
```

```
bool success = connection.JustExecute(string command, parameters);
```

## 8.20.2.15 JustExecute() [3/6]

Classic Execute with Object as Parameters.

#### **Parameters**

commandText	Command Text
parametersObject	Class with properties as Parameters

#### Returns

Success Boolean

## Example:

```
IBHConnection connection = new BHConnection();

var someClass = new SomeClass{
    Price = 5.5
}

bool success = connection.JustExecute(string command, someClass);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

## 8.20.2.16 JustExecute() [4/6]

Transaction. Classic Execute without output.

#### **Parameters**

commandText	Command Text	
bHTransaction	BHTransaction Class, contains connection and transaction	

#### Returns

Success Boolean

## Example:

```
IBHConnection connection = new BHConnection();
using(BHTransaction transaction = new BHTransaction())
{
    bool success = connection.JustExecute(string command, transaction);
}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

# 8.20.2.17 JustExecute() [5/6]

Transaction. Classic Execute without output.

## **Parameters**

comma	andText	Command Text
parame	eters	BHParameters Class, populated with black hole parameters
bHTrar	saction	BHTransaction Class, contains connection and transaction

#### Returns

Success Boolean

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();
parameters.Add("Price", 5.5);
```

```
using(BHTransaction transaction = new BHTransaction())
{
   bool success = connection.JustExecute(string command, parameters, transaction);
}
```

#### 8.20.2.18 JustExecute() [6/6]

Transaction. Classic Execute with Object as Parameters.

#### **Parameters**

commandText	Command Text
parametersObject	Class with properties as Parameters
bHTransaction	BHTransaction Class, contains connection and transaction

# Returns

Success Boolean

#### Example:

```
IBHConnection connection = new BHConnection();

var someClass = new SomeClass{
    Price = 5.5
}

using(BHTransaction transaction = new BHTransaction())
{
    bool success = connection.JustExecute(string command, someClass, transaction);
}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

#### 8.20.2.19 JustExecuteAsync() [1/6]

Asyncronous. Classic Execute without output.

#### **Parameters**

```
commandText | Command Text
```

## Returns

Success Boolean

#### Example:

```
IBHConnection connection = new BHConnection();
bool success = await connection.JustExecuteAsync(string command);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

#### 8.20.2.20 JustExecuteAsync() [2/6]

```
Task<br/>< bool > BlackHole.Core.IBHConnection.JustExecuteAsync ( string \ commandText, BHParameters \ parameters )
```

Asyncronous. Classic Execute with BHParameters.

#### **Parameters**

commandText	Command Text
parameters	BHParameters Class, populated with black hole parameters

#### Returns

Success Boolean

## Example:

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();
parameters.Add("Price", 5.5);
bool success = await connection.JustExecuteAsync(string command, parameters);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

#### 8.20.2.21 JustExecuteAsync() [3/6]

Asyncronous. Classic Execute with Object as Parameters.

#### **Parameters**

I	commandText	Command Text
	parametersObject	Class with properties as Parameters

#### Returns

Success Boolean

## Example:

```
IBHConnection connection = new BHConnection();

var someClass = new SomeClass{
    Price = 5.5
}

bool success = await connection.JustExecuteAsync(string command, someClass);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

#### 8.20.2.22 JustExecuteAsync() [4/6]

```
\label{total command}  \mbox{Task< bool} > \mbox{BlackHole.Core.IBHConnection.JustExecuteAsync (} \\ \mbox{string } \mbox{commandText,} \\ \mbox{BHTransaction } \mbox{bHTransaction )}
```

Asyncronous. Transaction. Classic Execute without output.

#### **Parameters**

commandText	Command Text
bHTransaction	BHTransaction Class, contains connection and transaction

#### Returns

Success Boolean

```
IBHConnection connection = new BHConnection();
using(BHTransaction transaction = new BHTransaction())
{
    bool success =await connection.JustExecuteAsync(string command, transaction);
}
```

#### 8.20.2.23 JustExecuteAsync() [5/6]

Asyncronous. Transaction. Classic Execute with BHParameters.

#### **Parameters**

commandText	Command Text
parameters	BHParameters Class, populated with black hole parameters
bHTransaction	BHTransaction Class, contains connection and transaction

#### Returns

Success Boolean

## Example:

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();

parameters.Add("Price", 5.5);

using(BHTransaction transaction = new BHTransaction())
{
   bool success = await connection.JustExecuteAsync(string command, parameters, transaction}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

#### 8.20.2.24 JustExecuteAsync() [6/6]

Asyncronous. Transaction. Classic Execute with Object as Parameters.

#### **Parameters**

commandText	Command Text
parametersObject	Class with properties as Parameters
bHTransaction	BHTransaction Class, contains connection and transaction

#### Returns

Success Boolean

# Example:

```
IBHConnection connection = new BHConnection();

var someClass = new SomeClass{
    Price = 5.5
}

using(BHTransaction transaction = new BHTransaction())
{
    bool success = await connection.JustExecuteAsync(string command, someClass, transaction)}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

## 8.20.2.25 QueryFirst< T >() [1/6]

```
T? BlackHole.Core.IBHConnection.QueryFirst< T > ( string commandText )
```

A Query that returns only the first Line of the result.

# **Template Parameters**

T   Output Type

#### **Parameters**

Command Text

## Returns

The First Line of the Result

#### Example:

```
IBHConnection connection = new BHConnection();
Customer? customer = connection.QueryFirst<Customer>(string command);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

## 8.20.2.26 QueryFirst< T >() [2/6]

```
T? BlackHole.Core.IBHConnection.QueryFirst< T > ( string commandText, BHParameters parameters)
```

A Query that takes BHParameters and returns only the first Line of the result.

# **Template Parameters**

```
T Output Type
```

## **Parameters**

commandText	Command Text	
parameters	BHParameters Class, populated with black hole parameters	]

#### Returns

The First Line of the Result

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();
parameters.Add("FirstName", "Nick");
Customer? customer = connection.QueryFirst<Customer>(string command, parameters);
```

#### 8.20.2.27 QueryFirst < T >() [3/6]

```
T? BlackHole.Core.IBHConnection.QueryFirst< T > ( string commandText, object parametersObject )
```

A Query that takes an Object as parameters and returns only the first Line of the result.

#### **Template Parameters**

```
T Output Type
```

#### **Parameters**

commandText	Command text
parametersObject	Class with properties as Parameters

#### Returns

The First Line of the Result

## Example:

```
IBHConnection connection = new BHConnection();
var someClass = new SomeClass{
    FirstName = "Nick"
};
Customer? customer = connection.QueryFirst<Customer>(string command, someClass);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

# 8.20.2.28 QueryFirst< T >() [4/6]

```
T? BlackHole.Core.IBHConnection.QueryFirst< T > ( string commandText, BHTransaction bHTransaction )
```

Transaction. A Query that returns only the first Line of the result.

Τ	Output Type

#### **Parameters**

commandText	Command Text
bHTransaction	BHTransaction Class, contains connection and transaction

#### Returns

The First Line of the Result

# Example:

```
IBHConnection connection = new BHConnection();
using(BHTransaction transaction = new BHTransaction())
{
   Customer? customer = connection.QueryFirst<Customer>(string command, transaction);
}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

# 8.20.2.29 QueryFirst< T >() [5/6]

Transaction. A Query that takes BHParameters and returns only the first Line of the result.

#### **Template Parameters**

Т	Output Type	
---	-------------	--

#### **Parameters**

commandText	Command Text
parameters	BHParameters Class, populated with black hole parameters
bHTransaction	BHTransaction Class, contains connection and transaction

#### Returns

The First Line of the Result

## Example:

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();

parameters.Add("FirstName", "Nick");

using(BHTransaction transaction = new BHTransaction())
{
    Customer? customer = connection.QueryFirst<Customer>(string command, parameters, transact)}
```

**Tip:** For Oracle and Postgres , Double Quotes are required for the Table and Column Names in your command text

Implemented in BlackHole.Core.BHConnection.

#### 8.20.2.30 QueryFirst< T >() [6/6]

Transaction. A Query that takes an Object as parameters and returns only the first Line of the result.

# **Template Parameters**

Τ	Output Type

#### **Parameters**

commandText	Command Text
parametersObject	Class with properties as Parameters
bHTransaction	BHTransaction Class, contains connection and transaction

## Returns

The First Line of the Result

```
IBHConnection connection = new BHConnection();
var someClass = new SomeClass{
    Price = 5.5
```

```
}
using(BHTransaction transaction = new BHTransaction())
{
    Customer? customer = connection.QueryFirst<Customer>(string command, someClass, transact)
}
```

## 8.20.2.31 Query < T >() [1/6]

```
List< T > BlackHole.Core.IBHConnection.Query< T > ( string \ commandText \ )
```

A Query that returns all Lines of the Result as List.

#### **Template Parameters**



#### **Parameters**

```
commandText | Command Text
```

## Returns

The First Line of the Result

#### Example:

```
IBHConnection connection = new BHConnection();
List<Customer> customers = connection.Query<Customer>(string command);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

# 8.20.2.32 Query < T >() [2/6]

A Query that takes BHParameters and returns all Lines of the Result as List.

Τ	Output Type

#### **Parameters**

commandText	Command Text
parameters	BHParameters Class, populated with black hole parameters

## Returns

The First Line of the Result

## Example:

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();

parameters.Add("FirstName", "Nick");
List<Customer> customers = connection.Query<Customer>(string command, parameters);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

# 8.20.2.33 Query < T >() [3/6]

```
List< T > BlackHole.Core.IBHConnection.Query< T > ( string\ commandText, object\ parametersObject\ )
```

A Query that takes an Object as parameters and returns all Lines of the Result as List.

## **Template Parameters**

T	Output Type

## **Parameters**

commandText	Command text	
parametersObject	Class with properties as Parameters	]

# Returns

The First Line of the Result

## Example:

```
IBHConnection connection = new BHConnection();

var someClass = new SomeClass{
    FirstName = "Nick"
};

List<Customer> customers = connection.Query<Customer>(string command, someClass);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

```
8.20.2.34 Query < T >() [4/6]
```

```
List< T > BlackHole.Core.IBHConnection.Query< T > ( string\ commandText, BHTransaction\ bHTransaction\ )
```

Transaction. A Query that returns all Lines of the Result as List.

## **Template Parameters**

```
T Output Type
```

#### **Parameters**

cor	mmandText	Command Text
bHTransaction BHTransaction Class, contains connection and		BHTransaction Class, contains connection and transaction

#### Returns

The First Line of the Result

# Example:

```
IBHConnection connection = new BHConnection();
using(BHTransaction transaction = new BHTransaction())
{
    List<Customer> customers = connection.Query<Customer>(string command, transaction);
}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

#### 8.20.2.35 Query < T >() [5/6]

Transaction. A Query that takes BHParameters and returns all Lines of the Result as List.

#### **Template Parameters**

|--|

#### **Parameters**

commandText	Command Text
parameters	BHParameters Class, populated with black hole parameters
bHTransaction BHTransaction Class, contains connection and transaction	

## Returns

The First Line of the Result

#### Example:

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();

parameters.Add("FirstName", "Nick");

using(BHTransaction transaction = new BHTransaction())
{
   List<Customer> customers = connection.Query<Customer>(string command, parameters, transaction)}
```

**Tip:** For Oracle and Postgres , Double Quotes are required for the Table and Column Names in your command text

Implemented in BlackHole.Core.BHConnection.

## 8.20.2.36 Query < T >() [6/6]

Transaction. A Query that takes an Object as parameters and returns all Lines of the Result as List.

T Output Type	
---------------	--

#### **Parameters**

commandText	Command Text
parametersObject	Class with properties as Parameters
bHTransaction BHTransaction Class, contains connection and transaction	

#### Returns

The First Line of the Result

# Example:

```
IBHConnection connection = new BHConnection();

var someClass = new SomeClass{
    Price = 5.5
}

using(BHTransaction transaction = new BHTransaction())
{
    List<Customer> customer = connection.Query<Customer>(string command, someClass, transaction)}
```

 $Implemented \ in \ BlackHole. Core. BHC onnection.$ 

# 8.20.2.37 QueryFirstAsync< T >() [1/6]

```
Task< T?> BlackHole.Core.IBHConnection.QueryFirstAsync< T > ( string \ commandText \ )
```

Asyncronous. A Query that returns only the first Line of the result.

# **Template Parameters**

T	Output Type	
---	-------------	--

#### **Parameters**

commandTe	ext   Command Text

#### Returns

The First Line of the Result

## Example:

```
IBHConnection connection = new BHConnection();
Customer? customer = await connection.QueryFirstAsync<Customer>(string command);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

## 8.20.2.38 QueryFirstAsync< T >() [2/6]

```
Task< T?> BlackHole.Core.IBHConnection.QueryFirstAsync< T > ( string \ commandText, BHParameters \ parameters )
```

Asyncronous. A Query that takes BHParameters and returns only the first Line of the result.

#### **Template Parameters**

```
T Output Type
```

#### **Parameters**

commandText	Command Text
parameters	BHParameters Class, populated with black hole parameters

#### Returns

The First Line of the Result

## Example:

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();

parameters.Add("FirstName", "Nick");

Customer? customer = await connection.QueryFirstAsync<Customer>(string command, parameters);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

## 8.20.2.39 QueryFirstAsync< T >() [3/6]

Asyncronous. A Query that takes an Object as parameters and returns only the first Line of the result.

## **Template Parameters**

```
T Output Type
```

#### **Parameters**

commandText	Command text
parametersObject	Class with properties as Parameters

#### Returns

The First Line of the Result

#### Example:

```
IBHConnection connection = new BHConnection();

var someClass = new SomeClass{
    FirstName = "Nick"
};

Customer? customer = await connection.QueryFirstAsync<Customer>(string command, someClass);
```

Tip: For Oracle and Postgres , Double Quotes are required for the Table and Column Names in your command text

 $Implemented \ in \ BlackHole. Core. BHC onnection.$ 

# 8.20.2.40 QueryFirstAsync< T >() [4/6]

Asyncronous. Transaction. A Query that returns only the first Line of the result.

# **Template Parameters**

T Output Type

#### **Parameters**

commandText	Command Text	
bHTransaction	BHTransaction Class, contains connection and transaction	1

#### Returns

The First Line of the Result

## Example:

```
IBHConnection connection = new BHConnection();

using(BHTransaction transaction = new BHTransaction())
{
    Customer? customer = await connection.QueryFirstAsync<Customer>(string command, transact)
```

**Tip:** For Oracle and Postgres , Double Quotes are required for the Table and Column Names in your command text

Implemented in BlackHole.Core.BHConnection.

# 8.20.2.41 QueryFirstAsync< T >() [5/6]

Asyncronous. Transaction. A Query that takes BHParameters and returns only the first Line of the result.

#### **Template Parameters**

```
T Output Type
```

#### **Parameters**

commandText	Command Text
parameters	BHParameters Class, populated with black hole parameters
bHTransaction	BHTransaction Class, contains connection and transaction

#### Returns

The First Line of the Result

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();

parameters.Add("FirstName", "Nick");

using(BHTransaction transaction = new BHTransaction())
{
   Customer? customer = await connection.QueryFirstAsync<Customer>(string command, parameter)}
```

## 8.20.2.42 QueryFirstAsync< T >() [6/6]

Asyncronous. Transaction. A Query that takes an Object as parameters and returns only the first Line of the result.

## **Template Parameters**

```
T Output Type
```

#### **Parameters**

commandText	Command Text
parametersObject	Class with properties as Parameters
bHTransaction	BHTransaction Class, contains connection and transaction

#### Returns

The First Line of the Result

```
IBHConnection connection = new BHConnection();

var someClass = new SomeClass{
    Price = 5.5
}

using(BHTransaction transaction = new BHTransaction())
{
    Customer? customer = await connection.QueryFirstAsync<Customer>(string command, someClase)
```

#### 8.20.2.43 QueryAsync< T >() [1/6]

```
Task< List< T > > BlackHole.Core.IBHConnection.QueryAsync< T > ( string commandText )
```

Asyncronous. A Query that returns all Lines of the Result as List.

#### **Template Parameters**

```
T Output List Type
```

#### **Parameters**

```
commandText | Command Text
```

#### Returns

List of Lines

#### Example:

```
IBHConnection connection = new BHConnection();
List<Customer> customers = await connection.QueryAsync<Customer>(string command);
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

# 8.20.2.44 QueryAsync< T >() [2/6]

```
Task< List< T >> BlackHole.Core.IBHConnection.QueryAsync< T > ( string commandText, BHParameters parameters )
```

Asyncronous. A Query that takes BHParameters and returns all Lines of the Result as List.

#### **Template Parameters**

```
T Output List Type
```

#### **Parameters**

commandText	Command Text
parameters	BHParameters Class, populated with black hole parameters

#### Returns

List of Lines

## Example:

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();

parameters.Add("FirstName", "Nick");

List<Customer> customers = await connection.QueryAsync<Customer>(string command, parameters)
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

# 8.20.2.45 QueryAsync< T >() [3/6]

```
Task< List< T > BlackHole.Core.IBHConnection.QueryAsync< T > ( string commandText, object parametersObject )
```

Asyncronous. A Query that takes an Object as parameters and returns all Lines of the Result as List.

## **Template Parameters**

```
T Output List Type
```

# **Parameters**

commandText	Command Text
parametersObject	Class with properties as Parameters

# Returns

List of Lines

## Example:

IBHConnection connection = new BHConnection();

```
var someClass = new SomeClass{
    FirstName = "Nick"
};
List<Customer> customers = await connection.QueryAsync<Customer>(string command, someClass);
```

# 8.20.2.46 QueryAsync< T >() [4/6]

```
Task< List< T >> BlackHole.Core.IBHConnection.QueryAsync< T > ( string commandText, BHTransaction bHTransaction )
```

Asyncronous. Transaction. A Query that returns all Lines of the Result as List.

#### **Template Parameters**

```
T Output List Type
```

#### **Parameters**

ſ	commandText	Command Text
	bHTransaction	BHTransaction Class, contains connection and transaction

#### Returns

List of Lines

#### Example:

```
IBHConnection connection = new BHConnection();
using(BHTransaction transaction = new BHTransaction())
{
    List<Customer> customers = await connection.QueryAsync<Customer>(string command, transaction)}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

## 8.20.2.47 QueryAsync< T >() [5/6]

```
Task< List< T > BlackHole.Core.IBHConnection.QueryAsync< T > ( string commandText, BHParameters parameters, BHTransaction bHTransaction )
```

Asyncronous. Transaction. A Query that takes BHParameters and returns all Lines of the Result as List.

#### **Template Parameters**

```
T Output List Type
```

#### **Parameters**

commandText	Command Text
parameters	BHParameters Class, populated with black hole parameters
bHTransaction	BHTransaction Class, contains connection and transaction

## Returns

List of Lines

#### Example:

```
IBHConnection connection = new BHConnection();
BHParameters parameters = new BHParameters();

parameters.Add("FirstName", "Nick");

using(BHTransaction transaction = new BHTransaction())
{
    List<Customer> customers = await connection.QueryAsync<Customer>(string command, parameter)}
```

**Tip:** For Oracle and Postgres, Double Quotes are required for the Table and Column Names in your command text Implemented in BlackHole.Core.BHConnection.

## 8.20.2.48 QueryAsync< T >() [6/6]

Asyncronous. Transaction. A Query that takes an Object as parameters and returns all Lines of the Result as List.

T Output List Type	
--------------------	--

#### **Parameters**

commandText	Command Text	
parametersObject	Class with properties as Parameters	
bHTransaction	BHTransaction Class, contains connection and transaction	

#### Returns

List of Lines

## Example:

```
IBHConnection connection = new BHConnection();

var someClass = new SomeClass{
    Price = 5.5
}

using(BHTransaction transaction = new BHTransaction())
{
    List<Customer> customers = await connection.QueryAsync<Customer>(string command, someClass)
```

Implemented in BlackHole.Core.BHConnection.

# 8.21 BlackHole.Core.IBHDataProvider< T, G > Interface Template Reference

Inheritance diagram for BlackHole.Core.IBHDataProvider< T, G >:



## **Public Member Functions**

- List< T > GetAllEntries ()
- List< T > GetAllEntries (BHTransaction transaction)
- List< Dto > GetAllEntries< Dto > ()
- List< Dto > GetAllEntries< Dto > (BHTransaction transaction)
- List< T > GetAllInactiveEntries ()
- List< T > GetAllInactiveEntries (BHTransaction transaction)

- T? GetEntryByld (G ld)
- T? GetEntryById (G Id, BHTransaction transaction)
- Dto? GetEntryById< Dto > (G Id)
- Dto? GetEntryById < Dto > (G Id, BHTransaction transaction)
- T? GetEntryWhere (Expression < Func < T, bool > > predicate)
- T? GetEntryWhere (Expression < Func < T, bool > > predicate, BHTransaction transaction)
- Dto? GetEntryWhere < Dto > (Expression < Func < T, bool > > predicate)
- Dto? GetEntryWhere< Dto > (Expression< Func< T, bool > > predicate, BHTransaction transaction)
- List< T > GetEntriesWhere (Expression< Func< T, bool > > predicate)
- List< T > GetEntriesWhere (Expression< Func< T, bool > > predicate, BHTransaction transaction)
- List< Dto > GetEntriesWhere< Dto > (Expression< Func< T, bool > > predicate)
- G? InsertEntry (T entry)
- G? InsertEntry (T entry, BHTransaction transaction)
- List< G?> InsertEntries (List< T > entries)
- List< G?> InsertEntries (List< T > entries, BHTransaction transaction)
- bool UpdateEntryById (T entry)
- bool UpdateEntryByld (T entry, BHTransaction transaction)
- bool UpdateEntryById< Columns > (T entry)
- bool UpdateEntryById< Columns > (T entry, BHTransaction transaction)
- bool UpdateEntriesByld (List< T > entries)
- bool UpdateEntriesByld (List< T > entries, BHTransaction transaction)
- bool UpdateEntriesById< Columns > (List< T > entries)
- bool UpdateEntriesByld< Columns > (List< T > entries, BHTransaction transaction)
- bool UpdateEntriesWhere (Expression< Func< T, bool > > predicate, T entry)
- bool UpdateEntriesWhere (Expression< Func< T, bool > > predicate, T entry, BHTransaction transaction)
- bool UpdateEntriesWhere < Columns > (Expression < T, bool > > predicate, Columns entry)
- bool UpdateEntriesWhere< Columns > (Expression< Func< T, bool > > predicate, Columns entry, BHTransaction transaction)
- bool DeleteAllEntries ()
- bool DeleteAllEntries (BHTransaction transaction)
- bool DeleteEntryById (G Id)
- bool DeleteEntryByld (G Id, BHTransaction transaction)
- bool DeleteInactiveEntryById (G Id)
- bool DeleteInactiveEntryById (G Id, BHTransaction transaction)
- bool ReactivateEntryById (G Id)
- bool ReactivateEntryByld (G ld, BHTransaction transaction)
- bool DeleteEntriesWhere (Expression< Func< T, bool > > predicate)
- bool DeleteEntriesWhere (Expression < Func < T, bool > > predicate, BHTransaction transaction)
- Task< List< T >> GetAllEntriesAsync ()
- Task< List< T > > GetAllEntriesAsync (BHTransaction transaction)
- Task< List< Dto > > GetAllEntriesAsync< Dto > ()
- Task< List< Dto > > GetAllEntriesAsync< Dto > (BHTransaction transaction)
- Task< List< T > > GetAllInactiveEntriesAsync ()
- Task< List< T >> GetAllInactiveEntriesAsync (BHTransaction transaction)
- Task< T?> GetEntryByldAsync (G ld)
- Task< T?> GetEntryByIdAsync (G Id, BHTransaction transaction)
- Task< Dto?> GetEntryByIdAsync< Dto > (G Id)
- Task< Dto?> GetEntryByIdAsync< Dto > (G Id, BHTransaction transaction)
- Task< T?> GetEntryAsyncWhere (Expression< Func< T, bool > > predicate)
- Task< T?> GetEntryAsyncWhere (Expression< Func< T, bool >> predicate, BHTransaction transaction)
- Task
   Dto?
   GetEntryAsyncWhere
   Dto
   (Expression
   Func
   T, bool
   > predicate)
- Task< Dto?> GetEntryAsyncWhere< Dto > (Expression< Func< T, bool > > predicate, BHTransaction transaction)

- Task< List< T >> GetEntriesAsyncWhere (Expression< Func< T, bool >> predicate)
- Task< List< T >> GetEntriesAsyncWhere (Expression< Func< T, bool >> predicate, BHTransaction transaction)
- Task< List< Dto >> GetEntriesAsyncWhere< Dto > (Expression< Func< T, bool >> predicate)
- Task< List< Dto > SetEntriesAsyncWhere< Dto > (Expression< Func< T, bool > > predicate, BHTransaction transaction)
- Task< G?> InsertEntryAsync (T entry)
- Task< G?> InsertEntryAsync (T entry, BHTransaction transaction)
- Task< List< G?> > InsertEntriesAsync (List< T > entries)
- Task < List < G?> > InsertEntriesAsync (List < T > entries, BHTransaction transaction)
- Task< bool > UpdateEntryByIdAsync (T entry)
- Task< bool > UpdateEntryByIdAsync (T entry, BHTransaction transaction)
- Task< bool > UpdateEntryByIdAsync< Columns > (T entry)
- Task< bool > UpdateEntryByIdAsync< Columns > (T entry, BHTransaction transaction)
- Task< bool > UpdateEntriesByldAsync (List< T > entries)
- Task< bool > UpdateEntriesByIdAsync (List< T > entries, BHTransaction transaction)
- Task< bool > UpdateEntriesByIdAsync< Columns > (List< T > entries)
- $\bullet \ \, \mathsf{Task} \! < \mathsf{bool} > \mathsf{UpdateEntriesByIdAsync} \! < \mathsf{Columns} > (\mathsf{List} \! < \mathsf{T} > \mathsf{entries}, \mathsf{BHTransaction} \, \mathsf{transaction})$
- Task< bool > UpdateEntriesAsyncWhere (Expression< Func< T, bool > > predicate, T entry)
- Task< bool > UpdateEntriesAsyncWhere (Expression< Func< T, bool > > predicate, T entry, BHTransaction transaction)
- Task< bool > UpdateEntriesAsyncWhere< Columns > (Expression< Func< T, bool > > predicate, Columns entry)
- Task< bool > UpdateEntriesAsyncWhere< Columns > (Expression< Func< T, bool > > predicate, Columns entry, BHTransaction transaction)
- Task< bool > DeleteAllEntriesAsync ()
- Task< bool > DeleteAllEntriesAsync (BHTransaction transaction)
- Task< bool > DeleteEntryByldAsync (G ld)
- Task< bool > DeleteEntryByldAsync (G Id, BHTransaction transaction)
- Task< bool > DeleteInactiveEntryByIdAsync (G Id)
- Task < bool > DeleteInactiveEntryByIdAsync (G Id, BHTransaction transaction)
- Task< bool > ReactivateEntryByIdAsync (G Id)
- Task< bool > ReactivateEntryByldAsync (G Id, BHTransaction transaction)
- Task< bool > DeleteEntriesAsyncWhere (Expression< Func< T, bool > > predicate)
- Task< bool > DeleteEntriesAsyncWhere (Expression< Func< T, bool > > predicate, BHTransaction transaction)
- G? GetIdWhere (Expression < Func < T, bool > > predicate)
- G? GetIdWhere (Expression< Func< T, bool > > predicate, BHTransaction transaction)
- List< G > GetIdsWhere (Expression< Func< T, bool > > predicate)
- List< G > GetIdsWhere (Expression< Func< T, bool > > predicate, BHTransaction transaction)
- Task< G?> GetIdAsyncWhere (Expression< Func< T, bool > > predicate)
- Task< G?> GetIdAsyncWhere (Expression< Func< T, bool >> predicate, BHTransaction transaction)
- Task< List< G >> GetIdsAsyncWhere (Expression< Func< T, bool >> predicate)
- Task< List< G >> GetIdsAsyncWhere (Expression< Func< T, bool >> predicate, BHTransaction transaction)
- JoinsData< Dto, T, TOther > InnerJoin< TOther, Tkey, Dto > (Expression< Func< T, Tkey > > key, Expression< Func< TOther, Tkey > > otherKey)
- JoinsData< Dto, T, TOther > OuterJoin< TOther, Tkey, Dto > (Expression< Func< T, Tkey > > key, Expression< Func< TOther, Tkey > > otherKey)
- JoinsData< Dto, T, TOther > LeftJoin< TOther, Tkey, Dto > (Expression< Func< T, Tkey > > key, Expression< Func< TOther, Tkey > > otherKey)
- JoinsData< Dto, T, TOther > RightJoin< TOther, Tkey, Dto > (Expression< Func< T, Tkey > > key, Expression< Func< TOther, Tkey > > otherKey)

## 8.21.1 Detailed Description

For custom commands, use IBHConnection Interface

T	Type of Entity
G	Type of Entity Id

Example: IBHDataProvider<Customers,int>\_customerService = new BHDataProvider<Customers,int>();

**Type Constraints** 

T: BlackHoleEntity<G>

#### 8.21.2 Member Function Documentation

# 8.21.2.1 GetAllEntries() [1/2]

```
List< T > BlackHole.Core.IBHDataProvider< T, G >.GetAllEntries ( )
```

Gets all the entries of the specific Table and returns an IList of Entities.

Returns

All Active Entities of the Table

#### Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
List<Customer>customers = _customerService.GetAllEntries();
```

#### 8.21.2.2 GetAllEntries() [2/2]

```
List< T > BlackHole.Core.IBHDataProvider< T, G >.GetAllEntries ( BHTransaction transaction)
```

Transaction.Gets all the entries of the specific Table and returns an IList of Entities.

Returns

All Active Entities of the Table

#### Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    List<Customer>customers = _customerService.GetAllEntries(transaction);
}
```

## 8.21.2.3 GetAllEntries < Dto >() [1/2]

```
List< Dto > BlackHole.Core.IBHDataProvider< T, G > .GetAllEntries< Dto > ( )
```

Selects only the columns of the specified Dto that exist on the Table and returns an IList of the Dto. Only the properties of the Dto that have the same name and type with some properties of the Entity will be returned. Unmatched properties will be null.

```
Dto Data Transfer Object
```

Returns

All Active Entities of the Table mapped to DTO

## Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
List<SubCustomer>customers = _customerService.GetAllEntries<SubCustomer>();
```

**Type Constraints** 

Dto: BlackHoleDto<G>

## 8.21.2.4 GetAllEntries < Dto >() [2/2]

Transaction. Selects only the columns of the specified Dto that exist on the Table and returns an IList of the Dto. Only the properties of the Dto that have the same name and type with some properties of the Entity will be returned. Unmatched properties will be null.

**Template Parameters** 

```
Dto Data Transfer Object
```

Returns

All Active Entities of the Table mapped to DTO

#### Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    List<SubCustomer>customers = _customerService.GetAllEntries<SubCustomer>(transaction);
}
```

**Type Constraints** 

Dto: BlackHoleDto<G>

#### 8.21.2.5 GetAllInactiveEntries() [1/2]

```
List< T > BlackHole.Core.IBHDataProvider< T, G > .GetAllInactiveEntries ( )
```

In case you are using the 'UseActivator' Attribute on the Entity this method will return an IList of the Inactive Entries in this Table.

Returns

All Incative Entities of the Table

## Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
List<Customer>customers = _customerService.GetAllInactiveEntries();
```

## 8.21.2.6 GetAllInactiveEntries() [2/2]

```
List< T > BlackHole.Core.IBHDataProvider< T, G >.GetAllInactiveEntries ( BHTransaction transaction )
```

Transaction.In case you are using the 'UseActivator' Attribute on the Entity this method will return an IList of the Inactive Entries in this Table.

Returns

All Incative Entities of the Table

## Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    List<Customer>customers = _customerService.GetAllInactiveEntries(transaction);
}
```

## 8.21.2.7 GetEntryByld() [1/2]

Returns the Entity from this Table that has the specified Id.

#### **Parameters**

Id Specified Id

Returns

Entity

## Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
Customer? customer = _customerService.GetEntryById(Guid Id);
```

# 8.21.2.8 GetEntryByld() [2/2]

```
T? BlackHole.Core.IBHDataProvider< T, G >.GetEntryById ( G Id, BHTransaction transaction)
```

Transaction.Returns the Entity from this Table that has the specified Id.

#### Parameters

```
Id Specified Id
```

Returns

Entity

## Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    Customer? customer = _customerService.GetEntryById(Guid Id, transaction);
}
```

## 8.21.2.9 GetEntryByld< Dto >() [1/2]

```
Dto? BlackHole.Core.IBHDataProvider< T, G >.GetEntryById< Dto > ( G \it{Id} )
```

Selects only the columns of the specified Dto that exist on the Table and returns a Dto of the Entity with the specified Id.

```
Dto Data Transfer Object
```

#### **Parameters**

```
Id Specified Id
```

#### Returns

Data Transfer Object

#### Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
SubCustomer? customer = _customerService.GetEntryById<SubCustomer>(Guid Id);
```

## **Type Constraints**

Dto: BlackHoleDto<G>

# 8.21.2.10 GetEntryByld< Dto >() [2/2]

Transaction. Selects only the columns of the specified Dto that exist on the Table and returns a Dto of the Entity with the specified Id.

## **Template Parameters**

```
Dto Data Transfer Object
```

## **Parameters**

```
Id Specified Id
```

## Returns

Data Transfer Object

## Example:

IBHDataProvider<Customer,Guid> \_customerService = new BHDataProvider<Customer,Guid>();

```
using(BHTransaction transaction = new BHTransaction())
{
    SubCustomer? customers = _customerService.GetEntryById<SubCustomer>(Guid Id,transaction);
}
```

**Type Constraints** 

Dto: BlackHoleDto<G>

# 8.21.2.11 GetEntryWhere() [1/2]

Generates an Sql command using the Lambda Expression, that filters the Entries of the table and returns the first one that matches the filters.

#### **Parameters**

predicate Lambda Expression

Returns

Entity

## Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
Customer? customer = _customerService.GetEntryWhere(x=>x.Name == 'george');
```

# 8.21.2.12 GetEntryWhere() [2/2]

Transaction.Generates an Sql command using the Lambda Expression, that filters the Entries of the table and returns the first one that matches the filters.

#### **Parameters**

predicate | Lambda Expression

Returns

Entity

# Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    Customer? customer = _customerService.GetEntryWhere(x => x.Name == 'george',transaction')}
```

## 8.21.2.13 GetEntryWhere < Dto >() [1/2]

Generates an Sql command using the Lambda Expression and the Dto properties that match with the Entity properties. Returns the Dto columns of the first Entry that satisfies these filters.

# **Template Parameters**

```
Dto Data Transfer Object
```

### **Parameters**

```
predicate Lambda Expression
```

Returns

Data Transfer Object

#### Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
SubCustomer? customer = _customerService.GetEntryWhere<SubCustomer>(x=>x.Name == 'george');
```

**Type Constraints** 

Dto: BlackHoleDto<G>

## 8.21.2.14 GetEntryWhere < Dto >() [2/2]

Transaction.Generates an Sql command using the Lambda Expression and the Dto properties that match with the Entity properties. Returns the Dto columns of the first Entry that satisfies these filters.

**Template Parameters** 

```
Dto Data Transfer Object
```

#### **Parameters**

```
predicate | Lambda Expression
```

Returns

Data Transfer Object

### Example:

```
IBHDataProvider<Customer, Guid> _customerService = new BHDataProvider<Customer, Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    SubCustomer? custmers = _customerService.GetEntryWhere<SubCustomer>(x=>x.Name == 'george')}
```

**Type Constraints** 

Dto: BlackHoleDto<G>

# 8.21.2.15 GetEntriesWhere() [1/2]

```
List< T > BlackHole.Core.IBHDataProvider< T, G >.GetEntriesWhere ( Expression< Func< T, bool > predicate)
```

Generates an Sql command using the Lambda Expression, that filters the Entries of the table and returns all Entries that match the filters.

#### **Parameters**

```
predicate Lambda Expression
```

Returns

IList of Entities

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
List<Customer> customers = _customerService.GetEntriesWhere(x=>x.Name == 'george');
```

### 8.21.2.16 GetEntriesWhere() [2/2]

Transaction.Generates an Sql command using the Lambda Expression, that filters the Entries of the table and returns all Entries that match the filters.

#### **Parameters**

```
predicate Lambda Expression
```

### Returns

**IList of Entities** 

### Example:

```
IBHDataProvider<Customer, Guid> _customerService = new BHDataProvider<Customer, Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    List<Customer> customers = _customerService.GetEntriesWhere(x=>x.Name == 'george', trans}
```

### 8.21.2.17 GetEntriesWhere < Dto >() [1/2]

Generates an Sql command using the Lambda Expression, that filters the Entries of the table and returns all Columns that match with the filters and the Dto properties.

# **Template Parameters**

```
Dto Data Transfer Object
```

# Parameters

```
predicate Lambda Expression
```

## Returns

IList of DTOs

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
List<Subcustomer> customers = _customerService.GetEntriesWhere<SubCustomer>(x=>x.Name =='getEntriesWhere)
```

**Type Constraints** 

Dto: BlackHoleDto<G>

# 8.21.2.18 GetEntriesWhere < Dto >() [2/2]

Transaction.Generates an Sql command using the Lambda Expression, that filters the Entries of the table and returns all Columns that match with the filters and the Dto properties.

# **Template Parameters**

```
Dto Data Transfer Object
```

#### **Parameters**

predicate Lambda Expression

Returns

IList of DTOs

### Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    List<SubCustomer> customers = _customerService.GetEntriesWhere<SubCustomer>(x=>x.Name == )
```

**Type Constraints** 

Dto: BlackHoleDto<G>

# 8.21.2.19 InsertEntry() [1/2]

```
G? BlackHole.Core.IBHDataProvider< T, G >.InsertEntry ( T entry )
```

Inserts the Entity into the table, generates a new Id and returns the Id.

```
entry Entity
```

Returns

Id of the Entity

## Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
Guid? Id = _customerService.InsertEntry(customer);
```

# 8.21.2.20 InsertEntry() [2/2]

Transaction. Inserts the Entity into the table, generates a new Id and returns the Id.

## Parameters

```
entry Entity
```

Returns

Id of the Entity

# Example:

```
IBHDataProvider<Customer, Guid> _customerService = new BHDataProvider<Customer, Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    Guid? Id = _customerService.InsertEntry(customer, transaction);
}
```

# 8.21.2.21 InsertEntries() [1/2]

```
List< G?> BlackHole.Core.IBHDataProvider< T, G >.InsertEntries ( List< T > entries )
```

Inserts a list of Entities into the table, generates a new Id of each one and returns the list of Ids.

```
entries Entities
```

Returns

Ids of the Entities

#### Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
List<Guid> Ids = _customerService.InsertEntries(List<Customer> customers);
```

### 8.21.2.22 InsertEntries() [2/2]

```
List< G?> BlackHole.Core.IBHDataProvider< T, G >.InsertEntries ( List< T > entries, BHTransaction transaction)
```

Transaction. Inserts a list of Entities into the table, generates a new Id of each one and returns the list of Ids.

## **Parameters**

```
entries Entities
```

Returns

Ids of the Entities

#### Example:

```
IBHDataProvider<Customer, Guid> _customerService = new BHDataProvider<Customer, Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    List<Guid> Ids = _customerService.InsertEntries(List<Customer> customers,transaction);
}
```

# 8.21.2.23 UpdateEntryByld() [1/2]

```
bool BlackHole.Core.IBHDataProvider< T, G >.UpdateEntryById ( T entry )
```

Finds the entry in the table that has the same Id with the input's Entity and updates all the columns based on the Entity's property values. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

```
entry Entity
```

# Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
bool success = _customerService.UpdateEntryById(customer);
```

# 8.21.2.24 UpdateEntryByld() [2/2]

Transaction. Finds the entry in the table that has the same Id with the input's Entity and updates all the columns based on the Entity's property values. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

#### **Parameters**

```
entry Entity
```

### Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    bool success = _customerService.UpdateEntryById(customer, transaction);
}
```

# 8.21.2.25 UpdateEntryByld< Columns >() [1/2]

```
bool BlackHole.Core.IBHDataProvider< T, G >.UpdateEntryById< Columns > ( T entry )
```

Finds the entry in the database table that has the same Id with the input's Entity and using a 'Columns' class that has properties with the same name and type with some properties of the Entity, to specifically update these columns on the database entry. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

## **Template Parameters**

Columns Class with Properties that match with some of the Entity's properties

```
entry Entity
```

# Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
bool success = _customerService.UpdateEntryById<SubCustomer>( customer);
```

# **Type Constraints**

Columns: class

# 8.21.2.26 UpdateEntryByld< Columns >() [2/2]

Transaction. Finds the entry in the database table that has the same Id with the input's Entity and using a 'Columns' class that has properties with the same name and type with some properties of the Entity, to specifically update these columns on the database entry. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

### **Template Parameters**

Columns Class with Properties that match with some of the Entity's properties

## **Parameters**



# Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    List<Customer>custmers = _customerService.GetAllEntries(transaction);
}
```

### **Type Constraints**

Columns : class

### 8.21.2.27 UpdateEntriesByld() [1/2]

```
bool BlackHole.Core.IBHDataProvider< T, G >.UpdateEntriesById ( List< T > entries )
```

Finds the entries in the table that have the same Id with the input's Entities and updates all the columns based on each Entity's property values. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

#### **Parameters**

```
entries List of Entities
```

# Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
bool success = _customerService.UpdateEntriesById(List<Customer> customers);
```

### 8.21.2.28 UpdateEntriesByld() [2/2]

```
bool BlackHole.Core.IBHDataProvider< T, G >.UpdateEntriesById ( List < T > entries, \\ BHTransaction \ transaction )
```

Transaction. Finds the entries in the table that have the same Id with the input's Entities and updates all the columns based on each Entity's property values. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

#### **Parameters**

```
entries List of Entities
```

### Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    bool success = _customerService.UpdateEntriesById(List<Customer> customers, transaction)}
```

# 8.21.2.29 UpdateEntriesByld< Columns >() [1/2]

```
bool BlackHole.Core.IBHDataProvider< T, G >.UpdateEntriesById< Columns > ( List< T > entries )
```

Finds the entries in the database table that has the same Id with the input's Entities and using a 'Columns' class that has properties with the same name and type with some properties of the Entity, to specifically update these columns on each database entry. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

### **Template Parameters**

Columns Class with	Properties that match with some of the Entity's properties
--------------------	--

#### **Parameters**

```
entries List of Entities
```

#### Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
bool success = _customerService.UpdateEntriesById<SubCustomer>( List<Customer> customers);
```

# **Type Constraints**

Columns: class

# 8.21.2.30 UpdateEntriesByld< Columns >() [2/2]

```
bool BlackHole.Core.IBHDataProvider< T, G >.UpdateEntriesById< Columns > ( List< T > entries, BHTransaction transaction)
```

Transaction. Finds the entries in the database table that has the same Id with the input's Entities and using a 'Columns' class that has properties with the same name and type with some properties of the Entity, to specifically update these columns on each database entry. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

### **Template Parameters**

```
Columns Class with Properties that match with some of the Entity's properties
```

#### **Parameters**

```
entries List of Entities
```

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
   bool success = _customerService.UpdateEntriesById<SubCustomer>(List<Customer> customers,
```

}

# **Type Constraints**

Columns: class

# 8.21.2.31 UpdateEntriesWhere() [1/2]

Finds the entries in the table using a Lambda Expression as filter and updates all the columns based on the inserted Entity's property values. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

#### **Parameters**

predicate	Lambda Expression
entry	Entity

# Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
bool success = _customerService.UpdateEntriesWhere(x=>x.Name == 'george' , customer);
```

# 8.21.2.32 UpdateEntriesWhere() [2/2]

Transaction. Finds the entries in the table using a Lambda Expression as filter and updates all the columns based on the inserted Entity's property values. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

#### **Parameters**

predicate	Lambda Expression
entry	Entity

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    bool success = _customerService.UpdateEntriesWhere(x=>x.Name == 'george' , customer, transaction)}
```

#### 8.21.2.33 UpdateEntriesWhere < Columns >() [1/2]

Finds the entries in the database table using a Lambda Expression as filter and uses a 'Columns' class that has properties with the same name and type with some properties of the Entity, to specifically update these columns on each database entry with the Columns Object's values. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

### **Template Parameters**

Columns   Class with Properties that match with some of the Entity's properties
---

#### **Parameters**

predicate	Lambda Expression
entry	Columns Object

## Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
bool success = _customerService.UpdateEntriesWhere<SubCustomer>(x=>x.Name == 'george' , SubCustomer)
```

### **Type Constraints**

Columns: class

# 8.21.2.34 UpdateEntriesWhere < Columns >() [2/2]

Transaction. Finds the entries in the database table using a Lambda Expression as filter and uses a 'Columns' class that has properties with the same name and type with some properties of the Entity, to specifically update these columns on each database entry with the Columns Object's values. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

# **Template Parameters**

Columns	Class with Properties that match with some of the Entity's properties
---------	---

#### **Parameters**

predicate	Lambda Expression
entry	Columns Object

#### Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    bool success = _customerService.UpdateEntriesWhere<SubCustomer>(x=>x.Name == 'george' ,
}
```

#### **Type Constraints**

Columns: class

# 8.21.2.35 DeleteAllEntries() [1/2]

```
bool BlackHole.Core.IBHDataProvider< T, G >.DeleteAllEntries ( )
```

Deletes All entires of the database table. If you are using a 'UseActivator' Attribute on this Entity, the entries get deactivated instead of deleted and they can only be accessed with the 'GetInactiveEntries' command. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

### Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
bool success = _customerService.DeleteAllEntries();
```

# 8.21.2.36 DeleteAllEntries() [2/2]

```
bool BlackHole.Core.IBHDataProvider<br/>< T, G >.DeleteAllEntries ( {\tt BHTransaction}\ transaction\ )
```

Transaction.Deletes All entires of the database table. If you are using a 'UseActivator' Attribute on this Entity, the entries get deactivated instead of deleted and they can only be accessed with the 'GetInactiveEntries' command. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    bool success = _customerService.DeleteAllEntries(transaction);
}
```

### 8.21.2.37 DeleteEntryByld() [1/2]

```
bool BlackHole.Core.IBHDataProvider< T, G >.DeleteEntryById ( G \mathit{Id} )
```

Finds and deletes the entry of the database table that has the same Id as the input. If you are using a 'UseActivator' Attribute on this Entity, the entry gets deactivated instead of deleted and it can only be accessed with the 'GetInactiveEntries' command. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

#### **Parameters**

```
Id Entry's Id
```

### Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
bool success = _customerService.DeleteEntryById(Guid Id);
```

# 8.21.2.38 DeleteEntryByld() [2/2]

Transaction. Finds and deletes the entry of the database table that has the same Id as the input. If you are using a 'UseActivator' Attribute on this Entity, the entry gets deactivated instead of deleted and it can only be accessed with the 'GetInactiveEntries' command. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

### **Parameters**

```
Id Entry's Id
```

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    bool success = _customerService.DeleteEntryById(Guid Id, transaction);
}
```

### 8.21.2.39 DeleteInactiveEntryByld() [1/2]

If you are using a 'UseActivator' Attribute on this Entity It finds the entry in the database table that is Inactive and has the same Id as the input and permanently deletes it from the database.

#### **Parameters**

```
Id Inactive Entry's Id
```

# Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
bool success = _customerService.DeleteInactiveEntryById(Guid Id);
```

### 8.21.2.40 DeleteInactiveEntryById() [2/2]

```
bool BlackHole.Core.IBHDataProvider< T, G >.DeleteInactiveEntryById ( G Id, BHTransaction transaction)
```

Transaction. If you are using a 'UseActivator' Attribute on this Entity It finds the entry in the database table that is Inactive and has the same Id as the input and permanently deletes it from the database.

#### **Parameters**

```
Id Inactive Entry's Id
```

# Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    bool success = _customerService.DeleteInactiveEntryById(Guid Id, transaction);
}
```

# 8.21.2.41 ReactivateEntryByld() [1/2]

Transaction. Activates again an Inactive Entry in the database.

<b>D</b>					
Pа	ra	m	ല	aı	r۹

ld	
----	--

Returns

#### Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
bool success = _customerService.ReactivateEntryById( Guid Id);
```

### 8.21.2.42 ReactivateEntryByld() [2/2]

```
bool BlackHole.Core.IBHDataProvider< T, G >.ReactivateEntryById ( G Id, BHTransaction transaction)
```

Activates again an Inactive Entry in the database.

## **Parameters**



Returns

### Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    bool success = _customerService.ReactivateEntryById(Guid Id, transaction);
}
```

### 8.21.2.43 DeleteEntriesWhere() [1/2]

Finds and deletes the entries of the database table that match with the Lambda Expression filters. If you are using a 'UseActivator' Attribute on this Entity, the entries get deactivated instead of deleted and they can only be accessed with the 'GetInactiveEntries' command. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

```
predicate Lambda Expression
```

# Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
bool success = _customerService.DeleteEntriesWhere(x=> x.Name == 'george');
```

# 8.21.2.44 DeleteEntriesWhere() [2/2]

Transaction. Finds and deletes the entries of the database table that match with the Lambda Expression filters. If you are using a 'UseActivator' Attribute on this Entity, the entries get deactivated instead of deleted and they can only be accessed with the 'GetInactiveEntries' command. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

#### **Parameters**

```
predicate Lambda Expression
```

## Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
   bool success = _customerService.DeleteEntriesWhere(x=> x.Name == 'george' ,transaction);
}
```

# 8.21.2.45 GetAllEntriesAsync() [1/2]

```
Task < List < T >> BlackHole.Core.IBHDataProvider < T, G >.GetAllEntriesAsync ()
```

Asyncronous. Gets all the entries of the specific Table and returns an IList of Entities.

## Returns

All Active Entities of the Table

```
IBHDataProvider<Customer, Guid> _customerService = new BHDataProvider<Customer, Guid>();
List<Customer> customers = await _customerService.GetAllEntriesAsync();
```

### 8.21.2.46 GetAllEntriesAsync() [2/2]

```
\label{task} Task < List < T >> BlackHole.Core.IBHDataProvider < T, G >.GetAllEntriesAsync ( \\ BHTransaction \ transaction )
```

Transaction. Asyncronous. Gets all the entries of the specific Table and returns an IList of Entities.

Returns

All Active Entities of the Table

### Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    List<Customer> customers = await _customerService.GetAllEntriesAsync(transaction);
}
```

# 8.21.2.47 GetAllEntriesAsync< Dto >() [1/2]

```
{\tt Task<\ List<\ Dto\ >>\ BlackHole.Core.IBHDataProvider<\ T,\ G\ >.GetAllEntriesAsync<\ Dto\ >\ (\ )}
```

Asyncronous. Selects only the columns of the specified Dto that exist on the Table and returns an IList of the Dto. Only the properties of the Dto that have the same name and type with some properties of the Entity will be returned. Unmatched properties will be null.

**Template Parameters** 

```
Dto Data transfer Object
```

Returns

All Active Entities of the Table mapped to DTO

# Example:

```
IBHDataProvider<Customer, Guid> _customerService = new BHDataProvider<Customer, Guid>();
List<SubCustomer> customers = await _customerService.GetAllEntriesAsync<SubCustomer>();
```

**Type Constraints** 

Dto: BlackHoleDto<G>

### 8.21.2.48 GetAllEntriesAsync< Dto >() [2/2]

```
\label{total decomposition} $$ Task< List< Dto >> BlackHole.Core.IBHDataProvider< T, G >.GetAllEntriesAsync< Dto > ( BHTransaction transaction)
```

Transaction. Asyncronous. Selects only the columns of the specified Dto that exist on the Table and returns an IList of the Dto. Only the properties of the Dto that have the same name and type with some properties of the Entity will be returned. Unmatched properties will be null.

#### **Template Parameters**

Dto	Data transfer Object
-----	----------------------

#### Returns

All Active Entities of the Table mapped to DTO

### Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    List<SubCustomer> customers = _customerService.GetAllEntriesAsync<SubCustomer> (transaction)}
```

### **Type Constraints**

Dto: BlackHoleDto<G>

# 8.21.2.49 GetAllInactiveEntriesAsync() [1/2]

```
{\tt Task<\ List<\ T>> BlackHole.Core.IBHDataProvider<\ T,\ G>.GetAllInactiveEntriesAsync\ (\ )}
```

Asyncronous. In case you are using the 'UseActivator' Attribute on the Entity this method will return an IList of the Inactive Entries in this Table.

# Returns

All Incative Entities of the Table

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
List<Customer> custmers = await _customerService.GetAllInactiveEntriesAsync();
```

### 8.21.2.50 GetAllInactiveEntriesAsync() [2/2]

```
\label{task} Task < List < T >> BlackHole.Core.IBHDataProvider < T, G >. GetAllInactiveEntriesAsync ( \\ BHTransaction \ transaction )
```

Transaction. Asyncronous. In case you are using the 'UseActivator' Attribute on the Entity this method will return an IList of the Inactive Entries in this Table.

Returns

All Incative Entities of the Table

### Example:

```
IBHDataProvider<Customer, Guid> _customerService = new BHDataProvider<Customer, Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    List<Customer> customers = await _customerService.GetAllInactiveEntriesAsync(transaction)}
```

# 8.21.2.51 GetEntryByldAsync() [1/2]

```
Task< T?> BlackHole.Core.IBHDataProvider< T, G >.GetEntryByIdAsync ( G Id )
```

Asyncronous. Returns the Entity from this Table that has the specified Id.

#### **Parameters**



Returns

Entity

## Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
Customer? customer = await _customerService.GetEntryByIdAsync(Guid Id);
```

## 8.21.2.52 GetEntryByldAsync() [2/2]

```
Task< T?> BlackHole.Core.IBHDataProvider< T, G >.GetEntryByIdAsync ( G Id, BHTransaction transaction)
```

Transaction. Asyncronous. Returns the Entity from this Table that has the specified Id.

Id Specified Id

Returns

Entity

# Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    Customer? customer = await _customerService.GetEntryByIdAsync(Guid Id, transaction);
}
```

## 8.21.2.53 GetEntryByldAsync< Dto >() [1/2]

```
Task< Dto?> BlackHole.Core.IBHDataProvider< T, G >.GetEntryByIdAsync< Dto > ( G \it{Id} )
```

Asyncronous. Selects only the columns of the specified Dto that exist on the Table and returns a Dto of the Entity with the specified Id.

# **Template Parameters**

```
Dto Data Transfer Object
```

### **Parameters**

Id Specified Id

Returns

Data Transfer Object

# Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
SubCustomer? customer = await _customerService.GetEntryByIdAsync<SubCustomer>(Guid Id);
```

# **Type Constraints**

Dto: BlackHoleDto<G>

### 8.21.2.54 GetEntryByldAsync< Dto >() [2/2]

```
Task< Dto?> BlackHole.Core.IBHDataProvider< T, G >.GetEntryByIdAsync< Dto > ( G Id, BHTransaction transaction)
```

Transaction. Asyncronous. Selects only the columns of the specified Dto that exist on the Table and returns a Dto of the Entity with the specified Id.

#### **Template Parameters**

Dto Data Transfer Object	t
--------------------------	---

### **Parameters**

```
Id Specified Id
```

## Returns

Data Transfer Object

# Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    SubCustomer? customer = await _customerService.GetEntryByIdAsync<SubCustomer>(Guid Id, to)
}
```

# **Type Constraints**

Dto: BlackHoleDto<G>

## 8.21.2.55 GetEntryAsyncWhere() [1/2]

```
\label{task} Task < T?> BlackHole.Core.IBHDataProvider < T, G >.GetEntryAsyncWhere ( \\ Expression < Func < T, bool >> predicate )
```

Asyncronous. Generates an Sql command using the Lambda Expression, that filters the Entries of the table and returns the first one that matches the filters.

# **Parameters**

```
predicate Lambda Expression
```

Returns

Entity

# Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
Customer? customer = await _customerService.GetEntryAsyncWhere(x => x.Name == 'george');
```

## 8.21.2.56 GetEntryAsyncWhere() [2/2]

```
Task< T?> BlackHole.Core.IBHDataProvider< T, G >.GetEntryAsyncWhere ( Expression< Func< T, bool > > predicate, BHTransaction transaction )
```

Transaction. Asyncronous. Generates an Sql command using the Lambda Expression, that filters the Entries of the table and returns the first one that matches the filters.

#### **Parameters**

```
predicate Lambda Expression
```

Returns

Entity

# Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    Customer? customer = await _customerService.GetEntryAsyncWhere(x=> x.Name =='george', t:
}
```

# 8.21.2.57 GetEntryAsyncWhere< Dto >() [1/2]

Asyncronous. Generates an Sql command using the Lambda Expression and the Dto properties that match with the Entity properties. Returns the Dto columns of the first Entry that satisfies these filters.

## **Template Parameters**

```
Dto Data Transfer Object
```

predicate Lambda Expression

Returns

Data Transfer Object

## Example:

```
 \begin{tabular}{ll} $\tt IBHDataProvider<Customer,Guid>\_customerService=new BHDataProvider<Customer,Guid>(); \\ \tt SubCustomer?\ customer=await\_customerService.GetEntryAsyncWhere<SubCustomer>(\ x=>\ x.Name) \\ \end{tabular}
```

**Type Constraints** 

Dto: BlackHoleDto<G>

### 8.21.2.58 GetEntryAsyncWhere < Dto >() [2/2]

```
Task< Dto?> BlackHole.Core.IBHDataProvider< T, G >.GetEntryAsyncWhere< Dto > ( Expression< Func< T, bool > > predicate, BHTransaction transaction )
```

Transaction. Asyncronous. Generates an Sql command using the Lambda Expression and the Dto properties that match with the Entity properties. Returns the Dto columns of the first Entry that satisfies these filters.

# **Template Parameters**

Dto Data Transfer Object

## **Parameters**

predicate | Lambda Expression

Returns

Data Transfer Object

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    SubCustomer? customer = await _customerService.GetEntryAsyncWhere<SubCustomer>(x => x.Na);
}
```

**Type Constraints** 

Dto: BlackHoleDto<G>

# 8.21.2.59 GetEntriesAsyncWhere() [1/2]

```
\label{total core.} Task < List < T >> Black Hole. Core. IBHData Provider < T, G >. Get Entries Async Where ( Expression < Func < T, bool >> predicate )
```

Asyncronous. Generates an Sql command using the Lambda Expression, that filters the Entries of the table and returns all Entries that match the filters.

#### **Parameters**

```
predicate | Lambda Expression
```

Returns

IList of Entities

# Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
List<Customer> customers = await _customerService.GetEntriesAsyncWhere(x => x.Name =='georgetentriesAsyncWhere)
```

# 8.21.2.60 GetEntriesAsyncWhere() [2/2]

```
Task< List< T > BlackHole.Core.IBHDataProvider< T, G >.GetEntriesAsyncWhere ( Expression< Func< T, bool > predicate, BHTransaction transaction )
```

Transaction. Asyncronous. Generates an Sql command using the Lambda Expression, that filters the Entries of the table and returns all Entries that match the filters.

## **Parameters**

```
predicate Lambda Expression
```

Returns

IList of Entities

# 8.21.2.61 GetEntriesAsyncWhere < Dto >() [1/2]

Asyncronous. Generates an Sql command using the Lambda Expression, that filters the Entries of the table and returns all Columns that match with the filters and the Dto properties.

#### **Template Parameters**

```
Dto Data Transfer Object
```

#### **Parameters**

predicate Lambda Expression

#### Returns

IList of DTOs

# Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
List<SubCustomer> customers = await _customerService.GetEntriesAsyncWhere<SubCustomer>(x =>
```

#### **Type Constraints**

Dto: BlackHoleDto<G>

### 8.21.2.62 GetEntriesAsyncWhere < Dto >() [2/2]

Transaction. Asyncronous. Generates an Sql command using the Lambda Expression, that filters the Entries of the table and returns all Columns that match with the filters and the Dto properties.

# **Template Parameters**

```
Dto Data Transfer Object
```

#### **Parameters**

```
predicate Lambda Expression
```

### Returns

IList of DTOs

### Example:

```
IBHDataProvider<Customer, Guid> _customerService = new BHDataProvider<Customer, Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    List<SubCustomer> customers = await _customerService.GetEntriesAsyncWhere<SubCustomer>();
}
```

# **Type Constraints**

Dto: BlackHoleDto<G>

# 8.21.2.63 InsertEntryAsync() [1/2]

Asyncronous. Inserts the Entity into the table, generates a new Id and returns the Id.

#### **Parameters**

```
entry Entity
```

## Returns

Id of the Entity

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
Guid? Id = await _customerService.InsertEntryAsync( customer);
```

### 8.21.2.64 InsertEntryAsync() [2/2]

Transaction. Asyncronous. Inserts the Entity into the table, generates a new Id and returns the Id.

#### **Parameters**

```
entry Entity
```

#### Returns

Id of the Entity

# Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    Guid? Id = await _customerService.InsertEntryAsync(customer, transaction);
}
```

# 8.21.2.65 InsertEntriesAsync() [1/2]

```
\label{eq:task} Task < List < G? > BlackHole.Core.IBHDataProvider < T, G > .InsertEntriesAsync ( \\ List < T > entries )
```

Asyncronous. Inserts a list of Entities into the table, generates a new Id of each one and returns the list of Ids.

### **Parameters**

```
entries List of Entities
```

#### Returns

lds of the Entities

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
List<Guid> Ids = await _customerService.InsertEntriesAsync(List<Customer> customers);
```

### 8.21.2.66 InsertEntriesAsync() [2/2]

```
Task< List< G?> > BlackHole.Core.IBHDataProvider< T, G >.InsertEntriesAsync ( List< T > entries, BHTransaction transaction )
```

Transaction. Asyncronous. Inserts a list of Entities into the table, generates a new Id of each one and returns the list of Ids.

#### **Parameters**

```
entries List of Entities
```

#### Returns

Ids of the Entities

### Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    List<Guid> Ids = await _customerService.InsertEntriesAsync(List<Customer> customers, transaction)
```

# 8.21.2.67 UpdateEntryByldAsync() [1/2]

Asyncronous. Finds the entry in the table that has the same Id with the input's Entity and updates all the columns based on the Entity's property values. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

### **Parameters**

```
entry Entity
```

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
bool success = await _customerService.UpdateEntryByIdAsync(customer);
```

### 8.21.2.68 UpdateEntryByldAsync() [2/2]

Transaction. Asyncronous. Finds the entry in the table that has the same Id with the input's Entity and updates all the columns based on the Entity's property values. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

#### **Parameters**

```
entry Entity
```

#### Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    bool success = await _customerService.UpdateEntryByIdAsync(customer, transaction);
}
```

# 8.21.2.69 UpdateEntryByldAsync< Columns >() [1/2]

```
\label{total condition} {\tt Task<\ bool\ > BlackHole.Core.IBHDataProvider<\ T,\ G >.UpdateEntryByIdAsync<\ Columns > (} \\ {\tt T\ entry\ )}
```

Asyncronous. Finds the entry in the database table that has the same Id with the input's Entity and using a 'Columns' class that has properties with the same name and type with some properties of the Entity, to specifically update these columns on the database entry. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

## **Template Parameters**

Columns Class with Properties that match with some of the Entity's properties

### **Parameters**

```
entry Entity
```

### Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
bool success = await _customerService.UpdateEntryByIdAsync<SubCustomer>( customer);
```

# **Type Constraints**

# Columns : class

# 8.21.2.70 UpdateEntryByldAsync< Columns >() [2/2]

Transaction.Asyncronous. Finds the entry in the database table that has the same Id with the input's Entity and using a 'Columns' class that has properties with the same name and type with some properties of the Entity, to specifically update these columns on the database entry. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

# **Template Parameters**

Columns Class with Properties that match with some of the Entity's properties

#### **Parameters**

```
entry Entity
```

# Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    bool success = await _customerService.UpdateEntryByIdAsync<SucCustomer>(customer, transaction)
```

# **Type Constraints**

Columns: class

# 8.21.2.71 UpdateEntriesByldAsync() [1/2]

```
Task<br/>< bool > BlackHole.Core.IBHDataProvider<br/>< T, G >.UpdateEntriesByIdAsync ( List<br/>< T > entries )
```

Asyncronous. Finds the entries in the table that have the same Id with the input's Entities and updates all the columns based on each Entity's property values. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

# **Parameters**

```
entries List of Entities
```

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
bool success = await _customerService.UpdateEntriesByIdAsync( List<Customer> customers);
```

# 8.21.2.72 UpdateEntriesByldAsync() [2/2]

```
\label{eq:task_problem} $\sf Task< bool > \sf BlackHole.Core.IBHDataProvider< T, G >. UpdateEntriesByIdAsync ($\sf List< T > entries, $\sf BHTransaction transaction )$
```

Transaction.Asyncronous. Finds the entries in the table that have the same Id with the input's Entities and updates all the columns based on each Entity's property values. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

### **Parameters**

```
entries List of Entities
```

#### Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
   bool success = await _customerService.UpdateEntriesByIdAsync(List<Customer> customers, t
}
```

### 8.21.2.73 UpdateEntriesByldAsync< Columns >() [1/2]

Asyncronous. Finds the entries in the database table that has the same Id with the input's Entities and using a 'Columns' class that has properties with the same name and type with some properties of the Entity, to specificaly update these columns on each database entry. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

## **Template Parameters**

#### **Parameters**

```
entries List of Entities
```

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
bool success = await _customerService.UpdateEntriesByIdAsync<SubCustomer>(List<Customer> customer> customer
```

**Type Constraints** 

Columns: class

# 8.21.2.74 UpdateEntriesByldAsync< Columns >() [2/2]

Transaction.Asyncronous. Finds the entries in the database table that has the same Id with the input's Entities and using a 'Columns' class that has properties with the same name and type with some properties of the Entity, to specifically update these columns on each database entry. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

**Template Parameters** 

Columns | Class with Properties that match with some of the Entity's properties

#### **Parameters**

```
entries List of Entities
```

### Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    bool success = await _customerService.UpdateEntriesByIdAsync<SubCustomer>(List<Customer)}</pre>
```

**Type Constraints** 

Columns : class

# 8.21.2.75 UpdateEntriesAsyncWhere() [1/2]

Asyncronous. Finds the entries in the table using a Lambda Expression as filter and updates all the columns based on the inserted Entity's property values. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

predicate	Lambda Expression
entry	Entity

#### Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
bool success = await _customerService.UpdateEntriesAsyncWhere(x => x.Name =='george', customerService)
```

### 8.21.2.76 UpdateEntriesAsyncWhere() [2/2]

Transaction.Asyncronous. Finds the entries in the table using a Lambda Expression as filter and updates all the columns based on the inserted Entity's property values. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

#### **Parameters**

predicate	Lambda Expression
entry	Entity

# Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
   bool success = await _customerService.UpdateEntriesAsyncWhere(x => x.Name == 'george', or )
```

### 8.21.2.77 UpdateEntriesAsyncWhere < Columns >() [1/2]

Asyncronous. Finds the entries in the database table using a Lambda Expression as filter and uses a 'Columns' class that has properties with the same name and type with some properties of the Entity, to specifically update these columns on each database entry with the Columns Object's values. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

# **Template Parameters**

	Columns	Class with Properties that match with some of the Entity's properties	
--	---------	---	--

#### **Parameters**

predicate	Lambda Expression
entry	Columns Object

### Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
bool success = await _customerService.UpdateEntriesAsyncWhere<SubCustomer>(x => x.Name ==' quality for the customer fo
```

### **Type Constraints**

Columns: class

# 8.21.2.78 UpdateEntriesAsyncWhere < Columns >() [2/2]

Transaction.Asyncronous. Finds the entries in the database table using a Lambda Expression as filter and uses a 'Columns' class that has properties with the same name and type with some properties of the Entity, to specificaly update these columns on each database entry with the Columns Object's values. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

#### **Template Parameters**

Columns	Class with Properties that match with some of the Entity's properties
---------	---

#### **Parameters**

predicate	Lambda Expression
entry	Columns Object

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    bool success = await _customerService.UpdateEntriesAsyncWhere<SubCustomer>(x x=> x.Name)}
```

**Type Constraints** 

Columns: class

#### 8.21.2.79 DeleteAllEntriesAsync() [1/2]

```
Task< bool > BlackHole.Core.IBHDataProvider< T, G >.DeleteAllEntriesAsync ( )
```

Asyncronous. Deletes All entires of the database table. If you are using a 'UseActivator' Attribute on this Entity, the entries get deactivated instead of deleted and they can only be accessed with the 'GetInactiveEntries' command. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

## Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
bool success = await _customerService.DeleteAllEntriesAsync();
```

## 8.21.2.80 DeleteAllEntriesAsync() [2/2]

Transaction.Asyncronous. Deletes All entires of the database table. If you are using a 'UseActivator' Attribute on this Entity, the entries get deactivated instead of deleted and they can only be accessed with the 'GetInactiveEntries' command. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

## Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    bool success = await _customerService.DeleteAllEntriesAsync(transaction);
}
```

# 8.21.2.81 DeleteEntryByldAsync() [1/2]

```
\label{eq:task_bool} {\tt Task<\ bool\ > BlackHole.Core.IBHDataProvider<\ T,\ G\ >.DeleteEntryByIdAsync\ (} \\ {\tt G\ Id\ )}
```

Asyncronous. Finds and deletes the entry of the database table that has the same Id as the input. If you are using a 'UseActivator' Attribute on this Entity, the entry gets deactivated instead of deleted and it can only be accessed with the 'GetInactiveEntries' command. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

#### **Parameters**

```
Id Entry's Id
```

## Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
bool success = await _customerService.DeleteEntryByIdAsync(Guid Id);
```

## 8.21.2.82 DeleteEntryByldAsync() [2/2]

```
Task<br/>< bool > BlackHole.Core.IBHDataProvider<br/>< T, G >.DeleteEntryByIdAsync ( G Id, BHTransaction transaction )
```

Transaction.Asyncronous. Finds and deletes the entry of the database table that has the same Id as the input. If you are using a 'UseActivator' Attribute on this Entity, the entry gets deactivated instead of deleted and it can only be accessed with the 'GetInactiveEntries' command. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

#### **Parameters**

```
Id Entry's Id
```

## Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    bool success = _customerService.DeleteEntryByIdAsync(Guid Id, transaction);
}
```

# 8.21.2.83 DeleteInactiveEntryByldAsync() [1/2]

Asyncronous. If you are using a 'UseActivator' Attribute on this Entity It finds the entry in the database table that is Inactive and has the same Id as the input and permanently deletes it from the database.

## **Parameters**

```
Id Inactive Entry's Id
```

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
bool success = await _customerService.DeleteInactiveEntryByIdAsync(Guid Id);
```

## 8.21.2.84 DeleteInactiveEntryByldAsync() [2/2]

```
Task<br/>< bool > BlackHole.Core.IBHDataProvider<br/>< T, G >.DeleteInactiveEntryByIdAsync ( G Id, BHTransaction transaction )
```

Transaction. Asyncronous. If you are using a 'UseActivator' Attribute on this Entity It finds the entry in the database table that is Inactive and has the same Id as the input and permanently deletes it from the database.

#### **Parameters**

```
Id Inactive Entry's Id
```

## Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
   bool success = await _customerService.DeleteInactiveEntryByIdAsync(Guid Id, transaction)}
```

# 8.21.2.85 ReactivateEntryByldAsync() [1/2]

```
Task<br/> bool > BlackHole.Core.IBHDataProvider<br/>< T, G >.ReactivateEntryByIdAsync ( G \mathit{Id} )
```

Transaction. Asyncronous. Activates again an Inactive Entry in the database.

## **Parameters**



Returns

IBHDataProvider<Customer,Guid> \_customerService = new BHDataProvider<Customer,Guid>();
bool success = await \_customerService.ReactivateEntryByIdAsync(Guid Id);

# 8.21.2.86 ReactivateEntryByldAsync() [2/2]

Asyncronous. Activates again an Inactive Entry in the database.

#### **Parameters**



Returns

## Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    bool success = await _customerService.ReactivateEntryByIdAsync(Guid Id, transaction);
}
```

## 8.21.2.87 DeleteEntriesAsyncWhere() [1/2]

```
\label{total condition} Task < bool > BlackHole.Core.IBHDataProvider < T, G > .DeleteEntriesAsyncWhere ( \\ Expression < Func < T, bool > > predicate )
```

Asyncronous. Finds and deletes the entries of the database table that match with the Lambda Expression filters. If you are using a 'UseActivator' Attribute on this Entity, the entries get deactivated instead of deleted and they can only be accessed with the 'GetInactiveEntries' command. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

#### **Parameters**

```
predicate Lambda Expression
```

# Example:

IBHDataProvider<Customer,Guid> \_customerService = new BHDataProvider<Customer,Guid>();

```
bool success = await _customerService.DeleteEntriesAsyncWhere( x => x.Name =='george');
```

## 8.21.2.88 DeleteEntriesAsyncWhere() [2/2]

```
\label{total condition}  \mbox{Task< bool} > \mbox{BlackHole.Core.IBHDataProvider< T, G >.DeleteEntriesAsyncWhere (} \\ \mbox{Expression< Func< T, bool} > > \mbox{predicate,} \\ \mbox{BHTransaction } transaction )
```

Transaction.Asyncronous. Finds and deletes the entries of the database table that match with the Lambda Expression filters. If you are using a 'UseActivator' Attribute on this Entity, the entries get deactivated instead of deleted and they can only be accessed with the 'GetInactiveEntries' command. !!Important!! => You must use 'await' operator if your next operation depends on this operation.

#### **Parameters**

```
predicate Lambda Expression
```

## Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    bool success = await _customerService.DeleteEntriesAsyncWhere( x => x.Name =='george',transaction)}
```

# 8.21.2.89 GetIdWhere() [1/2]

Finds the active entries of the database table that match with the Lambda Expression filters and returns the ld of the first entry.

## **Parameters**

```
predicate Lambda Expression
```

## Returns

Id of the Entry

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
Guid? Id = _customerService.GetIdWhere(x => x.Name == 'george');
```

# 8.21.2.90 GetIdWhere() [2/2]

Transaction. Finds the active entries of the database table that match with the Lambda Expression filters and returns the Id of the first entry.

## **Parameters**

```
predicate Lambda Expression
```

#### Returns

Id of the Entry

## Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    Guid? Id = _customerService.GetIdWhere(x => x.Name == 'george'transaction);
}
```

## 8.21.2.91 GetIdsWhere() [1/2]

Finds the active entries of the database table that match with the Lambda Expression filters and returns their lds.

## **Parameters**

```
predicate Lambda Expression
```

# Returns

List of Entry Ids

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
List<Guid> Ids = _customerService.GetIdsWhere(x => x.Name == 'george');
```

# 8.21.2.92 GetIdsWhere() [2/2]

Transaction. Finds the active entries of the database table that match with the Lambda Expression filters and returns their lds.

#### **Parameters**

```
predicate Lambda Expression
```

#### Returns

List of Entry Ids

## Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    List<Guid> Ids = _customerService.GetIdsWhere(x => x.Name == 'george',transaction);
}
```

## 8.21.2.93 GetIdAsyncWhere() [1/2]

```
\label{eq:core.IBHDataProvider} $$T$, $G>.$GetIdAsyncWhere ($$Expression< Func< T, bool>> predicate )$
```

Asyncronous. Finds the active entries of the database table that match with the Lambda Expression filters and returns the ld of the first entry.

## **Parameters**

```
predicate | Lambda Expression
```

# Returns

Id of the Entry

```
 \begin{tabular}{ll} $\tt IBHDataProvider < Customer, Guid> \_customerService = new BHDataProvider < Customer, Guid> (); \\ \tt Guid? Id = await \_customerService.GetIdAsyncWhere( x => x.Name == 'george'); \\ \end{tabular}
```

## 8.21.2.94 GetIdAsyncWhere() [2/2]

Transaction. Asyncronous. Finds the active entries of the database table that match with the Lambda Expression filters and returns the ld of the first entry.

#### **Parameters**

predicate Lambda Expression

#### Returns

Id of the Entry

# Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    Guid? Id = await _customerService.GetIdAsyncWhere(x=> x.Name =='george',transaction);
}
```

## 8.21.2.95 GetIdsAsyncWhere() [1/2]

Asyncronous. Finds the active entries of the database table that match with the Lambda Expression filters and returns their lds.

## **Parameters**

predicate Lambda Expression

Returns

List of Entry Ids

## Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
List<Guid> Ids = await _customerService.GetIdsAsyncWhere(x => x.Name=='george');
```

## 8.21.2.96 GetIdsAsyncWhere() [2/2]

```
Task< List< G >> BlackHole.Core.IBHDataProvider< T, G >.GetIdsAsyncWhere ( Expression< Func< T, bool >> predicate, BHTransaction transaction )
```

Transaction. Asyncronous. Finds the active entries of the database table that match with the Lambda Expression filters and returns their lds.

#### **Parameters**

```
predicate Lambda Expression
```

Returns

List of Entry Ids

## Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
using(BHTransaction transaction = new BHTransaction())
{
    List<Guid> Ids = await _customerService.GetIdsAsyncWhere(x => x.Name=='george',transaction)}
```

# 8.21.2.97 InnerJoin< TOther, Tkey, Dto >()

Starts a Joins sequence, with the first one as 'Inner Join' that can be continued with the BlackHole.Core.BHExtensions methods.

# **Template Parameters**

TOth	er	The other Table as Entity
Tk	ey	The type of their joint column
D	)to	The exported Data Transfer Object Class

## **Parameters**

key	Column of this Table	
otherKey	Column of the Other Table	

## Returns

The Calculated Data of this Joins Part

## Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
List<CustomerOrderDTO> customers = _customerService.InnerJoin<Order, Guid, CustomerOrderDTO>
```

## **Type Constraints**

Dto: BlackHoleDto

TOther : BlackHoleEntity Tkey : IComparable

# 8.21.2.98 OuterJoin< TOther, Tkey, Dto >()

Starts a Joins sequence, with the first one as 'Outer Join' that can be continued with the BlackHole.Core.BHExtensions methods.

# **Template Parameters**

TOther	The other Table as Entity
Tkey	The type of their joint column
Dto	The exported Data Transfer Object Class

## **Parameters**

key	Column of this Table	
otherKey	Column of the Other Table	

#### Returns

The Calculated Data of this Joins Part

## Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
List<CustomerOrderDTO> customers = _customerService.OuterJoin<Order, Guid, CustomerOrderDTO>
```

## **Type Constraints**

Dto: BlackHoleDto

TOther: BlackHoleEntity Tkey: IComparable

## 8.21.2.99 LeftJoin< TOther, Tkey, Dto >()

Starts a Joins sequence, with the first one as 'Left Join' that can be continued with the BlackHole.Core.BHExtensions methods.

## **Template Parameters**

TOther	The other Table as Entity
Tkey	The type of their joint column
Dto	The exported Data Transfer Object Class

## Parameters

key	Column of this Table	
otherKey	Column of the Other Table	

## Returns

The Calculated Data of this Joins Part

## Example:

```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
List<CustomerOrderDTO> customers = _customerService.LeftJoin<Order, Guid, CustomerOrderDTO>
```

## **Type Constraints**

Dto: BlackHoleDto

TOther : BlackHoleEntity Tkey : IComparable

# 8.21.2.100 RightJoin< TOther, Tkey, Dto >()

Starts a Joins sequence, with the first one as 'Right Join' that can be continued with the BlackHole.Core.BHExtensions methods.

## **Template Parameters**

TOther	The other Table as Entity
Tkey	The type of their joint column
Dto	The exported Data Transfer Object Class

#### **Parameters**

key	Column of this Table
otherKey	Column of the Other Table

## Returns

The Calculated Data of this Joins Part

## Example:

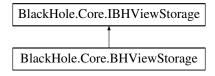
```
IBHDataProvider<Customer,Guid> _customerService = new BHDataProvider<Customer,Guid>();
List<CustomerOrderDTO> customers = _customerService.RightJoin<Order, Guid, CustomerOrderDTO>
```

## **Type Constraints**

Dto: BlackHoleDto
TOther: BlackHoleEntity
Tkey: IComparable

# 8.22 BlackHole.Core.IBHViewStorage Interface Reference

Inheritance diagram for BlackHole.Core.IBHViewStorage:



## **Public Member Functions**

- List< Dto > ExecuteView< Dto > ()
- List< Dto > ExecuteView< Dto > (BHTransaction transaction)
- Task< List< Dto > > ExecuteViewAsync< Dto > ()
- Task< List< Dto > > ExecuteViewAsync< Dto > (BHTransaction transaction)

# 8.22.1 Detailed Description

An Interface that contains the functionality to execute the Stored Views.

It is automatically injected into IServiceCollection on the startup and it can be used with Dependency Injection. It is required in order to use BHViewStorage methods.

## 8.22.2 Member Function Documentation

## 8.22.2.1 ExecuteView < Dto >() [1/2]

```
List< Dto > BlackHole.Core.IBHViewStorage.ExecuteView< Dto > ( )
```

Executes the stored view that has the inserted DTO as Identifier. If there is no view stored with this DTO it returns an empty IList.

**Template Parameters** 

```
Dto Class that the view will be mapped
```

#### Returns

List of the DTO

## Example:

```
IBHViewStorage viewStorage = new BHViewStorage();
List<OrdersDtoG> ordersJoin = viewStorage.ExecuteView<OrdersDtoG>();
```

Find: How to store a view here BlackHole.Core.BHExtensions.StoreAsView<Dto>

**Tip:** If the parameters of the view are always the same in your application, you can make a method to store all the views on the startup.

This will increase performace, as the Joins calculations will only be performed once.

**Type Constraints** 

Dto: BlackHoleDto

## 8.22.2.2 ExecuteView < Dto >() [2/2]

```
List< Dto > BlackHole.Core.IBHViewStorage.ExecuteView< Dto > ( {\tt BHTransaction}\ transaction\ )
```

Transaction. Executes the stored view that has the inserted DTO as Identifier. If there is no view stored with this DTO it returns an empty IList.

## **Template Parameters**

Dto Class that the view will be mapped

#### Returns

List of the DTO

## Example:

```
IBHViewStorage viewStorage = new BHViewStorage();
using(BHTransaction transaction = new BHTransaction())
{
    List<OrdersDtoG> ordersJoin = viewStorage.ExecuteView<OrdersDtoG> (transaction);
}
```

Find: How to store a view here BlackHole.Core.BHExtensions.StoreAsView<Dto>

**Tip:** If the parameters of the view are always the same in your application, you can make a method to store all the views on the startup.

This will increase performace, as the Joins calculations will only be performed once.

**Type Constraints** 

Dto: BlackHoleDto

## 8.22.2.3 ExecuteViewAsync< Dto >() [1/2]

```
{\tt Task} < {\tt List} < {\tt Dto} > > {\tt BlackHole.Core.IBHViewStorage.ExecuteViewAsync} < {\tt Dto} > (\ )
```

Asyncronous. Executes the stored view that has the inserted DTO as Identifier. If there is no view stored with this DTO it returns an empty IList.

#### **Template Parameters**

Dto Class that the view will be mapped

Returns

List of the DTO

## Example:

```
IBHViewStorage viewStorage = new BHViewStorage();
List<OrdersDtoG> ordersJoin = await viewStorage.ExecuteViewAsync<OrdersDtoG>();
```

Find: How to store a view here BlackHole.Core.BHExtensions.StoreAsView<Dto>

**Tip:** If the parameters of the view are always the same in your application, you can make a method to store all the views on the startup.

This will increase performace, as the Joins calculations will only be performed once.

**Type Constraints** 

Dto: BlackHoleDto

## 8.22.2.4 ExecuteViewAsync< Dto >() [2/2]

```
\label{total condition} $\sf Task< List< Dto >> BlackHole.Core.IBHViewStorage.ExecuteViewAsync< Dto > ($\sf BHTransaction transaction )$
```

Asyncronous. Transaction. Executes the stored view that has the inserted DTO as Identifier. If there is no view stored with this DTO it returns an empty IList.

**Template Parameters** 

Dto Class that the view will be mapped

Returns

List of the DTO

## Example:

```
IBHViewStorage viewStorage = new BHViewStorage();
using(BHTransaction transaction = new BHTransaction())
{
    List<OrdersDtoG> ordersJoin = await viewStorage.ExecuteViewAsync<OrdersDtoG> (transaction)}
```

Find: How to store a view here BlackHole.Core.BHExtensions.StoreAsView<Dto>

**Tip:** If the parameters of the view are always the same in your application, you can make a method to store all the views on the startup.

This will increase performace, as the Joins calculations will only be performed once.

**Type Constraints** 

Dto: BlackHoleDto

# 8.23 BlackHole.Entities.NotNullable Class Reference

Inherits Attribute.

# **Public Member Functions**

NotNullable ()

# 8.23.1 Detailed Description

It's an Attribute that can be used on the Entity's properties

It turns the property to a Non Nullable Column in the Table.

## 8.23.2 Constructor & Destructor Documentation

# 8.23.2.1 NotNullable()

```
BlackHole.Entities.NotNullable.NotNullable ( )
```

It turns the property to a Non Nullable Column in the Table.

## Example:

```
[NotNullable]
public decimal Price {get;set;}
```

# 8.24 BlackHole.Configuration.ServicesWithNamespace Class Reference

# **Public Member Functions**

- AssembliesUsed AddServicesFromNamespace (string servicesNamespace)
- AssembliesUsed AddServicesFromNamespaces (List< string > servicesNamespaces)
- void UseOtherAssembly (Assembly otherAssembly)

# 8.24.1 Detailed Description

Settings for the Services Namespaces that will be used Part of BlackHole.Configuration.ConnectionAdditionalSettings

## 8.24.2 Member Function Documentation

## 8.24.2.1 AddServicesFromNamespace()

```
\begin{tabular}{lll} Assemblies Used Black Hole. Configuration. Services With Namespace. Add Services From Namespace ( string services Namespace) \end{tabular}
```

Using only the services that are in the specified Namespace and inherit from BlackHole Services Classes.

#### **Parameters**

servicesNamespace	
-------------------	--

## Returns

BlackHole.Configuration.AssembliesUsed

## Example:

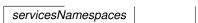
services.SuperNova(settings => settings.AddDatabase(connection => connection.UseMySql(connection)
.UseEntitiesInNamespaces(List<string> NamespacesFullNames).AddServicesFromNamespacesFullNames

## 8.24.2.2 AddServicesFromNamespaces()

```
AssembliesUsed BlackHole.Configuration.ServicesWithNamespace.AddServicesFromNamespaces (
List< string > servicesNamespaces)
```

Using only the services that are in the specified Namespaces and inherit from BlackHole Services Classes.

#### **Parameters**



## Returns

BlackHole.Configuration.AssembliesUsed

services.SuperNova(settings => settings.AddDatabase(connection => connection.UseMySql(connection)
.UseEntitiesInNamespaces(List<string> NamespacesFullNames).AddServicesFromNamespacesFullNames)

## 8.24.2.3 UseOtherAssembly()

```
\label{thm:configuration} \mbox{NervicesWithNamespace.UseOtherAssembly (} \\ \mbox{Assembly otherAssembly )}
```

Scans a specified assembly for BlackHole Entities and Services and uses only them.

#### **Parameters**

```
otherAssembly Full Assembly
```

## Example:

services.SuperNova(settings => settings.AddDatabase(connection => connection.UseMySql(connection)
.UseEntitiesInNamespaces(List<string> NamespacesFullNames).UseOtherAssembly

# 8.25 BlackHole.Entities.UseActivator Class Reference

Inherits Attribute.

## **Public Member Functions**

• UseActivator ()

# 8.25.1 Detailed Description

It's an Attribute that can be used on the Entity

Using this over a class, then the Column isActive of the entity will be used and instead of deleting the Entries, it will be setting them as inactive ,every time you preform a delete.

Inactive entries are ignored by all commands and can only be accessed with the methods 'GetAllInactiveEntries' and 'DeleteInactiveEntryById'

## 8.25.2 Constructor & Destructor Documentation

## 8.25.2.1 UseActivator()

```
BlackHole.Entities.UseActivator.UseActivator ( )
```

Using this over a class, then the Column isActive of the entity will be used and instead of deleting the Entries, it will be setting them as inactive ,every time you preform a delete.

Inactive entries are ignored by all commands and can only be accessed with the methods 'GetAllInactiveEntries' and 'DeleteInactiveEntryById'

## Example:

```
[UseActivator] public class Customer : BlackHoleEntity<int>
```

# 8.26 BlackHole.Entities.VarCharSize Class Reference

Inherits Attribute.

## **Public Member Functions**

VarCharSize (int Characters)

# 8.26.1 Detailed Description

It's an Attribute that can be used on the Entity's properties

Specifies the Size of a Varchar column in the database The default size is 255

# 8.26.2 Constructor & Destructor Documentation

# 8.26.2.1 VarCharSize()

Specifies the Size of a Varchar column in the database The default size is 255.

## **Parameters**

Characters The number of Characters. Varchar(n)

```
[VarCharSize(100)]
public string FirstName {get;set;}
```